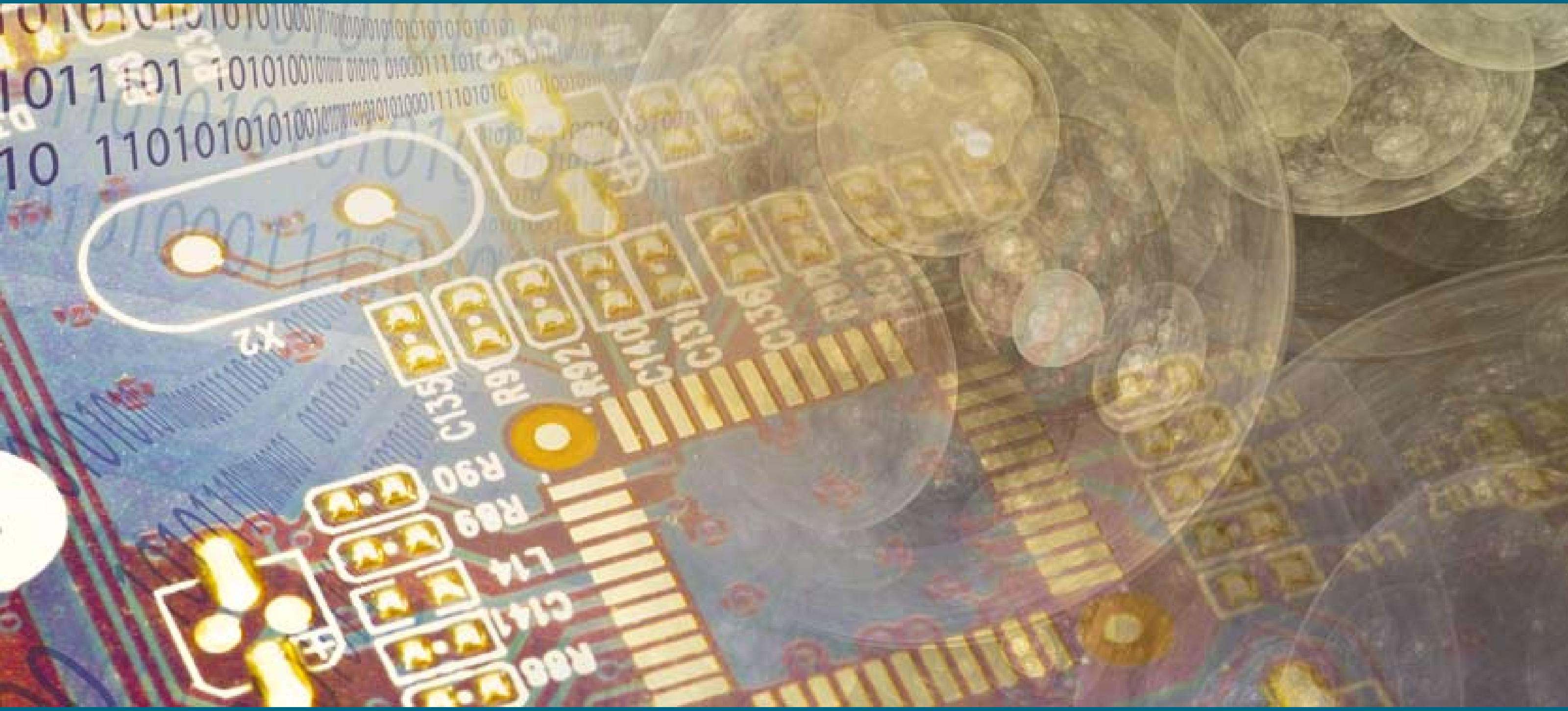
Journal of Artificial Evolution and Applications

Particle Swarms: The Second Decade

Guest Editors: Riccardo Poli, Jim Kennedy, Tim Blackwell, and Alex Freitas



Particle Swarms: The Second Decade

Journal of Artificial Evolution and Applications

Particle Swarms: The Second Decade

Guest Editors: Riccardo Poli, Jim Kennedy,
Tim Blackwell, and Alex Freitas

Contents

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 108972, 3 pages
doi:10.1155/2008/108972

Editorial

Particle Swarms: The Second Decade

Riccardo Poli,[1] Jim Kennedy,[2] Tim Blackwell,[3] and Alex Freitas[4]

[1] Department of Computing and Electronic Systems, University of Essex, Colchester CO4 3SQ, UK
[2] US Bureau of Labor Statistics, Washington, DC 20212 0001, USA
[3] Department of Computing, Goldsmiths College, University of London, London SE14 6NW, UK
[4] Computing Laboratory, The University of Kent, Canterbuy, Kent CT2 7NZ, UK

Correspondence should be addressed to Riccardo Poli, rpoli@essex.ac.uk

Received 7 April 2008; Accepted 7 April 2008

1. INTRODUCTION

Particle swarm optimisation (PSO) was born just over ten years ago. The initial ideas on particle swarms of Kennedy and Eberhart were aimed at producing computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities. The first simulations [1] were influenced by Heppner's and Grenander's work [2] and involved analogues of bird flocks searching for corn. These soon developed [1, 3, 4] into a powerful optimisation method—the particle swarm optimiser.

In PSO, a number of simple entities—the particles—are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved. Eventually, the swarm as a whole, like a flock of birds collectively foraging for food, moves close to an optimum of the fitness function.

The particle swarm is more than just a collection of particles. A particle by itself has almost no power to solve any problem; progress occurs only when the particles interact. Problem solving is a population-wide phenomenon, emerging from the individual behaviours of the particles through their interactions. In any case, populations are organised according to some sort of communication structure or topology, often thought of as a social network. Each particle communicates with some other particles and is affected by the best point found by any member of its topological neighbourhood. The potential kinds of topologies of social networks are hugely varied, but in practice certain types, such as rings and fully connected networks, have been used more frequently.

2. PARTICLE SWARM OPTIMISATION IS COMING OF AGE

The particle swarm paradigm, that was only a few years ago a curiosity, has now attracted the interest of researchers around the globe. In fact, simple queries in publication databases, such as IEEE Xplore and Google Scholar, reveal that the number of publications in particle swarm optimisation has been exponentially growing for the last few years.

The aim of this special issue is to attract papers on particularly innovative algorithms, speculative ideas, new theoretical approaches, and novel applications that could act as seeds for PSO research in its second decade. Topics we solicited included the following: novel empirical and theoretical analyses of PSO population dynamics; innovative studies and algorithms for setting PSO parameters; new adaptive and parameterless PSO; analyses and new proposals of social network topologies; PSOs for combinatorial and hierarchical search spaces; novel PSOs for dynamic problems, noisy functions, and multimodal functions; advanced barebones/distribution-based PSOs; unconventional hybrids of PSO with other techniques; as well as novel applications in engineering, biomedicine, clustering, classification, entertainment, finance, image and signal processing, graphics, computational intelligence, and robotics. Amazingly, as we will show in Section 3, we managed to attract high-quality submissions in almost all these areas.

As a first step in the direction of identifying new ideas for the second decade, we proposed and organised a workshop entitled “Particle Swarms: The Second Decade” at the Genetic and Evolutionary Computation Conference (GECCO) held in London in July 2007. Seven papers were selected by a peer-review process and presented at the workshop. The level of attendance was very good, the ideas presented were exciting, and the discussion that followed them was very lively.

Encouraged by this first success, we felt that we had to follow this on with a second, higher-impact initiative: this special issue of the Journal of Artificial Evolution and Applications. All of the presenters at the GECCO workshop were invited to extend their work and submit to this special issue, although the special issue was also open to new contributions. The special issue was a resounding success, attracting a total of 50 high-quality submissions. After a rigorous multistage review process, just under 20 papers were eventually accepted, and are now presented in this special issue. (Following the usual principles of blind review, papers coauthored by one of the guest editors were handled by another editor, so the former did not know which reviewers were assigned to his paper and had no influence on the editorial decision about the paper.)

3. THE PAPERS IN THIS SPECIAL ISSUE

We have chosen to divide the articles in this special issue into three main categories (although some spanned more than one): innovative theoretical/empirical analyses and theoretically sound design (Section 3.1), new exciting types of particle swarms (Section 3.2), and novel applications (Section 3.3). We will briefly introduce them in the next subsections.

3.1. Theory, analysis, and principled design of particle swarms

After a decade of relatively slow progress, the theory of PSO is now making significant and rapid progress, and this trend is likely to continue in the second decade. Indeed, four papers in this special issue deal with either theoretical explanations or theoretically driven design of PSOs.

The success or failure of stochastic optimisation algorithms depends on their ability to explore the search space associated with a problem effectively. The search is controlled by what is known as the *sampling distribution* of the optimiser. The article “Dynamics and stability of the sampling distribution of particle swarm optimisers via moment analysis” by R. Poli looks at the precise identification of the sampling distribution and its changes over time for standard forms of PSO. The article “Examination of particle tails” by T. Blackwell and D. Bratton focuses also on the characterisation of the way PSOs sample the search space, but, in this case, the analysis coarse-grains over the time variations of the distribution; the resulting distribution is shown to follow a power law.

One important source of innovation in PSO is the extension of the paradigm to the exploration of discrete search spaces. This is difficult because certain notions, such as the notion of velocity, are not easily extended to such spaces. So, until now, designing discrete PSOs has been essentially a black art. In this special issue, however, we are fortunate to have two articles—“Geometric particle swarm optimisation” by A. Moraglio et al. and “Forma analysis of particle swarm optimisation for permutation problems” by T. Gong and A. L. Tuson—which provide two different general approaches to derive theoretically sound PSOs for generic discrete search spaces.

3.2. Novel particle swarms

Broadly speaking, a particle swarm has two elements: particle dynamics and a mechanism for the sharing of information. The dynamics tells a particle how to move given the information that it has available; the result of this movement is subsequently communicated to other particles. These components, which mutually interact to deliver a swarm’s searching capability, can be modified in various ways to derive new particle swarms.

Novel particle dynamics are represented here by a number of papers. S. Kok and J. A. Snyman in “A strongly interacting dynamic particle swarm optimization method” consider how particle dynamics can be affected by both position (as in standard PSO) and function value at that position, thereby introducing gradient information explicitly into the update rules. In a novel approach, J. L. Fernández-Martínez and E. García-Gonzalo consider how PSOs can be derived from discretisations of continuous particle trajectories. Their paper “The generalized PSO: a new door to PSO evolution” presents theoretical stability analysis and experimental testing. A general trend in the work represented by these two papers is a growing interest in considering a swarm as an interacting system of “physical” particles, that is, as physical entities with continuous trajectories, momentum, energy dissipation, and force laws.

Two more papers consider alternative dynamics. In “Novel orthogonal momentum-type particle swarm optimization applied to solve large parameter optimization problems,” J.-L. Liu and C.-C. Chang introduce an alternative velocity update rules with a “delta momentum” rule. They combine this rule with fractional factorial design for efficient optimisation. In “A simplified recombinant PSO,” D. Bratton and T. Blackwell consider a series of simplifications to a biologically motivated dynamics based on genetic recombination. They demonstrate that velocity can be eliminated from the algorithm altogether, thereby making contact with distribution-based PSOs (“barebones” formulations).

The final two papers in this section consider how biological metaphors can induce changes to swarm algorithms at the population level. In “What else the evolution of PSO is telling us,” L. Diosan and M. Oltean suggest changes to the order and frequency by which particles are updated. They use a genetic algorithm to explore various strategies. In “Particle swarm optimization for multimodal functions: a clustering approach,” A. Passaro and A. Starita use k-means clustering to establish subswarm “niches.” This approach involves dynamic information topologies that are linked to local spatial behaviour, features that are surely likely to be the subject of much second decade work.

3.3. Applications

Particle swarm optimisation has been enormously successful at solving problems of practical interest. This is reflected in this special issue by approximately half the articles reporting novel and exciting applications for PSO. We briefly review them below. Before we start, however, we would like to emphasise that several of the particle swarms proposed in the articles mentioned in this section are discrete PSOs, a characteristic shared by two theoretical approaches mentioned in Section 3.1. This indicates how prolific and important the area of discrete PSOs is becoming. We expect this area to grow significantly in the second decade.

The article "Optimizing the operation sequence of a multi-head surface mounting machine using a discrete particle swarm optimization algorithm" by Y.-M. Chen and C.-T. Lin, for example, describes how PSO can be used to optimise the process of picking and placing components onto printed circuit boards. R. Armellin and M. Lavagna's article entitled "Multidisciplinary optimization of aerocapture maneuvers" shows how PSO can design maneuvers that ensure that a spacecraft is captured by the gravitational attraction of a planet, with minimal expenditure of energy.

In "A hybrid PSO/ACO algorithm for discovering classification rules in data mining," N. Holden and A. A. Freitas extend a PSO algorithm with ideas borrowed from ant colony optimisation in order to cope with nominal (nonnumerical) variables, and apply the proposed algorithm to a data mining problem. In the article "An improved particle swarm optimizer for placement constraints," S.-T. Hsieh et al. show how PSO can solve floor-planning problems very effectively. With their paper entitled "Inverse parameter identification technique using PSO algorithm applied to geotechnical modeling," J. Meier et al. show how the PSO can provide valuable solutions in the difficult area of inverse-problem solving.

In "Particle swarm optimization for antenna designs in engineering electromagnetics," N. Jin and Y. Rahmat-Samii propose a PSO tailored to antenna design in engineering. The proposed PSO can cope with both real and binary variables, as well as multiobjective problems. In "Generating complete bifurcation diagrams using a dynamic environment particle swarm optimization algorithm," J. Barrera et al. apply PSO to the analysis of dynamical systems, which are represented as a set of equations specifying how variables change over time. In "A discrete particle swarm optimization algorithm for uncapacitated facility location problem," A. R. Guner and M. Sevkli propose a discrete PSO, as well as a hybrid version with local search, for solving a combinatorial optimization problem. In "Particle swarm for attribute selection in Bayesian classification: an application to protein function prediction," E. S. Correa et al. propose another type of discrete PSO algorithm to the problem of selecting attributes in data mining, and apply the algorithm to a bioinformatics problem.

ACKNOWLEDGMENTS

We would like to thank the Editor-in-Chief, Stefano Cagnoni, for his support in putting together this special issue. Cecilia Di Chio and Dan Bratton are also warmly thanked for their help with the local organisation of the GECCO workshop from which this special issue eventually emerged. The many reviewers who generously volunteered their time to help us with this special issue are also thanked. Finally, we would like to thank EPSRC (Extended Particle Swarms project, GR/T11234/01) for financial support.

Riccardo Poli
Jim Kennedy
Tim Blackwell
Alex Freitas

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE Press, Perth, Australia, November-December 1995.

[2] H. Heppner and U. Grenander, "A stochastic non-linear model for coordinated bird flocks," in *The Ubiquity of Chaos*, S. Krasner, Ed., pp. 233–238, AAAS, Washington, DC, USA, 1990.

[3] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, IEEE Press, Nagoya, Japan, October 1995.

[4] R. Eberhart, P. K. Simpson, and R. W. Dobbins, *Computational Intelligence PC Tools*, Academic Press Professional, Boston, Mass, USA, 1996.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 761459, 10 pages
doi:10.1155/2008/761459

Research Article

Dynamics and Stability of the Sampling Distribution of Particle Swarm Optimisers via Moment Analysis

Riccardo Poli

Department of Computing and Electronic Systems, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

Correspondence should be addressed to Riccardo Poli, rpoli@essex.ac.uk

Received 23 July 2007; Accepted 3 December 2007

Recommended by T. Blackwell

For stochastic optimisation algorithms, knowing the probability distribution with which an algorithm allocates new samples in the search space is very important, since this explains how the algorithm really works and is a prerequisite to being able to match algorithms to problems. This is the only way to beat the limitations highlighted by the no-free lunch theory. Yet, the sampling distribution for velocity-based particle swarm optimisers has remained a mystery for the whole of the first decade of PSO research. In this paper, a method is presented that allows one to exactly determine all the characteristics of a PSO's sampling distribution and explain how it changes over time during stagnation (i.e., while particles are in search for a better personal best) for a large class of PSO's.

1. INTRODUCTION

Let us consider the basic form of PSO with inertia weight, which is controlled by the following two equations:

$$v_{t+1}^i = wv_t^i + \phi_1 \otimes (y^i - x_t^i) + \phi_2 \otimes (\hat{y} - x_t^i),$$
$$x_{t+1}^i = x_t^i + v_{t+1}^i, \tag{1}$$

where ϕ_1 and ϕ_2 represent vectors of random numbers uniformly distributed in $[0, c_1]$ and $[0, c_2]$, respectively, (c_1 and c_2 are nonnegative constants), $\otimes$ is component wise multiplication, y^i is the position of the ith particle's personal best, $\hat{y}$ is the position of the particle's neighbourhood best, and w is a constant in $[0, 1]$. Following Kennedy's graphical examinations of the trajectories of individual particles and their responses to variations in key parameters [1], traditionally, this model (and the related PSO with constriction [2], which is, provably, mathematically equivalent) has been studied theoretically under strong simplifying assumptions such as isolated single individuals, search stagnation (i.e., no improved solutions are found) and *absence of randomness* [2–12]. While these assumptions make it possible to make theoretical progress, they also make the results of the analyses difficult to carry over to the real PSO (where stochasticity is present).

Only very few attempts to understand the behaviour of the PSO in the *presence of stochasticity* have been made. For example, Clerc [13] analysed the distribution of velocities of one particle controlled by the standard PSO update rule with inertia and stochastic forces and was able to show that a particle's new velocity is the sum of three components: a forward force, a backward force, and noise. Kadirkamanathan et al. [14] were able to study the stability of particles in the presence of stochasticity by using Lyapunov stability analysis.

These are good and important first steps in the direction of modelling the real PSO. However, they only marginally scratch the surface in relation to one of the most important open problems in PSO research: the characterisation of the PSO's sampling distribution and its changes over time. Why is this question so fundamental? For two reasons. Firstly, this scientific knowledge gives us much greater understanding of an algorithm's behaviour. Secondly, on a more practical level, the only way to beat the limitations implied by NFL [15] is to match algorithms to problems. Naturally, at least in part, this can be achieved by a trial-and-error approach involving testing different algorithms and parameter settings on a particular problem or class of problems until a suitable match is found. A more analytic approach to this problem, however, requires two things: (a) knowing the search distribution of an algorithm and (b) checking whether this is well matched to the features of a problem's fitness landscape. Unfortunately, the sampling distribution of the PSO has been unavailable during the whole first decade (and beyond) of PSO research.

In recent work [16], we introduced a novel method, which allowed us to exactly determine the mean and standard deviation of the sampling distribution of the canonical PSO as well as their changes over any number of generations. The only assumption we made was stagnation, that is, we studied the sampling distribution produced by particles in search for a better personal best. In this paper, we generalise the technique and show how it can be used to determine any number of moments of the sampling distribution of PSO's.

The paper is organised as follows. In Section 2, we provide a summary of the results presented in [16]. In Section 3, we generalised the method to moments of any order. In Section 4, we apply the technique to compute the skewness and kurtosis of the canonical PSO's sampling distribution. In Section 5, we further extend the generality of our technique so as to cover a variety of PSO's, not just the one in (1). This allows us to compare, on a theoretical basis, different forms of PSO. In Section 6, we then use the moments of the PSO's sampling distribution to approximately reconstruct the distribution itself, making it possible, for the first time, to understand the search strategy implemented by the canonical PSO, while searching for new improvements. We discuss the results and conclude in Section 7.

2. DYNAMICS OF FIRST- AND SECOND ORDER MOMENTS OF THE PSO's SAMPLING DISTRIBUTION

2.1. *Derivation of dynamic equations for the moments*

During stagnation, each particle behaves independently. Also, each dimension is treated independently. So we can analyse each particle's behaviour in isolation. Therefore, we can drop the superscript i in (1), and rewrite them as

$$x_{t+1} = x_t(1+w) - x_t(\phi_1+\phi_2) - wx_{t-1} + \phi_1 y + \phi_2 \hat{y}, \quad (2)$$

where we used the relation $v_t = x_t - x_{t-1}$. Note that on the r.h.s. of this equation, while w, y, and $\hat{y}$ are constants, x_t, ϕ_1, ϕ_2, and x_{t-1} are stochastic variables. As a consequence, x_{t+1} on the l.h.s. of the equation is a stochastic variable too (in fact, it is a stochastic function of the stochastic variables appearing in the r.h.s.).

In [16], we applied the expectation operator to both sides of the equation obtaining

$$\begin{aligned} E[x_{t+1}] &= E[x_t](1+w) - E[x_t](E[\phi_1]+E[\phi_2]) \\ &\quad - wE[x_{t-1}] + E[\phi_1]y + E[\phi_2]\hat{y}, \end{aligned} \quad (3)$$

where we performed the substitution $E[x_t\phi_i] = E[x_t]E[\phi_i]$ because of the statistical independence between ϕ_i and x_t. Assuming that $c_1 = c_2 = c$, that is, the ϕ_1 and ϕ_2 are both uniformly distributed in $[0, c]$, we have $E[\phi_i] = c/2$, and so

$$E[x_{t+1}] = E[x_t](\rho - c) - wE[x_{t-1}] + c\frac{y+\hat{y}}{2}, \quad (4)$$

where we renamed$(1+w) = \rho$.

If we square both sides of (2), we obtain

$$x_{t+1}^2 = \left(x_t\rho - x_t\phi_1 - x_t\phi_2 - wx_{t-1} + \phi_1 y + \phi_2\hat{y}\right)^2. \quad (5)$$

Expanding the r.h.s., we obtain an equation expressing the stochastic variable x_{t+1}^2 as a function of other stochastic variables, that is, x_t^2, $x_{t-1}x_t$, ϕ_1^2, and so forth. By applying the expectation operator to both sides of this equation, one obtains

$$\begin{aligned} E[x_{t+1}^2] = {} & E[x_t^2](\rho^2 - 4\mu\rho + 2\nu + 2\mu^2) \\ & + E[x_{t-1}x_t](-2w\rho + 4w\mu) + E[x_{t-1}^2](w^2) \\ & + E[x_t](2\mu y\rho + 2\mu\hat{y}\rho - 2\nu y - 2\mu^2\hat{y} - 2\mu^2 y - 2\nu\hat{y}) \\ & + E[x_{t-1}](-2\mu y w - 2\mu\hat{y}w) + \nu y^2 + 2\mu^2 y\hat{y} + \nu\hat{y}^2, \end{aligned} \quad (6)$$

where we set $\mu = E[\phi_i] = c/2$ and $\nu = E[\phi_i^2] = c^2/3$, for brevity.

If, instead, we multiply both sides of (2) by x_t and apply the expectation operator, we obtain

$$E[x_{t+1}x_t] = E[x_t^2](\rho - c) - wE[x_t x_{t-1}] + E[x_t]c\frac{y+\hat{y}}{2}. \quad (7)$$

Equations (4), (6), and (7) form a system of coupled second-order difference equations using which, given appropriate initial conditions, one can compute $E[x_t]$, $E[x_t^2]$ and $E[x_t x_{t-1}]$ for any t.

2.2. *Analysis of the dynamics*

Equations (4), (6), and (7) can also be written in matrix notation as the extended first order system

$$\mathbf{z}(t+1) = M\mathbf{z}(t) + \mathbf{b}, \quad (8)$$

where

$$\begin{aligned} \mathbf{z}(t) &= \left(E[x_t] \ \ E[x_{t-1}] \ \ E[x_t^2] \ \ E[x_{t-1}^2] \ \ E[x_t x_{t-1}]\right)^T, \\ M &= \begin{pmatrix} \rho - c & -w & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ A1 & A2 & A3 & w^2 & 2w(2\mu - \rho) \\ 0 & 0 & 1 & 0 & 0 \\ cp & 0 & \rho - c & 0 & -w \end{pmatrix}, \\ \mathbf{b} &= \begin{pmatrix} cp \\ 0 \\ \nu y^2 + 2\mu^2 y\hat{y} + \nu\hat{y}^2 \\ 0 \\ 0 \end{pmatrix}, \end{aligned} \quad (9)$$

where $A1 = 4p(\mu\rho - \nu - \mu^2)$, $A2 = -4\mu wp$, $A3 = \rho^2 - 4\mu\rho + 2\nu + 2\mu^2$.

It is then trivial to verify under what conditions $E[x_t]$, $E[x_t^2]$, and $E[x_t x_{t-1}]$ will converge to stable fixed points. We need to have that all eigenvalues of M must be within the unit circle, that is, $\Lambda_m = \max_i|\lambda_i| < 1$. When this happens, we will say that the PSO is *order-2 stable*.

Naturally, when $\Lambda_m < 1$, in principle, we could derive (either symbolically or numerically) the fixed point for the system, which we will denote as $\mathbf{z}^*$. This would be simply given by

$$\mathbf{z}^* = (I - M)^{-1}\mathbf{b}. \tag{10}$$

When the system is order-2 stable, by the simple change of variables $\widetilde{\mathbf{z}}(t) = \mathbf{z}(t) - \mathbf{z}^*$, we can represent the dynamics of the system via the following linear homogeneous equation:

$$\widetilde{\mathbf{z}}(t+1) = M\widetilde{\mathbf{z}}(t), \tag{11}$$

which can trivially be integrated to obtain the following explicit solution for the dynamics of the first two moments:

$$\widetilde{\mathbf{z}}(t) = M^t\widetilde{\mathbf{z}}(0). \tag{12}$$

3. HIGHER-ORDER MOMENTS

In [16], we obtained recursions which describe the dynamics of first- and second-order moments of the sampling distribution of a standard PSO during stagnation, as summarised in the previous section. One may then wonder whether it would be possible to follow a similar approach to study the dynamics of higher-order moments.

3.1. Derivation of dynamic equations for higher-order moments

The fundamental question is "what quantities would we have to deal with if we took higher powers of both sides of (2) as we did to derive (5)?" Generally, the r.h.s. would be a sum of terms of the form

$$a_0 x_t^{a_1} x_{t-1}^{a_2} w^{a_3} \phi_1^{a_4} \phi_2^{a_5} y^{a_6} \hat{y}^{a_7}, \tag{13}$$

where a_k are suitable constants. Naturally, taking powers of (2) and then multiplying both sides by some power of x_t would also lead to equations involving terms such as those in (13). That is, for any choice of $b_1 \in \mathbb{N}$ and $b_2 \in \mathbb{N}$,

$$x_{t+1}^{b_1} x_t^{b_2} = \sum_i a_{0_i} x_t^{a_{1_i}} x_{t-1}^{a_{2_i}} w^{a_{3_i}} \phi_1^{a_{4_i}} \phi_2^{a_{5_i}} y^{a_{6_i}} \hat{y}^{a_{7_i}}, \tag{14}$$

where a_{k_i} are suitable constants. If we then take expectations for both sides, we obtain

$$E[x_{t+1}^{b_1} x_t^{b_2}] = \sum_i a_{0_i} w^{a_{3_i}} y^{a_{6_i}} \hat{y}^{a_{7_i}} E[\phi_1^{a_{4_i}}] E[\phi_2^{a_{5_i}}] E[x_t^{a_{1_i}} x_{t-1}^{a_{2_i}}], \tag{15}$$

where we used the independence of ϕ_1, ϕ_2, $x_t x_{t-1}$, and, of course, their powers. If ϕ_1 and ϕ_2 are uniformly distributed in the same range, namely, $[0, c]$, it is easy to verify that

$$E[\phi_j^n] = \frac{c^n}{n+1}. \tag{16}$$

So

$$E[x_{t+1}^{b_1} x_t^{b_2}] = \sum_i \omega_i E[x_t^{a_{1_i}} x_{t-1}^{a_{2_i}}], \tag{17}$$

where

$$\omega_i = \left(\frac{a_{0_i} w^{a_{3_i}} c^{a_{4_i}+a_{5_i}} y^{a_{6_i}} \hat{y}^{a_{7_i}}}{(1+a_{4_i})(1+a_{5_i})}\right). \tag{18}$$

It is important to note here that because (2) is linear in x_t and x_{t-1}, all the terms on the r.h.s. of (14) and (17) respect the relation $a_{1_i} + a_{2_i} \leq b_1 + b_2$. This implies that *it is possible to construct recursions for moments of arbitrary order*.

3.2. Complexity of the calculations

If one wanted to push the analysis developed in Section 2 up to order 3, one would need to instantiate (17) for $E[x_{t+1}^3]$, $E[x_{t+1}^2 x_t]$, $E[x_{t+1} x_t^2]$ and add the resulting equations to (4), (6), and (7). If one wanted to go to order 4, an additional set of four equations (for $E[x_{t+1}^4]$, $E[x_{t+1}^3 x_t]$, $E[x_{t+1}^2 x_t^2]$, and $E[x_{t+1} x_t^3]$) would be needed, bringing the total to 10.

More generally, in order to compute statistics of order n, one needs to construct and iterate

$$Q(n) = \frac{n \times (n+1)}{2} \tag{19}$$

second-order difference equations. Since, after expansion, the r.h.s. of (2) contains 7 atomic terms of the form in (13), the r.h.s. of (17) contains 7^{b_1} such terms (Note that the exponent b_2 does not influence the number of terms because the recursion for $E[x_{t+1}^{b_1} x_t^{b_2}]$ is obtained as follows: (a) we compute $x_{t+1}^{b_1}$, which is given by an expression containing 7^{b_1} terms; (b) we multiply each term by $x_t^{b_2}$, which changes the exponents a_{1_i} but does not alter the number of terms; (c) we apply the expectation operator, which again does not modify the number of terms.) So the total number of terms one needs to compute to construct the equations for order- n statistics is 7 (for the order 1 equations) plus $(7^2 + 7)$ (for the order 2 equations) plus $(7^3 + 7^2 + 7)$ (for the order 3 equations), and so forth. This gives us a total of

$$T(n) = \sum_{i=1}^{n} (n - i + 1) \times 7^i \tag{20}$$

terms. Note that $T(n)$ grows exponentially approximately as 1.36×7^n. So although the number of equations one needs to deal with grows quadratically, the computational effort required to instantiate them is exponential. For instance, $T(1) = 7$, $T(2) = 63$, $T(3) = 462$, $T(4) = 3262$, and $T(5) = 22869$. The growth in number of terms can be reduced if one makes explicit use of ρ (i.e., by adding the factor ρ^{a_8} in (13)). Then $T(n) = O(6^n)$. Either way, manually deriving equations for moments of order 3 is already very laborious. The process, however, is clearly mechanisable. This can be done using computer algebra systems, or by explicitly representing and manipulating the ω_i's in each equation (this is what we did). As a result of mechanisation, computing the equations for up to order 6 or 7 is feasible with an ordinary personal computer.

Some of the ω_i's in (17) present the same pattern of exponents for w, c, y, and $\hat{y}$, so terms can be collected leading to more compact equations (e.g., compare (5) and (6)). Also,

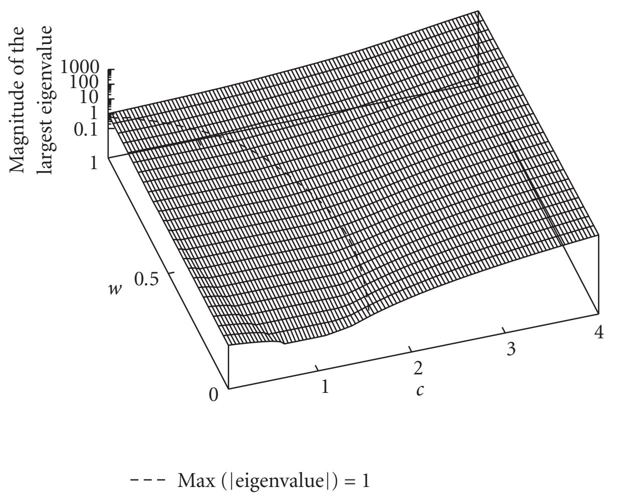


Figure 1: Magnitude of the largest eigenvalue of M_4 as a function of the parameters w and c. The curved line on the surface encloses the order-4 stable region.

given their size, one will normally want to study (e.g., integrate) (17) numerically. In this case, w, c, y, and $\widehat{y}$ are all numerical parameters. So the ω_i's become constants and, after collecting terms, each equation contains at most $Q(n)$ terms, which, as we know, is quadratic in the order n. As a result, although the complexity of the construction of the motion equations for the moments is exponential in the order of the moments, their numerical integration is only of order $O(n^4)$.

Naturally, the system of $Q(n)$ second-order difference equations necessary to predict the dynamics of moments of order 1 to n can be turned into a system of order 1 of the form in (8), via the choice

$$z(t) = \begin{pmatrix} E[x_t] \\ E[x_{t-1}] \\ E[x_t^2] \\ E[x_t x_{t-1}] \\ E[x_{t-1}^2] \\ E[x_t^3] \\ E[x_t^2 x_{t-1}] \\ E[x_t x_{t-1}^2] \\ E[x_{t-1}^3] \\ \vdots \\ E[x_{t-1}^n] \end{pmatrix}. \tag{21}$$

This effectively means adding artificial update equations of the form $E[x_t^k] = E[x_t^k]$ for $k = 1, \dots, n$, bringing the total to $Q'(n) = Q(n) + n$. The transition matrix for the system is, therefore, of size $Q'(n) \times Q'(n)$. We will denote this with M_n. For example, for $n = 4$, which would allow one to study the mean, variance, skewness, and kurtosis of the sampling distribution as a function of t, M_4 is merely a 14×14 matrix.

Interestingly, $Q'(n)$ grows so slowly that one can perform an eigenvalue analysis for any M_n that one is able to compute. That is, the expensive part of the process is the construction of M_n. Once this is done, iterating the system, establishing its stability, or finding its fixed points is a trivial matter.

In the next section, we provide results for statistics of order 3 and 4, that is, $n = 4$, a value of n for which computing M_n takes only a few seconds. However, before we do this, we need to consider the initial conditions for the system.

3.3. *Initial conditions*

In order to provide initial conditions for the moments' dynamics, we need to compute $E[x_0^k]$ and $E[x_1^k x_0^l]$ for generic $k > 0$ and $l \geq 0$.

If we are at the very first iteration of the PSO algorithm, the initial conditions for the dynamics of the moments are related to the ranges used to initialise positions and velocities in the PSO. Under the assumption that a particle's initial position, x_0, is chosen uniformly at random in a symmetric range $[-\Omega, \Omega]$, we have

$$E[x_0^k] = \begin{cases} 0 & \text{if } k \text{ is odd,} \\ \dfrac{\Omega^k}{k+1} & \text{otherwise.} \end{cases} \tag{22}$$

In order to compute $E[x_1^k x_0^l]$, we need to consider the equation

$$x_1 = x_0 + w v_0 - x_0(\phi_1 + \phi_2) + \phi_1 y + \phi_2 \widehat{y}, \tag{23}$$

where a particle's initial velocity, v_0, is a stochastic variable uniformly distributed the range $[-\Omega_v, \Omega_v]$ (often $\Omega_v = \Omega$). By taking the kth power of both sides of the equation, multiplying by x_0^l, and taking expectations, as we did to construct (17), one obtains the desired expressions for $E[x_1^k x_0^l]$. Like for (17), these expressions contain a number of terms that grows exponentially with n. However, this process, too, can be trivially mechanised.

If the PSO is not at its first iteration, and a new improvement has just been found to a particle's personal best or neighbourhood best, then x_0 is not a stochastic variable, but a constant determined by the position where the particle found the last improvement.

4. SKEWNESS AND KURTOSIS OF THE PSO's SAMPLING DISTRIBUTION

We constructed the recursions for moments of up to order 4 as described in the previous section for the canonical PSO. Naturally, for higher-order moments, we can do exactly the same type of eigenvalue analysis we did in [16] for the mean and standard deviation of the sampling distribution. For example, Figure 1 shows the magnitude of the largest eigenvalue, Λ_m, of M_4 as a function of the parameters w and c. The system is order-4 stable, that is, mean, variance, skewness, and kurtosis have a stable fixed point, whenever $\Lambda_m < 1$, that is, within the curved region enclosed by the contour shown in the figure.

For easier comparison, we show the lines where $\Lambda_m = 1$ for M_1, M_2, M_3, and M_4 in Figure 2 (ordered from top to

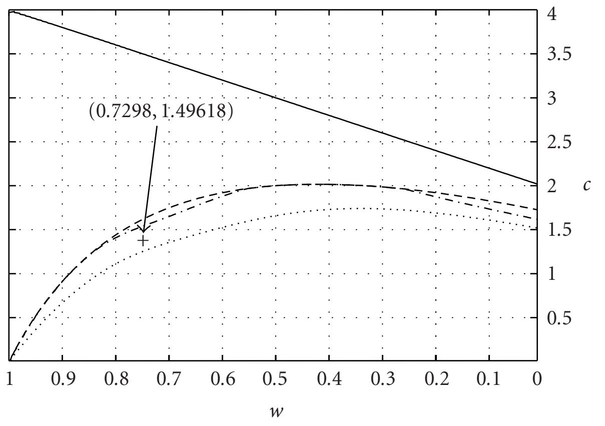
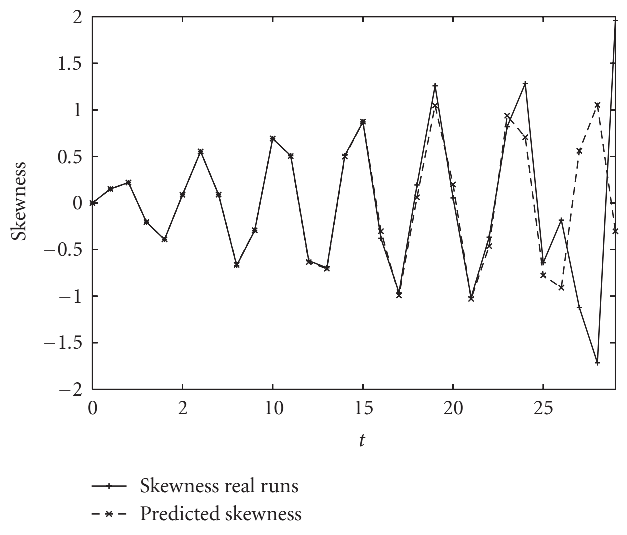


FIGURE 2: Plot of the regions of order-1, -2, -3, and -4 stability for the canonical PSO.

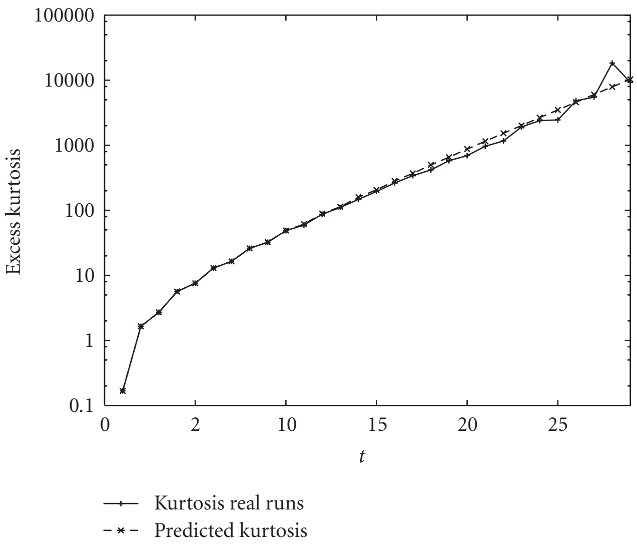
bottom, resp.). The regions of order-1, -2, -3, and -4 stability are nested. Note how the $\Lambda_m = 1$ lines for M_2 and M_3 coincide for many values of w. Note also that the standard setting, $c = 1.49618$ and $w = 0.7298$, lays within the narrow region of order-3 stability. This implies that while mean, variance, and skewness of the standard PSO tend to a fixed point, kurtosis is unstable and will tend to grow indefinitely. Interestingly, a growth in the kurtosis of samples was observed by Kennedy [17], although this was effectively computed under the assumption that the sampling distribution is time independent. So the values of x_t recorded in a run at t grows were treated as different samples from the same distribution, while we now know this is incorrect.

That the predictions of the model are exact is also confirmed by the comparison of the dynamics of predicted and recorded higher-order moments. Figure 3(a) shows a comparison between the skewness $E[(x_t - \mu_t)^3]/\sigma_t^3$ computed using our model and the average positions of the particle recorded in one billion (1 000 000 000) real runs in the first 30 iterations for the case $c = 1.49618$, $w = 0.7298$, $y = 0$, $\hat{y} = 1$, and $\Omega = 5$. As one can see, there is a very good match between the model's predictions and the stagnation behaviour of particles in real runs. Only after about 20–25 generations, the sampling errors start accumulating enough to show significant differences. We know that these differences are due to sampling errors because before performing 1 000 000 000 runs, we attempted to use much smaller sets of runs. With fewer runs, we observed that errors were visible well before generation 25. For example, with 1 000 000 runs, there is a good match up to around iteration 15.

As shown in Figure 3(b), the model also predicts very well the behaviour of the (excess) kurtosis $E[(x_t-\mu_t)^4]/\sigma_t^2-3$ of the sampling distribution. (Following standard practice, in this paper whenever we use the term "kurtosis," we will refer to the excess kurtosis $E[(x_t-\mu_t)^4]/\sigma_t^2-3$. The excess kurtosis of the normal distribution to 0.) Note that for $c = 1.49618$ and $w = 0.7298$, the system is order-3 stable, and so, although the oscillations of the skewness shown in Figure 3(a) appear to grow bigger and bigger, suggesting instability, this



(a)



(b)

FIGURE 3: Comparison between predicted and experimental skewness and (excess) kurtosis of the PSO sampling distribution for $c = 1.49618$, $w = 0.7298$, $y = 0$, $\hat{y} = 1$, and $\Omega = 5$. Kurtosis grows exponentially, and so it is plotted on a logarithmic scale. The first point is not plotted because the excess kurtosis was negative (-1.2).

is actually only a transient effect, as shown in Figure 4 where we integrate the equations over 200 generations instead of 30.

5. COMPARISON BETWEEN PSO's

In the previous sections, we studied the canonical PSO with the restriction that the acceleration coefficients, c_1 and c_2, were identical: $c_1 = c_2 = c$. One may wonder, however, whether allowing such coefficients to differ would produce qualitatively very different dynamics. For example, what would happen if we set one of the c_i's to zero as in a purely cognitive or purely social PSO model? This effectively would reduce to one the sources of random influences on a parti-

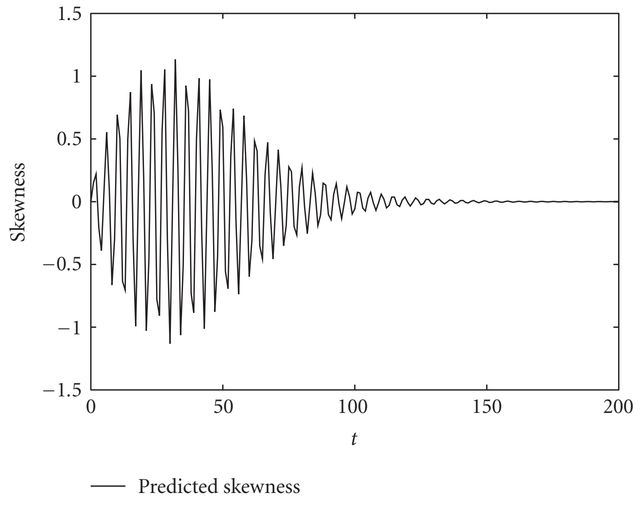


FIGURE 4: Predicted dynamics of the skewness of the PSO sampling distribution over 200 generations for $c = 1.49618$, $w = 0.7298$, $y = 0$, $\hat{y} = 1$, and $\Omega = 5$.

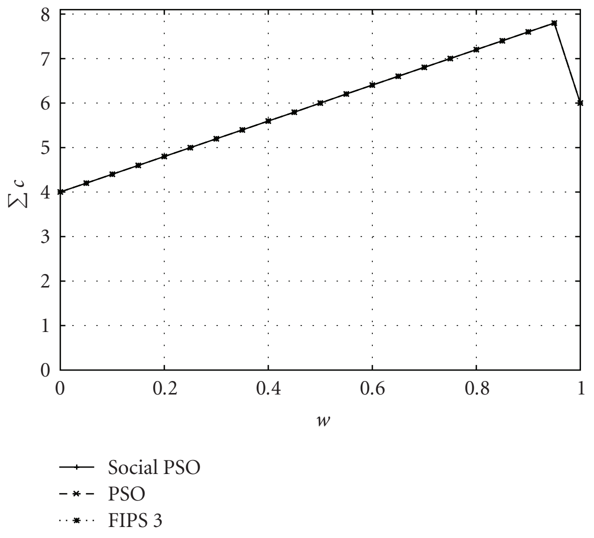


FIGURE 5: Lines below which the mean of the sampling distributions for a social PSO, a canonical PSO, and FIPS with a neighbourhood of three individuals have a fixed point (order-1 stability). The three lines coincide.

cle's dynamics. Conversely, one might wonder what would happen if we increased the number of such sources of influence, as is done, for example, in the fully informed particle swarm (FIPS) [18, 19].

5.1. Moments of generalised PSO's

To answer these (and other) important questions on the sampling distribution of different PSO models, we adopt a FIPS-like general class of PSO's described by the following difference equation:

$$x_{t+1} = x_t + wv_t + \sum_{i=1}^{m} \phi_i(\hat{y}_i - x_t), \tag{24}$$

where the ϕ_i's are stochastic variables uniformly distributed in the range $[0, c_i]$, c_i are constants, and the $\hat{y}_i$'s are the personal best positions of neighbours of the particle (the particle itself may be included in its own neighbourhood). Naturally, this equation can be converted into the following:

$$x_{t+1} = x_t(1 + w) - wx_{t-1} - \sum_{i=1}^{m} \phi_i x_t + \sum_{i} \phi_i \hat{y}_i, \tag{25}$$

which is a generalisation of (2). All of the steps we performed in Section 3 can be repeated for (25). These lead to recursions of the form in (17) with the only difference that the coefficients ω_i take the more general form

$$\omega_i = \left(\frac{a_{0_i} w^{a_{w_i}} c_1^{a_{c1_i}} \cdots c_m^{a_{cm_i}} \hat{y}_1^{a_{y1_i}} \cdots \hat{y}_m^{a_{ym_i}}}{\prod_{j=1}^{m} (1 + a_{cj_i})} \right), \tag{26}$$

where a_{0_i}, a_{w_i}, $a_{c1_i}, \ldots, a_{cm_i}$, $a_{y1_i}, \ldots, a_{ym_i}$ are appropriate constants.

Because (25) contains $3 + 2 \times m$ terms, the complexity of the expansion now grows as $O((3 + 2 \times m)^n)$, where n is the order of the moments we are interested in. So the larger m, the heavier the computation load required to compute M_n. Once the transition matrices M_n are computed, however, they are exactly of the same size for all PSO models within the class defined by (24). Initial conditions can be found following the approach described in Section 3. Calculations are expensive but can be mechanised. We did this for the examples described below.

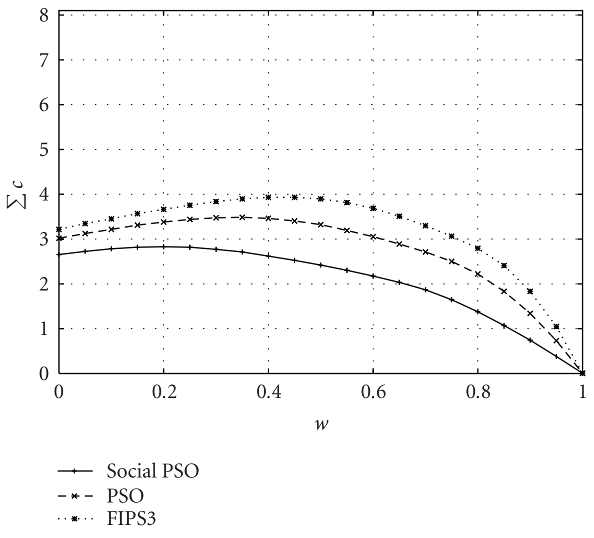


FIGURE 6: Lines below which the variance of the sampling distributions for a social PSO (bottom), a canonical PSO (middle), and FIPS with a neighbourhood of three individuals (top) have a fixed point (order-2 stability).

5.2. Three PSO's we want to compare

An extensive comparison of different PSO's is beyond the scope of this paper. However, as an example of the kind of

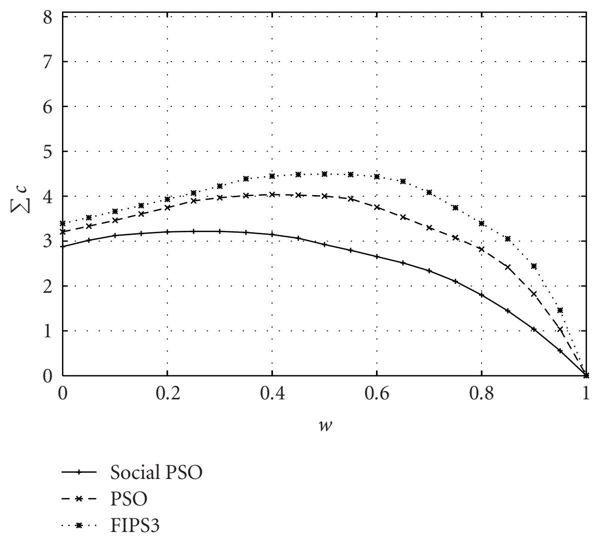


Figure 7: Lines below which a social PSO, a canonical PSO, and FIPS3 are order-3 stable (from bottom to top, resp.).

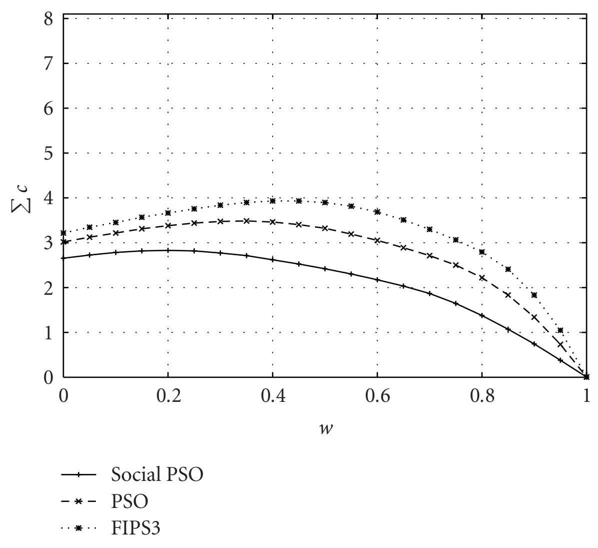


Figure 8: Lines below which the kurtosis of the sampling distributions for a social PSO (bottom), a canonical PSO (middle), and FIPS3 (top) have a fixed point (order-4 stability).

comparisons one can make using our approach, we considered the PSO's in (24) with $N = 3$. Within this class of PSO's, we considered three variants as follows:

(a) a purely social variant of PSO, which we will call *social PSO* for brevity, where $c_1 > 0$ and $c_2 = c_3 = 0$ (due to symmetries, the behaviour of a purely cognitive PSO where $c_2 > 0$ and $c_1 = c_3 = 0$ is effectively identical to that of this social PSO);

(b) the *canonical PSO* we have studied so far in the paper, which is obtained by setting $c_1 = c_2 > 0$ and $c_3 = 0$;

(c) the simplest version of FIPS with a neighbourhood of three individuals, for example, obtained using an *lbest* topology and an interaction radius of 1, where $c_1 = c_2 = c_3 > 0$. We will call this version *FIPS3*.

In order to perform a fair comparison of the stability properties of these PSO variants, we study them in conditions where the sum of the amplitudes of the random components, ϕ_i, is identical across models. That is, we set $c = \sum_i c_i$, and compare models with the same c value. Again, we analyse eigenvalues.

5.3. Results of the comparison

Figures 5–8 show the lines in the (w, c) plane where the magnitude of the largest eigenvalue of M_n , Λ_m is 1 for $n = 1, 2, 3, 4$ and for the three PSO variants mentioned above. Each figure represents a different moment. In the plot for the nth moment, the region below each line represents the set of all (w, c) pairs for which the nth moment of the distribution of the PSO corresponding to that line is stable. Let us analyse these figures in detail.

Firstly, we should note that the regions of order-1 stability for the three models are identical. This is because the dynamics of the mean of the three models is governed by equations of the same form, namely,

$$E[x_{t+1}] = E[x_t]\left(1 + w - \frac{c}{2}\right) - wE[x_{t-1}] + \text{constant}, \quad (27)$$

where the constant term may differ in different PSO variants. (This is irrelevant for the stability of the system, since stability is determined by the homogeneous part of the equation.) Note also that the rightmost point in each plot is an artifact due to the fact that, at $w = 1$, $\Lambda_m = 1$ for M_1 irrespective of the value of c.

The regions where the variance is stable for the three models, instead, are different, with FIPS3 having the largest region of order-2 stability, followed by the canonical PSO, and finally, by the social PSO. Exactly the same happens with skewness (Figure 7) and kurtosis (Figure 8), with the order-3 stability region largely coinciding with the order-2 ones also for FIPS3 and the social PSO.

These results are counter intuitive. One would expect that the more sources of randomness, the ϕ_i's, there are, the more a PSO should be unstable. However, the exact opposite happens. The social PSO, where the only influence is ϕ_1, is the least stable of all models, while FIPS3, which has three sources of randomness, is the most stable. What are the reasons for this behaviour?

We can understand this by rewriting (25) as follows:

$$x_{t+1} = x_t(1 + w) - wx_{t-1} - x_t\Phi_m + \Psi_m, \quad (28)$$

where $\Phi_m = \sum_{i=1}^{m}\phi_i$ and $\Psi_m = \sum_i \phi_i \hat{y}_i$. Both Φ_m and Ψ_m are the sum of independent and uniformly distributed variables: the variables ϕ_i in the case of Φ_m and the variables $\hat{y}_i\phi_i$ in the case of Ψ_m.

We know that $\sum_i c_i = c$. To simplify our treatment, let us further assume that the ϕ_i's are i.i.d., that is, that all c_i are identical, and so $c_i = c/m$. We can then apply the central limit

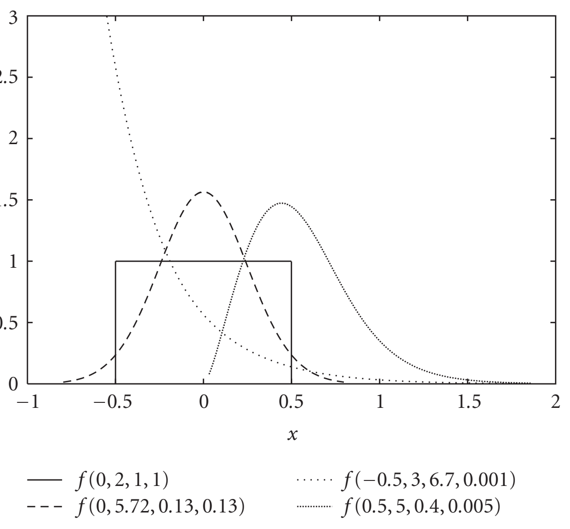


FIGURE 9: Examples of GLD density functions.

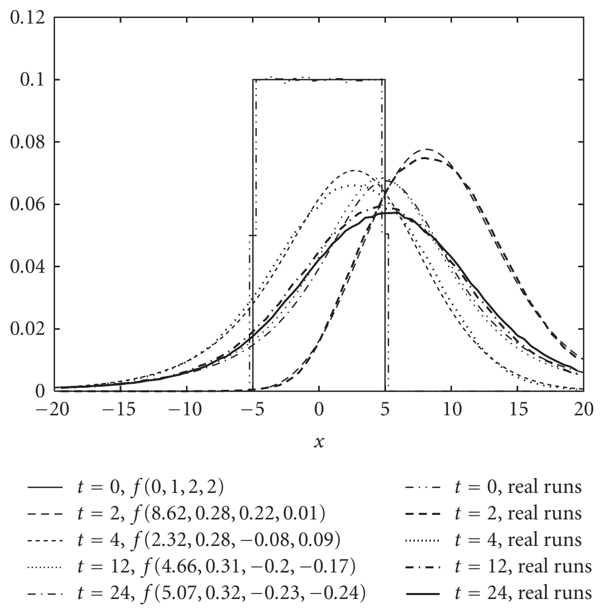


FIGURE 10: Estimates of the sampling distribution of a canonical PSO with parameters $c = 1.49618$, $w = 0.7298$, $y = 0$, $\hat{y} = 10$, and $\Omega = 5$ during stagnation, reconstructed via GLD best fitting vers. histograms over 1 000 000 real runs. Snapshots at times $t = 0$, 2, 4, 12, and 24 are shown. For each theoretical sampling distribution, we report the parameters of the corresponding GLD.

theorem to Φ_m. This predicts that for sufficiently large m, the distribution of Φ_m is approximately Gaussian with mean $\sum_i c_i/2 = c/2$ and variance $\sum_i(c_i^2/3 - (c_i/2)^2) = \sum_i c_i^2/12 = c^2/(12m)$. So the larger m, the smaller the variance of Φ_m and the stochasticity of (28).

In the case of the stochastic variable Ψ_m, the quantities $\phi_i\hat{y}_i$ are not identically distributed even if all c_i are identical. This is because, in principle, each $\hat{y}_i$ may be different. This prevents the use of the standard central limit theorem. We can, however, apply Lyapunov's central limit theorem to Ψ_m. The conditions for its application are as follows:

(1) the variables $\phi_i\hat{y}_i$ must have finite mean, which is the case since $\mu_i = E[\phi_i\hat{y}_i] = c_i\hat{y}_i/2 = c\hat{y}_i/(2m)$,

(2) the variables $\phi_i\hat{y}_i$ must have finite variance, which, again, is the case since $\sigma_i^2 = E[(\phi_i\hat{y}_i - \mu_i))^2] = (c_i\hat{y}_i)^2/12 = (c\hat{y}_i/m)^2/12$,

(3) the variables $\phi_i\hat{y}_i$ must have finite third central moment, which is satisfied since $r_i^3 = E[(\phi_i\hat{y}_i - \mu_i))^3] = 0$, and finally,

(4) the Lyapunov condition, $\lim_{m\to\infty}((\sum_{i=1}^m r_i^3)^{1/3}/(\sum_{i=1}^m \sigma_i^2)^{1/2}) = 0$, must be satisfied, which, again, is the case since all $r_i^3 = 0$.

Then for sufficiently large m, also the distribution of Ψ_m is approximately Gaussian with mean $\sum_i c\hat{y}_i/(2m) = (c/2) \times (\sum_i \hat{y}_i/m)$ and variance $(c^2/12m) \times (\sum_i \hat{y}_i^2/m)$. Note that $\sum_i \hat{y}_i/m$ and $\sum_i \hat{y}_i^2/m$ are the mean $\hat{y}_i$ and the mean $\hat{y}_i^2$, respectively. So these are finite quantities if, as is normally the case, all $\hat{y}_i$ are finite. (PSO search is normally confined to a prefixed, finite region of $\mathbb{R}^N$, and so all $\hat{y}_i$ must be finite.) So like for Φ_m, the larger m, the smaller the variance of Ψ_m, and consequently, the less the stochasticity of (28).

Effectively, the larger m, the more Φ_m and Ψ_m become deterministic and approach constant values. This explains why adding more and more sources of randomness—while keeping c constant—produces progressively more and more stable PSO's.

6. THE DENSITY FUNCTION OF THE CANONICAL PSO's SAMPLING DISTRIBUTION

The technique described in this paper, in principle, would allow one to determine all the moments of the sampling distribution of the PSO at all times. The question then is "could we derive the PSO sampling distribution itself ?" The answer is of course in the positive since knowing all the moments of a distribution implies knowing its moment generating function. This, in turn, allows one to obtain the density function of the distribution via inverse Laplace transform.

In practice, however, it is impossible to compute all the moments of the PSO sampling distribution. This is for two reasons. Firstly, there are infinitely many such moments. Secondly, as we have seen in the previous sections, the cost of computing moments is exponential in the order of the moments. The next question is then "to what extent can we still reconstruct the PSO's density function from a finite number of moments?" This is an instance of the well-known *truncated moment problem*, a difficult, inverse ill-posed problem for which many approaches have been proposed. Here, we consider only one such approach.

A particularly simple idea is to consider a family of density functions $f(x;\lambda_1,\lambda_2,\ldots)$ with parameters λ_1, λ_2, and so forth, with sufficient expressive power to represent distributions with widely different shapes, with more or less asymmetries, with tails of different characteristics, and so on. Then one can use optimisation techniques to identify the parameters of the distribution f which minimise the difference between the moments of f and the moments of the PSO's sampling distribution. This is called the *moment-matching method*. Once the parameters λ_1, λ_2, and so forth, are identified, f can be used as an approximation of the true PSO sampling distribution. This approach to reconstructing probability distributions from moments was proposed [20] (see also [21, 22]), where a *generalised lambda distribution* (GLD) was used. We adopt this same approach here.

GLD is a four-parameter distribution defined via its quantile function as follows:

$$Q(u) = \lambda_1 + \frac{1}{\lambda_2}\left(\frac{u^{\lambda_3} - 1}{\lambda_3} + \frac{1-(1-u)^{\lambda_4}}{\lambda_4}\right), \tag{29}$$

where $u \in [0, 1]$. Its density function is given by

$$f(x;\lambda_1,\lambda_2,\lambda_3,\lambda_4) = \left(\frac{dQ(u)}{du}\right)^{-1} = \frac{\lambda_2}{u^{(\lambda_3-1)} + (1-u)^{(\lambda_4-1)}}, \tag{30}$$

where $u = Q^{-1}(x)$.

The GLD is enormously flexible in terms of the shape of the distribution. For example, as shown in Figure 9, the uniform, Gaussian, exponential, and Gamma distributions are all special cases of GLD. Effectively, λ_1 determines the location of the distribution, λ_2 determines its scale, while λ_3 and λ_4 determine other shape characteristics. In particular, only if $\lambda_3 = \lambda_4$, the distribution is symmetric.

Because GLD has 4 parameters, all we need is four moments—the mean, variance, skewness, and kurtosis—of the PSO's sampling distribution in order to identify such parameters with the moment-matching method described above.

As an illustration, we apply this technique to reconstruct the sampling distribution during stagnation of a canonical PSO with parameters $c = 1.49618$, $w = 0.7298$, $y = 0$, $\hat{y} = 10$, and $\Omega = 5$. With these parameters, $\lim_{t\to\infty} E[x_t] = 5$. In Figure 10, we show snapshots at times $t = 0, 2, 4, 12$, and 24 of the theoretical sampling distribution together with estimates of the distribution based on 1 000 000 actual runs. In all cases, the match between the moments of the GLD and those of the PSO sampling distribution was exact (within experimental errors). Also, there is considerable agreement between the theoretical lines and histograms obtained in real runs. Note how widely the mean of the density function oscillates around 5 in the first few generations. Also note the asymmetry in the distributions due to the oscillations of the skewness.

7. DISCUSSION AND CONCLUSIONS

Several theoretical analyses of the dynamics of particle swarms have been offered in the literature over the last decade. These have been very illuminating. However, virtually, all have relied on substantial simplifications and on the assumption that the particles are deterministic. Naturally, these simplifications make it impossible to derive an exact characterisation of the sampling distribution of the PSO.

In previous work [16], by using surprisingly simple techniques, we started by exactly determining, perhaps, the most important characteristic of a PSO's sampling distribution, its variance, and we have been able to explain how it changes over any number of generations. The only assumption we made is stagnation. Here, we extended this technique to the study of higher-order statistics. In particular, we analysed, in detail, the skewness and kurtosis of the distribution. Because of the complexity of the calculations involved, this required mechanising the derivation of the recursions for these moments.

We applied the analysis to the PSO with inertia weight, but the analysis is also valid for the PSO with constriction, because of the well-known equivalence of these two models (via a simple parameter mapping). We also generalised our model so as to include FIPS. This made it possible to explicitly compare the stability of different forms of PSO, leading to a deeper understanding of their properties. In particular, we showed that while FIPS and standard forms of PSO present exactly the same order-1 stability, in FIPS, higher-order moments are more stable than in the other PSO's, and we were able to explain why this is the case, using two forms of central limit theorem (Section 5). Elsewhere [23], we also applied the same type of analysis to derive the stability regions of a simpler form of velocity-based PSO.

Finally, with all these tools in hand, we went in search for the "holy grail"—the actual PSO sampling density function. We treated the problem as an ill-posed inverse problem, which we regularised thanks to the use (and best fit) of a family of distributions—the generalised lambda distribution (GLD). All empirical evidence we have suggests that this distribution approximates very closely the sampling behaviour of PSO's (naturally, with parameters that are functions of time, i.e., $\lambda_i = \lambda_i(t)$). We will attempt to characterise the PSO's sampling distribution in more depth in future research.

Whether or not GLD is the exact PSO sampling distribution or just a very good approximation, if one could determine (again, either exactly or approximately) how the $\lambda_i(t)$'s are affected by the parameters c, w, y, and $\hat{y}$ and by the initial conditions x_0 and v_0, it would then be possible to accurately simulate the behaviour of the PSO by sampling from $f(x;\lambda_1(t),\lambda_2(t),\lambda_3(t),\lambda_4(t))$. This is easily done since GLD deviates can trivially be produced by picking u uniformly at random in the interval $[0, 1]$ and applying (29), that is, $Q(U[0, 1])$ is generalised lambda distributed. We will study this more sophisticated form of bare-bones PSO [17] in future research.

Bare-bone PSO's have some resemblance with estimation of distribution algorithms (EDA's). EDA's are powerful population-based searchers where the variation operations traditionally implemented via crossover and mutation in evolutionary algorithms are replaced by the process of sampling from a distribution (a review of EDA's is available in [24]). The distribution is modified generation after generation, by using information obtained from the more fit individuals in the population. The objective of these changes is to increase the probability of generating individuals with high fitness. As one can clearly see from this description, a bare-bone PSO performs a very similar form of search to an EDA. However, there is a difference between the two. While in a typical EDA the whole new population is generated from a single distribution, in a bare-bone PSO, each individual has a personal sampling distribution from which only one sample is drawn at each iteration. (The sampling distributions of different particles may share some characteristics, since the sampling distribution of a particle is determined by both personal best and neighbourhood best, and the neighbourhood best can easily be the same for different particles.) Nonetheless, one could still interpret a bare-bone PSO as an unusual type of EDE in which multiple distributions are used and updated based on fitness during the search.

ACKNOWLEDGMENTS

The author would like to acknowledge the support by EPSRC (Grant no. GR/T11234/01, "Extended Particle Swarms (XPS)") and the help and comments from Maurice Clerc, David Broomhead, Susanna Wau Men Au-Yeung, and Roberto Renó. The anonymous reviewers and the editors are also thanked for their useful feedback.

REFERENCES

[1] J. Kennedy, "The behavior of particles," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., pp. 581–589, Springer, San Diego, Calif, USA, March 1998.

[2] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[3] E. Ozcan and C. K. Mohan, "Analysis of a simple particle swarm optimization system," in *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 8, pp. 253–258, ASME Press, St. Louis, Mo, USA, 1998.

[4] E. Ozcan and C. K. Mohan, "Particle swarm optimization: surfing the waves," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1939–1944, IEEE Service Center, Washington, DC, USA, July 1999.

[5] F. van den Bergh, *An analysis of particle swarm optimizers*, Ph.D. thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[6] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, New York, NY, USA, 2005.

[7] K. Yasuda, A. Ide, and N. Iwasaki, "Adaptive particle swarm optimization," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1554–1559, Washington, DC, USA, October 2003.

[8] T. M. Blackwell, "Particle swarms and population diversity," *Soft Computing*, vol. 9, no. 11, pp. 793–802, 2005.

[9] B. Brandstatter and U. Baumgartner, "Particle swarm optimization—mass-spring system analogon," *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 997–1000, 2002.

[10] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[11] E. F. Campana, G. Fasano, and A. Pinto, "Dynamic system analysis and initial particles position in particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '06)*, Indianapolis, Ind, USA, May 2006.

[12] E. F. Campana, G. Fasano, D. Peri, and A. Pinto, "Particle swarm optimization: efficient globally convergent modifications," in *Proceedings of the 3rd European Conference on Computational Mechanics, Solids, Structures and Coupled Problems in Engineering (ECCM '06)*, C. A. Mota Soares, J. A. C. Martins, H. C. Rodrigues, and J. A. C. Ambrosio, Eds., Lisbon, Portugal, June 2006.

[13] M. Clerc, "Stagnation analysis in particle swarm optimisation or what happens when nothing happens," Tech. Rep. CSM-460, Department of Computer Science, University of Essex, Colchester, UK, August 2006.

[14] V. Kadirkamanathan, K. Selvarajah, and P. J. Fleming, "Stability analysis of the particle dynamics in particle swarm optimizer," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 245–255, 2006.

[15] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[16] R. Poli and D. Broomhead, "Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation," in *Proceedings of the 9th Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 134–141, ACM Press, London, UK, July 2007.

[17] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 80–87, Indianapolis, Ind, USA, April 2003.

[18] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1671–1676, Honolulu, Hawaii, USA, May 2002.

[19] R. Mendes, *Population topologies and their influence in particle swarm performance*, Ph.D. thesis, Departamento de Informatica, Escola de Engenharia, Universidade do Minho, Braga, Portugal, 2004.

[20] J. S. Ramberg, P. R. Tadikamalla, E. J. Dudewicz, and E. F. Mykytka, "A probability distribution and its uses in fitting data," *Technometrics*, vol. 21, no. 2, pp. 201–214, 1979.

[21] A. Lakhany and H. Mausser, "Estimating the parameters of the generalized lambda distribution," *Algo Research Quarterly*, vol. 3, no. 3, pp. 47–58, 2000.

[22] S. W. M. AuYeung, "Finding probability distributions from moments," M.S. thesis, Imperial College, London, UK, 2003.

[23] R. Poli, D. Bratton, T. M. Blackwell, and J. Kennedy, "Theoretical derivation, analysis and empirical evaluation of a simpler particle swarm optimiser," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, IEEE Press, Singapore, September 2007.

[24] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Boston, Mass, USA, 2002.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 893237, 10 pages
doi:10.1155/2008/893237

Research Article

Examination of Particle Tails

Tim Blackwell and Dan Bratton

Department of Computing Goldsmiths, University of London, New Cross, London SE14 6NW, UK

Correspondence should be addressed to Tim Blackwell, tim.blackwell@gold.ac.uk

Received 20 July 2007; Accepted 29 November 2007

Recommended by Riccardo Poli

The tail of the particle swarm optimisation (PSO) position distribution at stagnation is shown to be describable by a power law. This tail fattening is attributed to particle bursting on all length scales. The origin of the power law is concluded to lie in multiplicative randomness, previously encountered in the study of first-order stochastic difference equations, and generalised here to second-order equations. It is argued that recombinant PSO, a competitive PSO variant without multiplicative randomness, does not experience tail fattening at stagnation.

1. INTRODUCTION

This paper focuses on the effective sampling distribution of particle swarm optimisation (PSO). PSO is a population-based global optimisation technique. The algorithm is made of two essential steps: particle movement (dynamics) through a search space of solutions and an indirect particle interaction which is mediated by information sharing within a social network. Particles place personal "attractors" in the search space at positions corresponding to the best location (as determined by the evaluation of an objective function) that they have visited. The dynamical update rule which specifies the acceleration of a particle i is a sum of an attraction towards a personal attractor and an attraction towards the best attractor amongst other particles in i's social neighbourhood [1–3]. The particle acceleration is added to particle velocity in a discretisation of particle kinematics; the velocity is added in turn to particle position and a new trial solution is achieved. Despite the simplicity of this scheme, the algorithm is effective over the standard benchmark problems and has increasing numbers of real-world applications. However, little is known about how a PSO achieves its results.

The sampling distribution provides the probability density of particle placement between and around fixed attractors. This scenario occurs when the swarm "stagnates," the swarm is not improving, the network of attractors is fixed, and the particles are moving independently of each other, that is, the particles decouple. Stagnation can occur in the full model, or can be imposed for the purposes of the oretical analysis. In general, particles are expected to search in the immediate vicinity of the attractors, and indeed this behaviour is necessary for convergence. The central portion of the sampling distribution is therefore coincident with this region. However, the swarm must retain an ability to explore outside the attractor vicinity if premature convergence, a problem in complex environments, is to be avoided. This exploration is due to the finite tail of the sampling distribution; there is a small probability that a particle will be displaced far from the region of convergence, and this probability increases with tail fatness.

Some insights into the PSO sampling distribution have been provided by a study of "bare bones" formulations [4]. In this work, Kennedy replaced the particle update rule with sampling from a Gaussian distribution with mean at the centroid of the personal and neighbourhood best positions of each particle, and standard deviation was set to the separation between them. Performance comparisons with a few benchmark problems were disappointing (later confirmed in [5]), but the addition by hand of particle "bursts" ameliorated the situation somewhat, indicating that the tails of this Gaussian distribution function are too thin to enable escape from stagnation. Further evidence for this conclusion has been provided by a study of Lévy bare bones [5] where more extensive trials showed that fat tails, as provided by the Lévy distribution, improve bare bones performance so that it becomes effectively equivalent to standard PSO. A recent

```
0. initialise swarm
FOR EACH particle $i$
    randomise $\vec{v}_i$, $\vec{x}_i$, set $\vec{p}_i = \vec{x}_i$
FOR EACH particle $i$
    1. find neighbourhood best
        $\vec{g}_i = \arg\min(f(p_j),\ j \in N_i)$
    FOR EACH dimension $d$
        2. move particle
            $F(p) : x_{id} \leftarrow x_{id}$
        3. update memory
            IF $f(\vec{x}_i) < f(\vec{p}_i)$ THEN
                $\vec{p}_i \leftarrow \vec{x}_i$
    END
END
```

Algorithm 1: Particle swarm opimisation.

theoretical analysis of bare bones PSO has been given in [6]. However, a Gaussian bare bones version of a "fully informed" particle swarm (FIPS) was able to deliver good performance over a small testbed of functions [7], which suggests that particle tails might not be so important, at least for FIPS.

It is known that decoupled PSO exhibits bursts of outliers [8]. Bursts are temporary excursions, along a coordinate axis, of the particle to large distances from the attractors. A burst will typically grow to a maximum and then return through a number of damped oscillations to the region of the attractors. The addition of bursts to Gaussian bare bones increases the chance of particle displacement away from the attractors, fattening the tail of the overall sampling distribution. Lévy distributions also produce particle outliers, but these outliers are not correlated; they do not occur in sequence along a single axis. At the moment, the circumstances, when a sequence of correlated outliers (as manifest, for example, in PSO bursts) might prove to be fortuitous, are unknown. This paper investigates the tail of the PSO sampling distribution. Our results show that this tail is described by power laws and is therefore fatter than Gaussian. This fattening is indeed due to bursting, the origin of which is concluded to lie in a process known as multiplicative randomness.

Bursting is already known to occur in first-order stochastic difference equations [9]. This paper generalises the first order results to second order difference equations with multiplicative randomness, a class of equations that includes standard PSO. Since the amount of bursting is dependent on the range of the probability distributions, the paper begins by discussing possible parameter regimes. This is accomplished by a formulation of PSO as a finite difference equation. The paper continues with a series of experiments which reveal the power law tails of the sampling distribution. Section 4 introduces multiplicative randomness and power law tails in first-order processes and extends these results to second-order difference equations. A competitive reformulation PSO without multiplicative randomness has recently been proposed (recombinant PSO, [10]). Section 5 investigates decoupled recombinant PSO and demonstrates the consequent thinness of the distribution tail. The results of this paper are drawn together in a concluding section and some open research questions are outlined.

2. PSO AND STOCHASTIC DIFFERENCE EQUATIONS

This section recasts PSO as a stochastic difference equation (SDE). Since this is an unfamiliar form of PSO, a few relations from the literature governing parameter choice are gathered together and presented here.

Each PSO particle i in the swarm has dynamic variables position and velocity, $\vec{x}_i$ and $\vec{v}_i$, and a memory $\vec{p}_i$ of a past position visited. Furthermore, each particle is embedded in a social, rather than spatial, neighbourhood N_i of informers. The algorithm is outlined in Algorithm 1. A general dynamic rule $F(p)$ for a single particle position update is

$$\begin{aligned} v_{id}(t+1) &= wv_{id}(t) + \sum_{j=1}^{K} \Phi_j(t)(p_{jd} - x_{id}(t)), \\ x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1). \end{aligned} \tag{1}$$

The sum in (1) is over K informers p_i and for each dimension d. In standard PSO [11], however, $K = 2$ and the two attractors p_i and p_{g_i} are the personal and neighbourbood best positions. In this case,

$$\Phi_{1,2} = \phi_{1,2} u_{1,2}, \tag{2}$$

where $\phi_{1,2}$ are "acceleration" constants and $u_{1,2} \sim U(0,1)$ are random numbers drawn from the uniform distribution on the unit interval. This "inertia weight" (IW) formulation of PSO [12] derives its name from the parameter w which imitates inertia in the sense that it weights the tendency to move in a straight line at constant speed ($w = 1$) to the tendency to move erratically about the attractors Φ. Other formulations of PSO include Kennedy and Mendes' [13] fully informed particle swarm (FIPS), wherein a particle is influenced by $K > 2$ neighbours, $\Phi_j = (1/K)\phi_j u_j$, and the "constricted" Clerc-Kennedy (CK) swarm [14] which is equivalent to (1) with the identification $\chi = w$, $\phi_{1,2} = \chi\phi_{1,2}^{\text{CK}}$.

At stagnation, we only need to consider a single particle in one dimension, so particle labels i will be dropped. By virtue of $v(t) = x(t) - x(t-1)$, (1) can be rewritten as a second-order stochastic difference equation (SDE)

$$x(t+1) + a(t)x(t) + bx(t-1) = c(t) \tag{3}$$

with

$$\begin{aligned} a(t) &= \sum_j \Phi_j(t) - w - 1, \\ b &= w, \\ c(t) &= \sum_j \Phi_j(t) p_j, \end{aligned} \tag{4}$$

where the parameters a and c are stochastic variables because of the presence of random numbers in (2). (However, they are not independent because the same random numbers u_1 and u_2 appear in a and c.)

Constant parameter SDEs have been studied by many authors in a number of domains (e.g., [15] gives references to population dynamics, epidemics, ARCH(1) processes, investment portfolios, immigrant populations, and internet usage). A constant parameter second-order difference equation, $a(t) = a$, $c(t) = a$, $\Phi(t) = \Phi$, is obtained by replacing the random variables $u_{1,2}$ by a constant u. Stability conditions can then be found by substituting the trial solution $x = \lambda^t$ into (3) and considering roots of the resulting characteristic equation $\lambda^2 + a\lambda + b = c$ (see, e.g., [16]). Stability then requires $|\lambda| < 1$. Complex, and therefore oscillatory, solutions are found if

$$\begin{gathered} 0 < b < 1, \\ a^2 < 4b, \end{gathered} \tag{5}$$

and stable real soutions exist for

$$\begin{gathered} |a| < 1 + b, \\ ca^2 \geq 4b. \end{gathered} \tag{6}$$

The stability conditions can be combined:

$$\begin{gathered} |b| < 1, \\ |a| < 1 + b. \end{gathered} \tag{7}$$

In order to relate these stability conditions to the stochastic model of (3) and (4), a number of authors have suggested replacing the random variable $u_{1,2}$ by its maximum, that is, $u = 1$ [14, 17, 18]. In terms of PSO parameters w and Φ (assumed constant), this gives

$$\begin{gathered} |w| < 1, \\ 0 < \sum_j \Phi_j < 2(1 + w). \end{gathered} \tag{8}$$

Poli and Broomhead [19] has extended this analysis by retaining randomness and deriving stability relations for the expectations of the first and second moments of position. The relation for the expected value of x is equivalent to the above analysis with $u = 0.5$, so the upper bound on the acceleration parameters ϕ is therefore twice the $u = 1$ bound.

Standard PSO is implemented with equal acceleration parameters so that $\phi_1 = \phi_2$. Defining $\phi_1 + \phi_2 = \phi$, the dynamics for the inertia weight (IW) or the Clerc-Kennedy (CK) formulations is

$$\text{IW}: v(t+1) = wv(t) + \frac{\phi^{\text{IW}}}{2}\left[u_1\left(p_i - x(t)\right) + u_2\left(p_2 - x(t)\right)\right], \tag{9}$$

$$\text{CK}: v(t+1) = \chi v(t) + \chi\frac{\phi^{\text{CK}}}{2}\left[u_1\left(p_i - x(t)\right) + u_2\left(p_2 - x(t)\right)\right]. \tag{10}$$

Equation (8) gives, at $u = 1$,

$$0 < \phi^{\text{IW}} < 2(1 + w), \tag{11}$$

$$0 < \phi^{\text{CK}} < \frac{2(1+\chi)}{\chi}. \tag{12}$$

The CK condition for complex eigenvalues and oscillation, $a^2 < 4b$, becomes $a = \chi\phi$, $b = \chi$,

$$\begin{gathered} |\chi| < 1, \\ 1 + \frac{1}{\chi} - \frac{2}{\sqrt{\chi}} < \phi^{\text{CK}} < 1 + \frac{1}{\chi} - \frac{2}{\sqrt{\chi}}. \end{gathered} \tag{13}$$

In order to simplify the choice of χ and ϕ^{CK}, Clerc and Kennedy suggest a single relation $\phi^{\text{CK}} = \phi^{\text{CK}}(\chi)$,

$$\phi^{\text{CK}} = \frac{1}{\chi} + \chi + 2 \tag{14}$$

which can be simply rewritten as

$$\chi\phi^{\text{CK}} = (\chi + 1)^2 \tag{15}$$

and can be easily seen to satisfy (13). This relation is usually inverted in the literature,

$$\chi^{\text{CK}} = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}, \quad \phi > 4, \tag{16}$$

and a common choice is $\phi^{\text{CK}} = 4.1$, $\chi \approx 0.73$.

As $\phi^{\text{CK}} \to 4$, $\chi \to 1$, the system becomes unstable, and as ϕ^{CK} grows from 4, χ decreases from 1 and the system is increasingly damped. In terms of the inertia weight formulation, these parameters correspond to $w \approx 0.73$ and $\phi_{\text{IW}} \approx 3.0$. Many trials of PSO over commonly used test functions have found that best performance is attained at ϕ close to the $u = 1$ stability condition. The reason behind this can be elucidated by considering the statistical distribution of $x(t)$.

3. DISTRIBUTION TAIL

This section presents an experimental investigation of the tail of sampling distribution for decoupled PSO. In each experiment described here, $x(1)$ and $x(2)$ are random starting positions between the two fixed attractors, $p_1 = -0.5$ and $p_2 = 0.5$. Henceforth, only the IW form of PSO will be considered (the Clerc-Kennedy form is a simple parameter redefinition) and the suffix IW will be dropped for ease of notation.

Figure 1 shows the development of a spectacular burst for the system defined by (9) at $w = 0.75$, $\phi = 3.0$. The particle is close to the $u = 1$ instability condition since, from (11), $\phi_{\max} = 3.5$. Figure 1 shows a burst of two orders of magnitude, as measured in units of the intrinsic scale $|p_1 - p_2|$.

Figure 2 shows the frequency N of particle distance $r = |x|$ for the same system as Figure 1 for a run of 10^6 iterations. A logarithmic scale (all logs in this paper are to base 10) has been used for the y-axis so that the infrequent but large bursts are visible on the plot. For this single run, the mean distance was 0.747 (standard deviation 1.05) and all distances are in the interval $[1.01 \times 10^{-6}, 105]$. Many updates are therefore over very small distances from the origin, which is the fixed point $\langle c\rangle/(1 + \langle a\rangle + b)$ of the expectation value of x. Although the standard deviation is of the order of the attractor separation, r can range over 8 orders of magnitude.

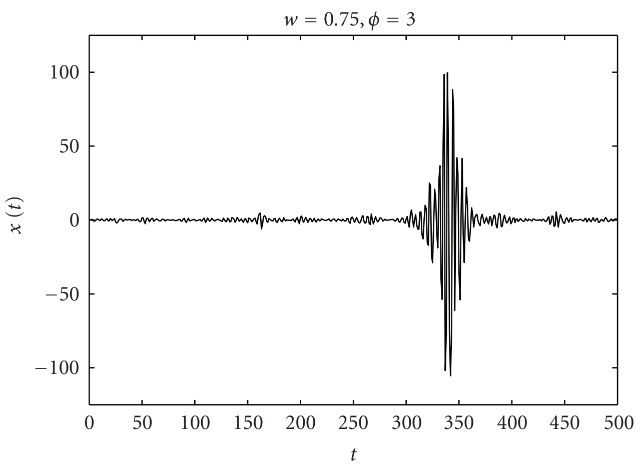


Figure 1: A burst of outliers in decoupled PSO.

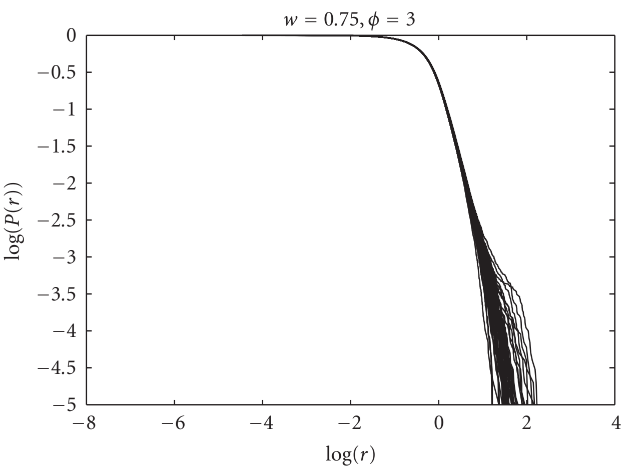


Figure 3: Cumulative probability distribution P versus particle distance r from the origin. The plot shows 50 runs.

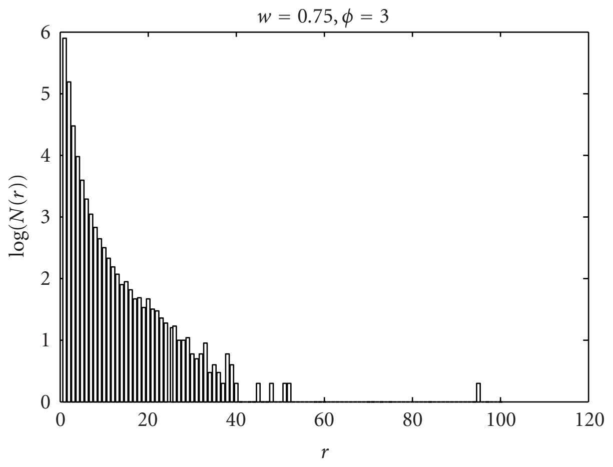


Figure 2: Frequency N of particle distances r from the origin for 10^6 iterations of decoupled PSO.

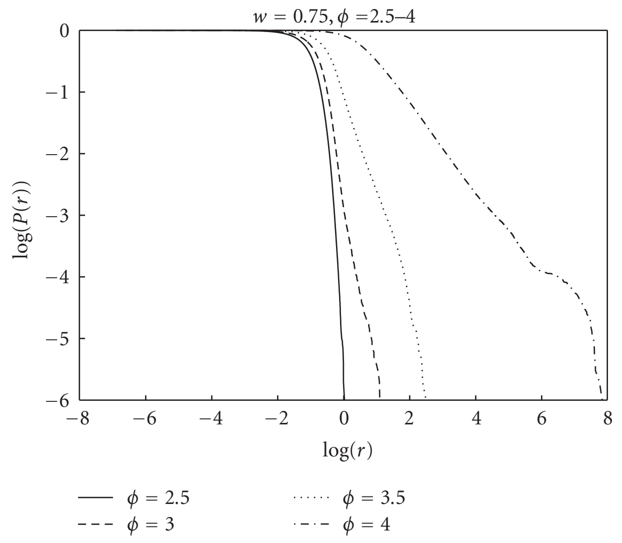


Figure 4: Cumulative probability distribution of particle distances r for various values of acceleration parameter ϕ.

Bursts would be expected to fatten the tail of the particle distance distribution $p(r)$ when compared to distributions with exponential falloffs such as a Gaussian. Evidence for possible power law fattening of the distribution tail, $p(r) \sim r^{-\alpha}$, where $p(r)dr$ is the probability of a particle at distance r, would be revealed in a plot of the logarithm of the cumulative distribution function, $P(r)$, where $P(R) = \text{prob}\,(r > R)$. A cumulative plot (also known as a rank/frequency plot [20]) reduces sampling errors in the tail of the plot, even with logarithmic binning [21]. A relation $p(r) \sim r^{-\alpha}$ corresponds to $P(R) \sim r^{1-\alpha}$ and a plot of $\log(P)$ against $\log(r)$ would show a straight line with gradient $1-\alpha$.

Figure 3 shows cumulative distributions for 50 runs of 10^5 iterations, once more for the decoupled PSO defined by $w = 0.75$, $\phi = 3.0$, $p_{1,2} = \pm 0.5$, $x(1)$, $x(2) = U(p_1, p_2)$. All runs have been plotted on this figure to give an idea of the deviations between runs. The straight portion in Figure 3 is evidence for a power law (although the underlying distribution might be log-normal; see discussion in the following paragraphs).

Figure 4 shows cumulative probability distribution plots for four values of ϕ at $w = 0.75$. This data is collected over 50 runs of 10^6 iterations for each value of ϕ. Each line shows a straight central part. The lines curve inwards towards the end of the sample where probabilities are small ($< 10^{-5}$) and there are just a few events. Once more, a large part of the distribution is concentrated in the region between the attractors, $r < 1$. The power laws become established by $r \approx 1.0$, the separation of the attractors. At $\phi = 4.0$ the power law is evident for some 4 orders of magnitude.

Putative power laws as revealed by log-log plots are hardly distinguished from log-normal laws $p(x) \sim \exp(-\ln(x-\mu)^2)$ over four or less orders of magnitude [21]. These plots therefore only show that the tails *might* be

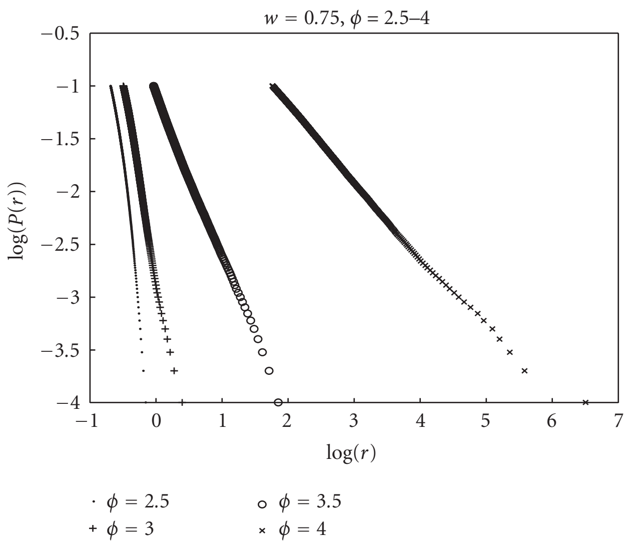


FIGURE 5: Power-law tails at various values of acceleration parameter ϕ.

modelled by a power-law distribution. The underlying distribution could be power law or another distribution, such as the log normal, whose tail can be approximated by a power law over some range.

Figure 5 plots the same data as Figure 4 but over the interval $10^{-4} < P(r) < 10^{-1}$ where the power laws are becoming established. For clarity, every 1000th r in this range has been plotted. The gradients and correlation coefficients of the four lines are $-3.94(-0.987)$, $-3.74(-0.998)$, $-1.08(-0.999)$, and $-0.73(-0.999)$ for $\phi = 2.5, 3.0, 3.5, 4.0$, respectively. (A correlation coefficient of -1.0 indicates perfect negative correlation.)

If the sampling distribution $p(r)$ does follow a power law falloff, $p(r) \sim r^{-\alpha}$ for $r > r_{\min}$ (as these results suggest), then $\langle r \rangle$ is well defined only for $\alpha > 2$ because $\int_{r_{\min}}^{\infty} r p(r) dr$ diverges for $\alpha < 2$. According to these experiments, at $\phi = 3.5$, the tail falls off as $p \sim r^{-2.08}$, so $\langle r \rangle$ is finite at this value (and at smaller values) of ϕ, but $\langle r \rangle$ diverges at larger values. This edge is just at the $u = 1$ stability condition. Lower values of u, and hence higher values of ϕ, lead to systems whose empirical mean over a finite number of iterations will be finite but will nevertheless vary enormously, sometimes taking on large values, in order to respect the formal divergence of the mathematical mean.

From the condition $P(r) = 0.1$, Figure 5 shows that 10 per cent of the particle positions are at distances greater than 0.21, 0.32, 0.92, and 56.8 from the origin for $\phi = 2.5, 3.0, 3.5$, and 4.0, respectively. This indicates that good coverage of the region $p_1 < r < p_2$ is attained for $\phi = 2.5$, 3.0 and 3.5. On the other hand, there is no coverage outside this interval for $\phi = 2.5$, showing that this PSO concentrates all its search between the attractors. At $\phi = 3.0$, particles will move outside this region, enhancing exploration away from the fixed points in the interactive model (where p_1 and p_2 can be updated). At $\phi = 3.5, 4.0$, the frequent bursts often take r far from the attractors. These large bursts cannot help with convergence, but they might help diversify the fully coupled system.

An analysis of the data for the 50 runs at $\phi = 4.0, w = 0.75$ found a probability of 10 per cent for positions of at least 10^{40}. $\phi = 4.0$ is between the $u = 1$ and $u = 0.5$ stability conditions (3.5 and 7.0, resp.). Although none of these runs exploded, bursts of extremely high amplitude were common. The inference from these experiments is that the $u = 1$ stability condition corresponds to power law tails with bounded mean. Moving ϕ beyond the $u = 1$ condition leads to unbounded mean displacements and little exploration of the region between the attractors. This may explain the popular empirical choices $w \approx 0.73$ and $\phi_{\mathrm{IW}} \approx 3.0$.

4. MULTIPLICATIVE RANDOMNESS

The PSO dynamics can produce, under stagnation, outlying particles. However, these outliers particles are not isolated; rather, large excursions exist in bursts or sequences of increasing and then decreasing amplitudes away from the origin. This must be true because arbitrarily large steps $r(t+1) - r(t)$ are prohibited;

$$\begin{aligned} r(t+1) &< \left| -a(t)x(t) - bx(t-1) + c(t) \right| \\ &< \max(|a|)r(t) + br(t-1) + \max(|c|). \end{aligned} \tag{17}$$

(This is in contrast to a bare bones formulations which sample from a probability distribution N. N might itself has fat tails but outliers need not be correlated and arbitrarily large steps $r(t+1) - r(t)$ are possible. Bursts are a feature of the finite difference equations.)

The previous section presented an evidence that the decoupled PSO shows power-law behaviour when close to constant-u instability. Power-law tails are found in many natural systems. Well-known examples include the distribution of earthquake magnitudes, frequency of words in a language, wealth of the richest people, and physical quantities close to a phase transition. Although power laws have been regarded as an indicator of self-organisation (e.g., [22]), this explanation is not necessary [21, 23].

Other possible explanations of bursts include resonance. Certainly, (3) has a driving term $c(t)$ and a spring-like term $\Phi(t)(p_i - x(t))$, (1), and might be expected to resonate. However, the system does not have a well-defined resonant frequency because the spring constants Φ are themselves random.

Intermittent chaotic systems show periods of constant amplitude punctuated by erratic bursts [24]. However, decoupled PSO is not chaotic in the stable regime. Another simpler explanation for power laws can be found in the theory of random multiplicative processes [9].

4.1. First-order SDE

Considering the first order SDE

$$x(t+1) = -a(t)x(t), \tag{18}$$

then $x(t) = (-)^t a(t-1)a(t-2) \cdots a(0)x(0)$. The distribution of x is therefore given by the distribution of products of random numbers. The logarithm of $x(t)$ is equal to a sum of logarithms of random numbers, and by the central limit theorem, the distribution of $\log(x)$ will be normal. The distribution of $x(t)$ is therefore log normal, and log-normal laws are well approximated by power laws over intervals of four or less orders of magnitude [21]. This simple argument shows that fat, power-law tails can derive from first-order multiplicative processes.

However, (18) is a very poor approximation to PSO. The second-order SDE defined by (3) reduces to the first-order SDE considered above for $b = 0$, $c(t) = 0$. This corresponds to a PSO with $w = 0$ and $\sum \Phi_i p_i = 0$. This implies that $u_1 = u_2$ and $p_1 + p_2 = 0$; giving a PSO,

$$x(t+1) = x(t) + \frac{\phi}{2}\left[\left(p_1 - u_3 x(t)\right) + \left(p_2 - u_3 x(t)\right)\right]r, \tag{19}$$

where u_3 is a random value. (19) was tested over a suite of 14 objective functions, duplicating the test conditions of [11], with very poor results.

The first-order SDE with additive noise,

$$x(t+1) + a(t)x(t) = c(t), \tag{20}$$

has been studied by Sornette and other workers (e.g., see [9] for $a(t) < 0$). This system contains both multiplicative $a(t)$ and additive $c(t)$ randomness. Decoupled PSO reduces to (20) if the inertia weight is set to zero, $w = 0$,

$$x(t+1) = x(t) + \frac{\phi}{2}\left[u_1\left(p_1 - x(t)\right) + u_2\left(p_2 - x(t)\right)\right]. \tag{21}$$

Once more, performance of the fully coupled version of (21) is very poor over the above test function set. Without velocity, these PSOs cannot move through the search space and are doomed to local exploration around the initial particle configuration.

Equation (20) exhibits a regime of power-law behaviour. With $c(t) = 0$, we recover model (18) which is log normal in its central part [25]. For c finite, iterating (19) gives the solution of (19) as

$$x(n) = \left(\prod_{l=0}^{n-1} a(l)\right)x(0) + \left(\sum_{l=0}^{n-2} c(l) \prod_{m=l+1}^{n-1} a(m)\right) + c(n-1) \tag{22}$$

which shows that the fate of x is determined by the multiplications over a. The surprising feature is that (21) exhibits interesting behaviour in the *stable* regime $\langle a \rangle < 1$ [9]. This behaviour, namely intermittent bursts and power law tails to the distribution of x, is contingent on $\max(a(t)) > 1$ so that amplification is possible, and upon the injection of noise, $c \neq 0$ so that convergence to the fixed point is prevented.

Rewriting the $w = 0$ PSO as

$$x(t+1) + \left[\frac{\phi}{2}(u_1 + u_2) - 1\right]x(t) = \frac{\phi}{2}(u_1 p_1 + u_2 p_2) \tag{23}$$

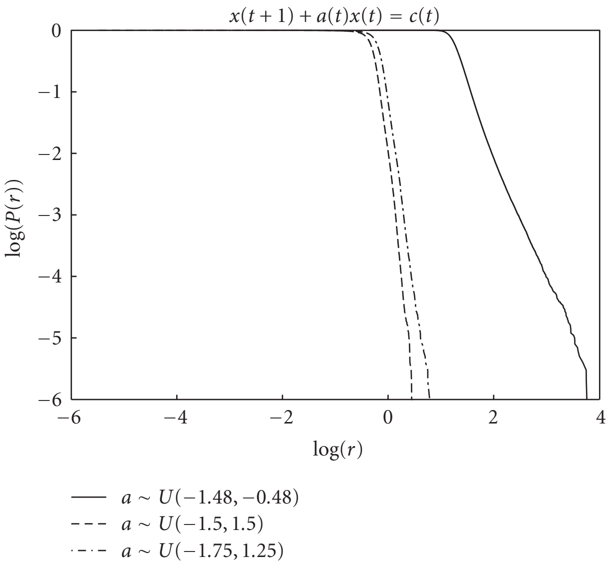


Figure 6: Investigation of stochastic first-order equation for various ranges of random variables $a(t)$.

facilitates the comparison with (20). The fixed- u stability condition is $0 < \phi u < 2$. Without loss of generality, we can place $p_1 = 1.0$, $p_2 = 0$ so that $c(t) = (\phi/2)u_1$. From the $u = 1$ stability condition, $\phi < 2$, $c(t) \sim U(0, c_{\max})$, $c_{\max} < 1$. Furthermore, $a(t) \in [-1, \phi]$, although the distribution within this interval is triangular rather than uniform. This means that the $w = 0$ PSO differs from (20) in two respects: $a(t)$ can become positive and c and a are not independent. Indeed, $a(t) = [c(t) + (\phi/2)u_2 - 1]$.

These changes were investigated by trials on Sornette and Cont's original system, (20) with $a \sim U(a_{\min}, a_{\max})$ and $c(t) \sim U(0, 1)$. The results are shown in Figure 6. The plots depict average distances $r = |x|$ from the origin over 50 runs of 10^6 iterations and show results for four uniform distributions of a, each with $|\langle a \rangle| < 1.0$ and $\max(|a| > 1)$. Line (i), $a \sim U(-1.48, -0.48)$, corresponds to the system previously studied by Sornette and Cont [26], line (ii), $a \sim U(-1.5, 1.5)$, is a symmetrical distribution with $\langle a \rangle = 0$, and line (iii), $a \sim U(-1.75, 1.25)$, has $\langle a \rangle = -0.25$.

The plot shows the fat power-law tail for the Sornette-Cont system. The results also show thin tails when the mean a is close to zero (lines (ii) and (iii)). This can be explained by the relative drop in probability of amplification, $|a| > 1$. In SDEs (ii) and (iii), $\text{prob}(|a| > 1) = 1/3$, whereas in the Sornette-Cont system, $\text{prob}(|a| > 1) = 0.48$. The second term of (22) can be ignored during a large burst $|x(n)| \gg |x(0)| \gg \max(c)$,

$$x(n) \approx \prod_{l=0}^{n-1} a(l)x(0). \tag{24}$$

Sequences of $a(l)$ with large products will occur less often in (ii) and (iii) compared to (i), and the distribution tail is quenched. A comparison of SDEs (ii) and (iii) reveals a slightly fatter tail for (iii). This is because $\max(|a|)$ is larger in this system, and amplification is increased because products of a's of a given length n, $\prod_{l=0}^{n-1} a(l)x(0)$, can attain higher values.

In summary, power-law tail behaviour is possible in first-order SDEs, although the fatness of the tail will depend on the choice of parameters; fattening bursts are possible if $\max(|a|) > 1$ and the distribution assumes a power-law tail for finite c. Standard PSO is badly approximated by a first-order SDE (however, a novel first-order PSO variant, a simplified version of recombinant PSO, does produce good search behaviour [27]). The next section examines second-order SDEs which provide a much closer model of standard PSO.

4.2. Second-order SDE

The second-order stochastic system with uniform distributions

$$x(t+1) + U(a_l, a_u)x(t) + bx(t-1) = U(c_l, c_u) \tag{25}$$

has not, to our knowledge, been studied in the burst regime $\max(|a|) > 1$. (25) is closely related to the decoupled PSO with $p_1 = 1$, $p_2 = 0$:

$$x(t+1) + \left[\frac{\phi}{2}(u_1 + u_2) - w - 1\right]x(t) + wx(t-1) = \frac{\phi}{2}u_1. \tag{26}$$

Replacing $u_1 + u_2$ with a uniform distributions leads to

$$x(t+1)+U(-w-1,\ \phi-w-1)x(t)+wx(t-1)=U\left(0, \frac{\phi}{2}\right) \tag{27}$$

and in particular, for the $\phi = 3.0$, $w = 0.75$ system,

$$x(t+1) + U(-1.75, 1.25)x(t) + 0.75x(t-1) = U(0, 1.5). \tag{28}$$

Figure 7 shows the cumulative distribution $r = |x|$ for (28) for 50 runs of 10^6 iterations. Comparison with the first-order SDEs of Figure 6 shows that this second-order SDE has very fat tails with 10% of all positions at distances of 4×10^5 or more from the origin. The frequent and large bursts are not well modelled by a power law.

This behaviour is in contrast to the decoupled PSO of (26), plotted in Figure 3, which shows a very much more restrained tail. The principle difference between (26) and (28) is the replacement of the triangular-shaped random number distribution by a uniform distribution. Although $\langle a \rangle$ and $\max(|a|)$ are identical, the uniform distribution increases $\text{prob}(|a| > 1)$, leading to an increased probability of large products $\prod_{l=0}^{n-1} a(l)$.

The contribution of such products to a burst can be seen by formally deriving a solution for $x(t)$ as a sum over products of random variables. Rewriting (25) as

$$x_n = (La_n - bL^2)x_n + c_n, \tag{29}$$

where L is the lag operator $Lx_j = x_{j-1}$, $L^2x_j = x_{j-2}$, $La_j = a_{j-1}L$, $a(t) = -a_j$, and iterating back in time,

$$x_n = (La - bL^2)(La - bL^2)x_n + (La - bL^2)c_n + c_n. \tag{30}$$

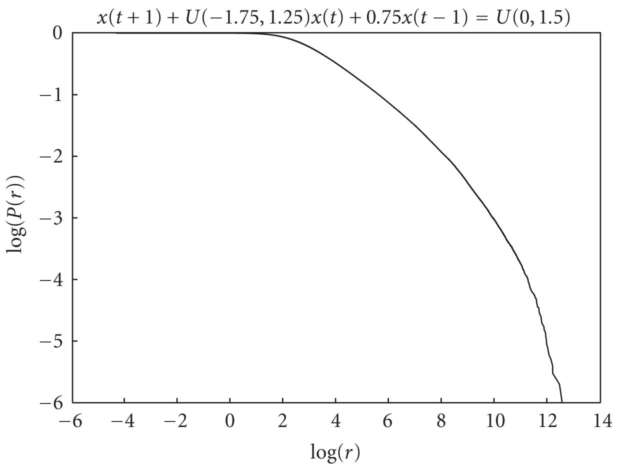


FIGURE 7: Investigation of a stochastic second-order equation.

Hence, after m iterations,

$$x_n = (La - bL^2)^m x_n + \left(\sum_{j=0}^{m-1} (La - bL^2)^j\right)c_n. \tag{31}$$

Equation (31) reduces to the solution of the first-order difference equation, (22), for $m = n$, $b = 0$.

During a large burst $|x_n| \gg x_{n-2m} \gg \max(c)$,

$$\begin{aligned} x_n &\approx (La - bL^2)^m x_n \\ &= ((La)^m - (La)^{m-1}b - (La)^{m-2}b(La) - \cdots)x_n \end{aligned} \tag{32}$$

which shows that the amplification of the burst is enhanced compared to a first-order burst (24), by the terms in b. The validity of this remark depends on the relative sign of each term appearing in (32). To estimate just how big the enhancement might be, suppose that all the a's are positive and the x's alternate in sign. The probability distribution function $P_m(A)$ of the product of the random multipliers $A = a_1a - 2 \cdots a_m$ is approximated by $P_m(A) = (p(A^{1/m})^m$ (see, e.g., [9]) for large A, where $p(a)$ is the distribution of a. Then, for large A, we can replace each random variable a in (31) by the geometric mean $a = A^{1/m}$. This gives an upper bound to x_m, $x_m < ((a + |b|)^m x_{m-1}$ where (by assumption) $x_{n-1} > x_{n-2m}$. This second order burst is boosted in comparison to a first-order burst $x_n = a^m x_{m-n}$. Although such large bursts will be rare, they will contribute to the overall probability distribution of x because, although exponentially unlikely, they are of exponential size (see, e.g., the argument of Redner [25] for the product of a binary sequence $xyxxyxyy$).

The oscillatory pattern of any large burst can be deduced by considering $x_{n+1} = a_nx_n - 0.75x_{n-1}$ where $|x_{n+1}| > |x_n| > |x_{n-1}| \gg \max(c)$. Suppose, for the sake of argument, that x_{n-1} is positive. Then $|x_{n+1}|$ will be maximised if $a_nx_n < 0$ with the result $x_{n+1} < 0$. Hence, x_{n+1} and x_{n-1} differ in sign. x_n might be positive or negative; in either case, we expect x_{n+2} to be of the reverse sign in a large burst. The conclusion is that large bursts are likely to follow a pattern $\text{sign}(x) = \{++--++--\}$ as demonstrated in a close-up of the burst of Figure 8.

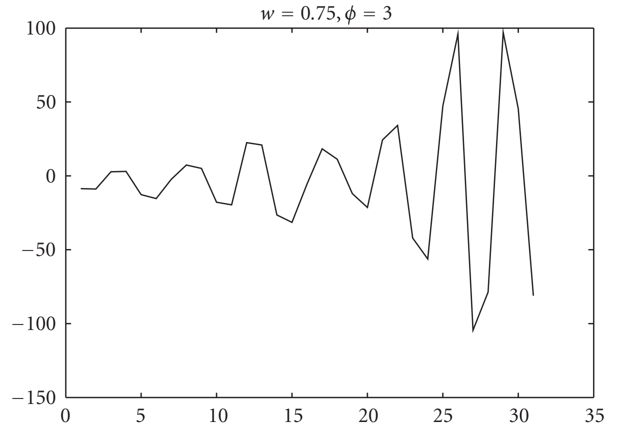


FIGURE 8: Start of a large burst, with deviations $\{+ + - - + + - -\}$ about $x = 0$.

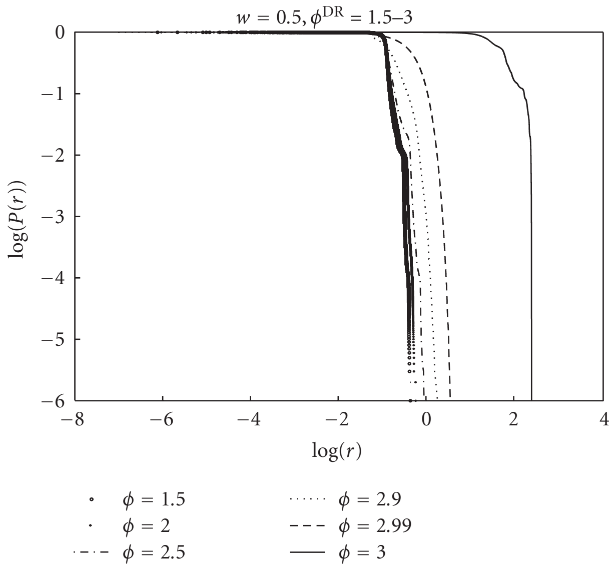


FIGURE 9: Plot of cumulative probability r for PSO-DR.

5. ADDITIVE RANDOMNESS

If distribution tails in SDEs are caused solely by multiplicative randomness, a second-order SDE with only additive randomness, that is, $a, b = \text{const.}$, $c = c(t)$ should be tail free. Recently, a novel PSO variant, recombinant PSO (DR), has been proposed with only additive randomness [10]. Recent work tests PSO-DR for various neighbourhoods and parameter choices with impressively competitive results over a suite of 14 common benchmarks [28]. PSO-DR is similar to the PSO-IW, (9), except that one of the informers is replaced by a discrete recombination of a particle's immediate neighbours in a ring topology,

$$\text{DR}: v(t+1) = wv(t) + \frac{\phi^{\text{DR}}}{2}\left[\left(p_1 - x(t)\right) + \left(p_2 - x(t)\right)\right], \tag{33}$$

where $p_2 = \eta p_l + (1-\eta)p_r$, $\eta = U\{0,1\}$, and p_l and p_r are left and right neighbours and p_1 is either the personal best position of particle i, or the best position in i's neighbourhood, depending on the particular formulation of PSO-DR. The stability condition from (11) is $0 < \phi^{\text{DR}} < 2(1+w)$.

Figure 9 reports on the cumulative distribution of particle separation r for (33). The distributions for $w = 0.5$ and various ϕ up to the maximum stable value of $\phi = 3.0$ were collected for 50 runs of 10^6 iterations with $x(1), x(2) \sim U(-0.5, 0.5)$, $p_1 = -0.5$, $p_l = 0.5$, $p_r = 1.0$.

The cumulative distributions are flat for small r, and then drop vertically at a cutoff $r - c$, suggesting $p(0 \geq r \geq r_c) = U(0, r_c)$ (although the log-log plot is not sensitive enough to show variations from uniformity). The noncritical systems $\phi \leq 2.9$ place the majority of the positions between the attractors. At subcriticality, $\phi = 2.99$, the system is inclined to explore beyond the attractors, $r_c > 1$. At instability, r_c is between 50 and 100. A vertical drop-off beyond r_c would be an evidence that additive-SDE does not develop tails, and this is confirmed by these plots, except perhaps for $\phi = 2.99$ which appears to have a finite, but very large slope.

In fact, PSO models such as (33) might produce tails from a resonance effect. This is because the spring constants are fixed and the system has a defined natural frequency. For the case of PSO-DR, setting $p_3 = (p_1 + p_2)/2$ gives a simple oscillator with force law $F = \phi(p_3 - x)$ and natural frequency $\omega = \sqrt{(\phi)}$. The periodic time, $T = 2\pi/\omega$ for $\phi = 1$ (this is empirically a good value for interacting PSO-DR [28]), is therefore about 6 with the implication that an oscillating p_3 on the timescale of 6 iterations could drive the oscillator and amplify x. This could happen from a shifting neighbour best position p_1, or from an oscillation between p_l and p_r in the p_2 term, or by a combination of the two.

6. CONCLUSIONS

This paper has investigated the position distribution of decoupled PSO. Particular attention has been paid to the tail of this distribution, a regime dominated by power laws. The origin of these tails lies with the phenomenon of particle bursts. Bursts occur at all scales with decreasing probability for increasing size. The accumulation of bursts of all sizes results in distribution tail fattening. In order to study how these bursts might develop, decoupled PSO has been formulated as a second-order stochastic difference equation. Fat distribution tails, well modelled by power laws (although the underlying distribution might be log normal, or some other distribution with a power-law regime), arise from multiplicative randomness, a phenomenon previously encountered in first-order SDEs, and generalised here to second order processes.

This conclusion is valid for first- or second-order SDEs where the multiplicative random variable a is capable of amplification $(\max(|a|) > 1)$, but does not permit system explosion $(\langle |a| \rangle < 1)$. According to the theoretical analysis presented here, bursting in a second-order SDE is built from a sum of multiplicative stochastic processes, and the burst size is boosted by the second-order parameter b. This result explains the observation that the particle distribution shifts as more iterations are collected. The distribution is dominated by large but rare events that are only manifest after many iterations.

A stability condition for the PSO parameters w and ϕ can be achieved by replacing the random variables in the difference equation by a constant u. The inference from a set of experiments is that the $u = 1$ stability condition leads to power-law tails with bounded mean. Moving ϕ beyond this condition leads to unbounded mean displacements. The popular $\phi = 3.0, w = 0.75$ PSO, is within the stable region and has weak power-law tails, which enhance exploration, yet also has good coverage of the region close to the attractors, enabling convergence.

There is a tantalising possibility that the removal of stochasticity from the dynamics might render PSO amenable to further theoretical analysis. A recombinant PSO, which is demonstrably competitive to standard PSO, almost achieves this miracle. The acceleration parameters are constant in PSO-DR, but randomness, and hence diversity regeneration, is manifest in a jiggling of the attractor components. This jiggling will persist even at times of stagnation. PSO-DR, replete with just additive randomness of this sort, does not, according to the theoretical and empirical arguments supplied here, enjoy bursting activity. It is conjectured that this transfer of randomness from the dynamics to the information network generates enough noise to mitigate against premature convergence in the coupled model, despite the thin tails of the decoupled equations. These issues are taken up in [27, 28]. (See also [29].)

PSO bursts differ from the outliers generated by bare bones swarms in two respects: the outliers occur in sequence, and they are one dimensional. Bursting will therefore produce periods of rectilinear motion where the particle will have a large velocity parallel to a coordinate axis. Whether bursts are generally beneficial, or a hindrance, to a fully interactive PSO and under what circumstances, is the subject of ongoing research [28]. What does appear to be certain is that a distribution of outliers which decreases slower than a Gaussian tail is important if PSO is to escape premature convergence in difficult environments. Standard PSO achieves this through multiplicative randomness and this occurs even in the decoupled system; recombinant PSO can only, however, gain its outliers through the interaction of the particles.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of EPSRC XPS Project (GR/T11234/01).

REFERENCES

[1] A. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, New York, NY, USA, 2005.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, 1995.

[3] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intellligence*, vol. 1, no. 1, pp. 33–57, 2007.

[4] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 80–87, Indianapolis, Ind, USA, April 2003.

[5] T. Richer and T. Blackwell, "The Lévy particle swarm," in *IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 3150–3157, Vancouver, Canada, July 2006.

[6] R. Poli, W. Langdon, M. Clerc, and C. Stephens, "Continuous optimisation made easy? Finite element models of evolutionary strategies, genetic algorithms and particle swarm optimisers," in *Proceedings of the Foundations of Genetic Algorithms Workshop (FOGA '07)*, Mexico City, Mexico, January 2007.

[7] J. Kennedy, "Dynamic-probabilistic particle swarms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 201–207, Washington, DC, USA, June 2005.

[8] J. Kennedy, "Probability and dynamics in the particle swarm," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 1, pp. 340–347, Portland, Ore, USA, June 2004.

[9] D. Sornette, "Multiplicative processes and power laws," *Physical Review E*, vol. 57, no. 4, pp. 4811–4813, 1998.

[10] J. Peña, A. Upegui, and E. Sanchez, "Particle swarm optimization with discrete recombination: an online optimizer for evolvable hardware," in *Proceedings of the1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS '06)*, pp. 163–170, Istanbul, Turkey, June 2006.

[11] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, Honolulu, Hawaii, USA, April 2007.

[12] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[13] J. Kennedy and R. Mendes, "Neighbor topologies in fully-informed and best of-neighborhood particle swarms," in *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications (SMCia '03)*, pp. 45–50, June 2003.

[14] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability and convergence in a multi-dimensional space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2000.

[15] R. Sornette, "Linear stochastic dynamics with nonlinear fractal properties," *Physica A*, vol. 250, no. 1–4, pp. 295–314, 1998.

[16] J. D. Hamilton, *Time Series Analysis*, Princeton University Press, Princeton, NJ, USA, 1994.

[17] E. Ozcan and C. K. Mohan, "Particle swarm optimization: surfing the waves," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, p. 1944, Washington, DC, USA, July 1999.

[18] F. ven den Burgh, *An analysis of particle swarm optimizers*, Ph.D. thesis, University of Pretoria, Pretoria, South Africa, 2002.

[19] R. Poli and D. Broomhead, "Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 134–141, London, UK, July 2007.

[20] G. K. Zipf, *Human Behaviour and the Principle of Least Action*, Addison Wesley, Reading, Mass, USA, 1949.

[21] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, 2005.

[22] C. T. P. Bak and K. Wiesenfeld, "Self-organized criticality: an explanation for 1/f noise," *Physical Review Letters*, vol. 59, no. 4, pp. 381–384, 1987.

[23] D. Sornette, "Mechanism for power laws without self-organization," *International Journal of Modern Physics*, vol. 13, no. 2, pp. 133–136, 2001.

[24] E. Ott, *Chaos in Dynamical Systems*, Cambridge University Press, New York, NY, USA, 2002.

[25] S. Redner, "Random multiplicative processes," *American Journal of Physics*, vol. 58, no. 3, pp. 267–273, 1990.

[26] D. Sornette and R. Cont, "Convergent multiplicative processes repelled from zero: power laws and truncated power laws," *Journal de Physique I France*, vol. 7, pp. 431–444, 1997.

[27] D. Bratton and T. Blackwell, "A simplified recombinant PSO," to appear in *Artificial Evolution and Applications*.

[28] D. Bratton and T. Blackwell, "Understanding particle swarms through simplification: a study of recombinant PSO," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 2621–2628, London, UK, July 2007.

[29] M. Clerc, *Particle Swarm Optimization*, ISTE, London, UK, 2006.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 143624, 14 pages
doi:10.1155/2008/143624

Research Article

Geometric Particle Swarm Optimization

Alberto Moraglio,[1] Cecilia Di Chio,[2] Julian Togelius,[3] and Riccardo Poli[2]

[1] Centre for Informatics and Systems of the University of Coimbra, Polo II - University of Coimbra, 3030-290 Coimbra, Portugal
[2] Department of Computing and Electronic Systems, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
[3] Dalle Molle Institute for Artificial Intelligence (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland

Correspondence should be addressed to Alberto Moraglio, moraglio@dei.uc.pt

Received 21 July 2007; Accepted 4 December 2007

Recommended by T. Blackwell

Using a geometric framework for the interpretation of crossover of recent introduction, we show an intimate connection between particle swarm optimisation (PSO) and evolutionary algorithms. This connection enables us to generalise PSO to virtually any solution representation in a natural and straightforward way. The new geometric PSO (GPSO) applies naturally to both continuous and combinatorial spaces. We demonstrate this for the cases of Euclidean, Manhattan, and Hamming spaces and report extensive experimental results. We also demonstrate the applicability of GPSO to more challenging combinatorial spaces. The Sudoku puzzle is a perfect candidate to test new algorithmic ideas because it is entertaining and instructive as well as being a nontrivial constrained combinatorial problem. We apply GPSO to solve the Sudoku puzzle.

1. INTRODUCTION

Particle swarm optimization (PSO) is a relatively recently devised population-based stochastic global optimization algorithm [1]. PSO has many similarities with evolutionary algorithms, and has also proven to have robust performance over a variety of difficult optimization problems. However, the original formulation of PSO requires the search space to be continuous and the individuals to be represented as vectors of real numbers.

There is a number of extensions of PSO to combinatorial spaces with various degrees of success [2, 3]. (Notice that applications of traditional PSO to combinatorial optimization problems cast as continuous optimization problems are not extensions of the PSO algorithm.) However, every time a new solution representation is considered, the PSO algorithm needs to be rethought and adapted to the new representation. In this article, we extend PSO to richer spaces by making use of a rigorous mathematical generalization of the notion (and motion) of particles to a general class of spaces. This approach has the advantage that a PSO can be derived in a principled way for any search space belonging to the given class.

In particular, we show *formally* how a general form of PSO (without the inertia term) can be obtained by using theoretical tools developed for evolutionary algorithms with geometric crossover and geometric mutation. These are representation-independent operators that generalize many pre-existing search operators for the major representations, such as binary strings [4], real vectors [4], permutations [5], syntactic trees [6], and sequences [7]. (The inertia weight was not part of the original proposal of PSO, it was later introduced by Shi and Eberhart [8].)

Firstly, we formally derive geometric PSOs (GPSOs) for Euclidean, Manhattan, and Hamming spaces and discuss how to derive GPSOs for virtually any representation in a similar way. Then, we test the GPSO theory experimentally: we implement the specific GPSO for Euclidean, Manhattan, and Hamming spaces and report extensive experimental results showing that GPSOs perform very well.

Finally, we also demonstrate that GPSO can be specialized easily to nontrivial combinatorial spaces. In previous work [9], we have used the geometric framework to design an evolutionary algorithm to solve the Sudoku puzzle and obtained very good experimental results. Here, we apply GPSO to solve the Sudoku puzzle.

In Section 2, we introduce the geometric framework and introduce the notion of multiparental geometric crossover. In Section 3, we recast PSO in geometric terms and generalize it to generic metric spaces. In Section 4, we apply these notions to the Euclidean, Manhattan, and Hamming spaces. In Section 5, we discuss how to specialize the general PSO automatically to virtually any solution representation using geometric crossover. Then, in Section 6, we report experimental results with the GPSOs for Euclidean, Manhattan, and Hamming spaces, and we compare them with a traditional PSO. In Section 7, we apply GPSO to Sudoku, and we describe the results in Section 8. Finally, in Section 9, we present conclusions and future work.

2. GEOMETRIC FRAMEWORK

Geometric operators are defined in geometric terms using the notions of line segment and ball. These notions and the corresponding genetic operators are well defined once a notion of distance in the search space is defined. Defining search operators as functions of the search space is opposite to the standard way [10] in which the search space is seen as a function of the search operators employed.

2.1. Geometric preliminaries

In the following, we give necessary preliminary geometric definitions and extend those introduced in [4]. For more details on these definitions, see [11].

The terms *distance* and *metric* denote any real valued function that conforms to the axioms of identity, symmetry, and triangular inequality. A simple connected graph is naturally associated to a metric space via its *path metric*: the distance between two nodes in the graph is the length of a shortest path between the nodes. Distances arising from graphs via their path metric are called *graphic distances*. Similarly, an edge-weighted graph with strictly positive weights is naturally associated to a metric space via a *weighted path metric*.

In a metric space (S, d), a *closed ball* is a set of the form $B(x; r) = \{y \in S \mid d(x, y) \leq r\}$, where $x \in S$ and r is a positive real number called the radius of the ball. A *line segment* is a set of the form $[x; y] = \{z \in S \mid d(x, z) + d(z, y) = d(x, y)\}$, where $x, y \in S$ are called extremes of the segment. Metric ball and metric segment generalize the familiar notions of ball and segment in the Euclidean space to any metric space through distance redefinition. In general, there may be more than one shortest path (*geodesic*) connecting the extremes of a metric segment: the metric segment is the union of all geodesics.

We assign a structure to the solution set by endowing it with a notion of distance d. $M = (S, d)$ is, therefore, a solution *space* and $L = (M, g)$, where g is the fitness function, is the corresponding *fitness landscape*.

2.2. Geometric crossover

Definition 1 (geometric crossover). A binary operator is a geometric crossover under the metric d if all offsprings are in the segment between its parents.

The definition is *representation-independent* and, therefore, crossover is well defined for any representation. Being based on the notion of metric segment, *crossover is a function only of the metric d* associated with the search space.

The class of geometric crossover operators is very broad. For vectors of reals, various types of blend or line crossovers, box recombinations, and discrete recombinations are geometric crossovers [4]. For binary and multary strings, all homologous crossovers are geometric [4, 12]. For permutations, PMX, Cycle crossover, merge crossover, and others are geometric crossovers [5]. We describe this in more detail in Section 2.3 since we will use the permutation representation in this paper. For syntactic trees, the family of homologous crossovers are geometric [6]. Recombinations for several more complex representations are also geometric [4, 5, 7, 13].

2.3. Geometric crossover for permutations

In previous work, we have studied various crossovers for permutations, revealing that PMX [14], a well-known crossover for permutations, is geometric under swap distance. Also, we found that Cycle crossover [14], another traditional crossover for permutations, is geometric under swap distance and under Hamming distance (geometricity under Hamming distance for permutations implies geometricity under swap distance, but not *vice versa*). Finally, we showed that geometric crossovers for permutations based on edit moves are naturally associated with sorting algorithms: picking offspring on a minimum path between two parents corresponds to picking partially sorted permutations on the minimal sorting trajectory between the parents.

2.4. Geometric crossover landscape

Geometric operators are defined as functions of the distance associated with the search space. However, the search space does not come with the problem itself. The problem consists of a fitness function to optimize and a solution set, but not a neighbourhood relationship. The act of putting a structure over the solution set is part of the search algorithm design and it is a designer's choice. A fitness landscape is the fitness function plus a structure over the solution space. So for each problem, there is one fitness function but as many fitness landscapes as the number of possible different structures over the solution set. In principle, the designer could choose the structure to assign to the solution set completely independently from the problem at hand. However, because the search operators are defined over such a structure, doing so would make them decoupled from the problem at hand, hence turning the search into something very close to random search.

In order to avoid this, one can exploit problem knowledge in the search. This can be achieved by carefully designing the connectivity structure of the fitness landscape. For example, one can study the objective function of the problem and select a neighbourhood structure that couples the distance between solutions and their fitness values. Once this is done, the problem knowledge can be exploited by search operators to perform better than random search, even if the search operators are problem independent (as in the case of geometric crossover and mutation).

Under which conditions is a landscape well searchable by geometric operators? As a rule of thumb, geometric mutation and geometric crossover work well on landscapes where the closer pairs of solutions are, the more correlated their fitness values. Of course this is no surprise: the importance of landscape smoothness has been advocated in many different contexts and has been confirmed in uncountable empirical studies with many neighborhood search metaheuristics [15, 16]. To summarize, consider the following.

(i) Rule of thumb 1: If we have a good distance for the problem at hand, then we have good geometric mutation and good geometric crossover.
(ii) Rule of thumb 2: A good distance for the problem at hand is a distance that makes the landscape "smooth."

2.5. Product geometric crossover

In recent work [12], we have introduced the notion of product geometric crossover.

Theorem 1. *Cartesian product of geometric crossover is geometric under the sum of distances.*

This theorem is very useful because it allows one to build new geometric crossovers by combining crossovers that are known to be geometric. In particular, this applies to crossovers for mixed representations. The elementary geometric crossovers do not need to be independent, to form a valid product geometric crossover.

2.6. Multiparental geometric crossover

To extend geometric crossover to the case of multiple parents, we need the following definitions [17].

Definition 2. A family $\mathcal{X}$ of subsets of a set X is called *convexity on* X if

(C1) the empty set $\varnothing$ and the universal set X are in $\mathcal{X}$,
(C2) $\mathcal{D} \subseteq \mathcal{X}$ is nonempty, then $\bigcap \mathcal{D} \in \mathcal{X}$, and
(C3) $\mathcal{D} \subseteq \mathcal{X}$ is nonempty and totally ordered by inclusion, then $\bigcup \mathcal{D} \in \mathcal{X}$.

The pair $(X, \mathcal{X})$ is called *convex structure*. The members of $\mathcal{X}$ are called *convex sets*. By the axiom (C1), a subset A of X of the convex structure is included in at least one convex set, namely, X.

From axiom (C2), A is included in a smallest convex set, the *convex hull* of A:

$$\operatorname{co}(A) = \bigcap \{C \mid A \subseteq C \in \mathcal{X}\}. \tag{1}$$

The convex hull of a finite set is called a *polytope*.

The axiom (C3) requires *domain finiteness* of the convex hull operator: a set C is convex if it includes $\operatorname{co}(F)$ for each finite subset F of C.

The convex hull operator applied to a set of cardinality two is called *segment operator*. Given a metric space $M = (X, d)$, the segment between a and b is the set $[a, b]_d = \{z \in X \mid d(x, z) + d(z, y) = d(x, y)\}$. The abstract *geodetic convexity* $\mathcal{C}$ on X induced by M is obtained as follows: a subset C of X is geodetically convex, provided $[x, y]_d \subseteq C$ for all x, y in C. If co denotes the convex hull operator of $\mathcal{C}$, then for all $a, b \in X : [a, b]_d \subseteq \operatorname{co}\{a, b\}$. The two operators need not to be equal: there are metric spaces in which metric segments are not all convex.

We can now provide the following extension.

Definition 3 (multiparental geometric crossover). In a multiparental geometric crossover, given n parents $p_1, p_2, \dots, p_n$, their offspring are contained in the metric convex hull of the parents $\operatorname{co}(\{p_1, p_2, \dots, p_n\})$ for some metric d.

Theorem 2 (decomposable three-parent recombination). *Every multiparental recombination $RX(p_1, p_2, p_3)$ that can be decomposed as a sequence of 2-parental geometric crossovers under the same metric GX and GX', so that $RX(p_1, p_2, p_3) = GX(GX'(p_1, p_2), p_3)$, is a three-parental geometric crossover.*

Proof. Let P be the set of parents and $\operatorname{co}(P)$ their metric convex hull. By definition of metric convex hull, for any two points $a, b \in \operatorname{co}(P)$, their offspring are in the convex hull $[a, b] \subseteq \operatorname{co}(P)$. Since $P \subseteq \operatorname{co}(P)$, any two parents $p_1, p_2 \in P$ have offspring $o_{12} \in \operatorname{co}(P)$. Then, any other parent $p_3 \in P$, when recombined with o_{12}, produces offspring o_{123} in the convex hull $\operatorname{co}(P)$. So the three-parental recombination equivalent to the sequence of geometric crossover $GX'(p_1, p_2) \to o_{12}$ and $GX(o_{12}, p_3) \to o_{123}$ is a multiparental geometric crossover. □

3. GEOMETRIC PSO

3.1. Canonical PSO algorithm and geometric crossover

Consider the canonical PSO in Algorithm 1. It is well known [18] that one can write the equation of motion of the particle without making explicit use of its velocity.

Let x be the position of a particle and v its velocity. Let $\hat{x}$ be the current best position of the particle and let $\hat{g}$ be the global best. Let v' and v'' be the velocity of the particle and $x' = x + v$ and $x'' = x' + v'$ its position at the next two time ticks. The equation of velocity update is the linear combination: $v' = \omega v + \phi_1(\hat{x} - x') + \phi_2(\hat{g} - x')$, where ω, ϕ_1, and ϕ_2 are scalar coefficients. To eliminate velocities, we substitute the identities $v = x' - x$ and $v' = x'' - x'$ in the equation of velocity update and rearrange it to obtain an equation that expresses x'' as function of x and x' as follows:

$$x'' = (1 + \omega - \phi_1 - \phi_2)x' - \omega x + \phi_1\hat{x} + \phi_2\hat{g}. \tag{2}$$

If we set $\omega = 0$ (i.e., the particle has no inertia), x'' becomes independent on its position x two time ticks earlier. If we call $w_1 = 1 - \phi_1 - \phi_2$, $w_2 = \phi_1$, and $w_3 = \phi_2$, the equation of motion becomes

$$x'' = w_1 x' + w_2\hat{x} + w_3\hat{g}. \tag{3}$$

In these conditions, the main feature that allows the motion of particles is the ability to perform linear combinations

```
(1) for all particle i do
(2)     initialize position $x_i$ and velocity $v_i$
(3) end for
(4) while stop criteria not met do
(5)     for all particle i do
(6)         set personal best $\hat{x}_i$ as best position found so far by the particle
(7)         set global best $\hat{g}$ as best position found so far by the whole swarm
(8)     end for
(9)     for all particle i do
(10)        update velocity using equation
                $v_i(t+1) = \kappa(\omega v_i(t) + \phi_1 U(0,1)(\hat{g}(t) - x_i(t)) + \phi_2 U(0,1)(\hat{x}_i(t) - x_i(t)))$,
            where, typically, either ($\kappa = 0.729$, $\omega = 1.0$) or ($\kappa = 1.0$, $\omega < 1$)
(11)        update position using equation
                $x_i(t+1) = x_i(t) + v_i(t+1)$
(12)    end for
(13) end while
```

Algorithm 1: Standard PSO algorithm.

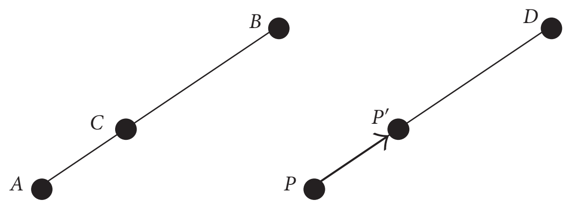


Figure 1: Geometric crossover and particle motion.

of points in the search space. As we will see in the next section, we can achieve this same ability by using multiple (geometric) crossover operations. This makes it possible to obtain a generalization of PSO to generic search spaces.

In the following, we illustrate the parallel between an evolutionary algorithm with geometric crossover and the motion of a particle in PSO (see Figure 1). Geometric crossover picks offspring C on a line segment between parents A and B. Geometric crossover can be interpreted as a motion of a particle: consider a particle P that moves in the direction of a point D reaching, in the next time step, position P'. If one equates parent A with the particle P and parent B with the direction point D, the offspring C is, therefore, the particle at the next time step P'. The distance between parent A and offspring C is the magnitude of the velocity of the particle P. Notice that the particle moves from P to P': this means that the particle P is replaced by the particle P' in the next time step. In other words, the new position of the particle replaces the previous position. Coming back to the evolutionary algorithm with geometric crossover, this means that the offspring C replaces its parent A in the new population. The fact that at a given time all particles move is equivalent to say that each particle is selected for "mating." Mating is a weighted multirecombination involving the memory of the particle and the best in the current population.

In the standard PSO, weights represent the propensity of a particle towards memory, sociality, and cognition. In the GPSO, they represent the attractions towards the particle's previous position, the particle's best position, and the swarm's best position.

Naturally, particle motion based on geometric crossover leads to a form of search that cannot extend beyond the convex hull of the initial population. Mutation can be used to allow nonconvex search. We explain these ideas in detail in the following sections.

3.2. Geometric interpretation of linear combinations

Definition 4. A *convex combination* of vectors $v_1, \dots, v_n$ is a linear combination $a_1v_1 + a_2v_2 + a_3v_3 + \cdots + a_nv_n$, where all coefficients $a_1, \dots, a_n$ are nonnegative and add up to 1.

It is called "convex combination" because when vectors represent points in space, the set of all convex combinations constitutes the convex hull.

A special case is $n = 2$, where a point formed by the convex combination will lie on a straight line between two points. For three points, their convex hull is the triangle with the points as vertices.

Theorem 3. *In a PSO with no inertia ($\omega = 0$) and where acceleration coefficients are such that $\phi_1 + \phi_2 \le 1$, the next position x' of a particle is within the convex hull formed by its current position x, its local best $\hat{x}$, and the swarm best $\hat{g}$.*

Proof. As we have seen in Section 3.1, when $\omega = 0$, a particle's update equation becomes the linear combination in (3). Notice that this is an affine combination since the coefficients of x', $\hat{x}$, and $\hat{g}$ add up to 1. Interestingly, this means that the new position of the particle is coplanar with x', $\hat{x}$, and $\hat{g}$. If we restrict w_2 and w_3 to be positive and their sum to be less than 1, (3) becomes a convex combination. *Geometrically, this means that the new position of the particle is in the convex hull formed by (or more informally, is between) its previous position, its local best, and the swarm best.* □

```
(1) for all particle i do
(2)     initialize position $x_i$ at random in the search space
(3) end for
(4) while stop criteria not met do
(5)     for all particle i do
(6)         set personal best $\hat{x}_i$ as best position found so far by the particle
(7)         set global best $\hat{g}$ as best position found so far by the whole swarm
(8)     end for
(9)     for all particle i do
(10)            update position using a randomized convex combination
                    $x_i = CX((x_i, w_1), (\hat{g}, w_2), (\hat{x}_i, w_3))$
(11)        mutate $x_i$
(12)    end for
(13) end while
```

Algorithm 2: Geometric PSO algorithm.

In the next section, we generalize this simplified form of PSO from real vectors to generic metric spaces. As mentioned above, mutation will be required to extend the search beyond the convex hull.

3.3. Convex combinations in metric spaces

Linear combinations are well defined for vector spaces: algebraic structures endowed with scalar product and vectorial sum. A metric space is a set endowed with a notion of distance. The set underlying a metric space does not normally come with well-defined notions of scalar product and sum among its elements. Therefore, a linear combination of its elements is not defined. How can we then define a convex combination in a metric space? Vectors in a vector space can easily be understood as points in a metric space. However, the interpretation of scalars is not as straightforward: what do the scalar weights in a convex combination mean in a metric space?

As seen in Section 3.2, a convex combination is an algebraic description of a convex hull. However, even if the notion of convex combination is not defined for metric spaces, convexity in metric spaces is still well defined through the notion of metric convex set that is a straightforward generalization of traditional convex set. Since convexity is well defined for metric spaces, we still have hope to generalize the scalar weights of a convex combination trying to make sense of them in terms of distance.

The weight of a point in a convex combination can be seen as a measure of relative linear "attraction" toward its corresponding point, versus attractions toward the other points of the combination. The closer a weight to 1, the stronger the attraction to the corresponding point. The point resulting from a convex combination can be seen as the equilibrium point of all the attraction forces. The distance between the equilibrium point and a point of the convex combination is, therefore, a decreasing function of the level of attraction (weight) of the point: the stronger the attraction, the smaller its distance to the equilibrium point. This observation can be used to reinterpret the weights of a convex combination in a metric space as follows: $y = w_1x_1 + w_2x_2 + w_3x_3$ with w_1, w_2, and w_3 greater than zero and $w_1 + w_2 + w_3 = 1$ is generalized to mean that y is a point such that $d(x_1, y) = f(w_1)$, $d(x_2, y) = f(w_2)$, and $d(x_3, y) = f(w_3)$, where f is a decreasing function.

This definition is formal and valid for all metric spaces, but it is nonconstructive. In contrast, a convex combination not only defines a convex hull, but also tells how to reach all its points. So how can we actually pick a point in the convex hull respecting the above distance requirements? Geometric crossover will help us with this, as we show in the next section.

To summarize, the requirements for a convex combination in a metric space are as follows.

(1) *Convex weights*: the weights respect the form of a convex combination: $w_1, w_2, w_3 > 0$ and $w_1 + w_2 + w_3 = 1$.
(2) *Convexity*: the convex combination operator combines x_1, x_2, and x_3 and returns a point in theirmetric convex hull (or simply triangle) under the metric of the space considered.
(3) *Coherence between weights and distances*: the distances to the equilibrium point are decreasing functions of their weights.
(4) *Symmetry*: the same value assigned to w_1, w_2, or w_3 has the same effect (e.g., in a equilateral triangle, if the coefficients have all the same value, the distances to the equilibrium point are the same).

3.4. Geometric PSO algorithm

The generic GPSO algorithm is illustrated in Algorithm 2. This differs from the standard PSO (Algorithm 1), in that:

(i) there is no velocity,
(ii) the equation of position update is the convex combination,
(iii) there is mutation, and
(iv) the parameters w_1, w_2, and w_3 are nonnegative and add up to one.

The specific PSOs for the Euclidean, Manhattan, and Hamming spaces use the randomized convex combination operators described in Section 4 and space-specific mutations. The randomization introduced by the randomized convex combination and by the mutation are of different types. The former allows us to pick points at random exclusively within the convex hull. The latter, as mentioned in Section 3.1, allows us to pick points outside the convex hull.

4. GEOMETRIC PSO FOR SPECIFIC SPACES

4.1. Euclidean space

The GPSO for the Euclidean space is *not* an extension of the traditional PSO. We include it to show how the general notions introduced in the previous section materialize in a familiar context. The convex combination operator for the Euclidean space is the traditional convex combination that produces points in the traditional convex hull.

In Section 3.3, we have mentioned how to interpret the weights in a convex combination in terms of distances. In the following, we show analytically how the weights of a convex combination affect the relative distances to the equilibrium point. In particular, we show that the relative distances are decreasing functions of the corresponding weights.

Theorem 4. *In a convex combination, the distances to the equilibrium point are decreasing functions of the corresponding weights.*

Proof. Let a, b, and c be three points in $\mathbb{R}^n$ and let $x = w_a a + w_b b + w_c c$ be a convex combination. Let us now decrease w_a to $w'_a = w_a - \Delta$ such that w'_a, w'_b, and w'_c still form a convex combination and that the relative proportions of w_b and w_c remain unchanged: $w'_b/w'_c = w_b/w_c$. This requires w'_b and w'_c to be $w'_b = w_b(1+\Delta/(w_b+w_c))$ and $w'_c = w_c(1+\Delta/(w_b+w_c))$. The equilibrium point for the new convex combination is

$$x' = (w_a - \Delta)a + w_b(1 + \Delta/(w_b + w_c))b + w_c(1 + \Delta/(w_b + w_c))c. \quad (4)$$

The distance between a and x is

$$|a - x| = |w_b(a - b) + w_c(a - c)|, \quad (5)$$

and the distance between a and the new equilibrium point is

$$\begin{aligned} |a - x'| &= |w_b(1 + \Delta/(w_b + w_c))(a - b) \\ &\quad + w_c(1 + \Delta/(w_b + w_c))(a - c)| \\ &= (1 + \Delta/(w_b + w_c))|a - x|. \end{aligned} \quad (6)$$

So when w_a decreases ($\Delta > 0$) and w_b and w_c maintain the same relative proportions, the distance between the point a and the equilibrium point x increases ($|a - x'| > |a - x|$). Hence, the distance between a and the equilibrium point is a decreasing function of w_a. For symmetry, this applies to the distances between b and c and the equilibrium point: they are decreasing functions of their corresponding weights w_b and w_c, respectively. □

The traditional convex combination in the Euclidean space respects the four requirements for a convex combination presented in Section 3.3.

4.2. Manhattan space

In the following, we first define a multiparental recombination for the Manhattan space and then prove that it respects the four requirements for being a convex combination presented in Section 3.3.

Definition 5 (box recombination family). Given two parents a and b in $\mathbb{R}^n$, a box recombination operator returns offspring o such that $o_i \in [\min(a_i, b_i), \max(a_i, b_i)]$ for $i = 1, \dots, n$.

Theorem 5 (geometricity of box recombination). *Any box recombination is a geometric crossover under Manhattan distance.*

Proof. Theorem 5 is an immediate consequence of the product geometric crossover (Theorem 1). □

Definition 6 (three-parent box recombination family). Given three parents a, b, and c in $\mathbb{R}^n$, a box recombination operator returns offspring o such that $o_i \in [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$ for $i = 1, \dots, n$.

Theorem 6 (geometricity of a three-parent box recombination). *Any three-parent box recombination is a geometric crossover under Manhattan distance.*

Proof. We prove it by showing that any multiparent box recombination $BX(a, b, c)$ can be decomposed as a sequence of two simple box recombinations. Since the simple box recombination is geometric (Theorem 5), this theorem is a simple corollary of the multiparental geometric decomposition theorem (Theorem 2).

We will show that $o' = BX(a, b)$ followed by $BX(o', c)$ can reach any offspring $o = BX(a, b, c)$. For each i, we have $o_i \in [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$. Notice that $[\min(a_i, b_i), \max(a_i, b_i)] \cup [\min(a_i, c_i), \max(a_i, c_i)] = [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$. We have two cases: (i) $o_i \in [\min(a_i, b_i), \max(a_i, b_i)]$ in which case o_i is reachable by the sequence $BX(a, b)_i \rightarrow o_i, BX(o, c)_i \rightarrow o_i$; (ii) $o_i \notin [\min(a_i, b_i), \max(a_i, b_i)]$, then it must be in $[\min(a_i, c_i), \max(a_i, c_i)]$ in which case o_i is reachable by the sequence $BX(a, b)_i \rightarrow a_i, BX(a, c)_i \rightarrow o_i$. □

Definition 7 (weighted multiparent box recombination). Given three parents a, b, and c in $\mathbb{R}^n$ and weights w_a, w_b, and w_c, a weighted box recombination operator returns offspring o such that $o_i = w_{a_i} a_i + w_{b_i} b_i + w_{c_i} c_i$ for $i = 1, \dots, n$, where w_{a_i}, w_{b_i}, and w_{c_i} are a convex combination of randomly perturbed weights with expected values w_a, w_b, and w_c.

The difference between box recombination and linear recombination (Euclidean space) is that in the latter, the weights w_a, w_b, and w_c are randomly perturbed only once and the same weights are used for all the dimensions, whereas the former one has a different randomly perturbed version of the weights for each dimension.

The weighted multiparent box recombination belongs to the family of multiparent box recombination because $o_i = w_{a_i} a_i + w_{b_i} b_i + w_{c_i} c_i \in [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$ for $i = 1, \dots, n$, hence, it is geometric.

Theorem 7 (coherence between weights and distances). *In weighted multiparent box recombination, the distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof. The proof of Theorem 7 is a simple variation of that of Theorem 4. □

In summary, in this section, we have introduced the weighted multiparent box recombination and shown that it is a convex combination operator satisfying the four requirements of a metric convex combination for the Manhattan space: convex weights (Definition 6), convexity (geometricity, Theorem 6), coherence (Theorem 7), and symmetry (self evident).

4.3. Hamming space

In this section, we first define a multiparental recombination for binary strings, that is, a straightforward generalization of mask-based crossover with two parents and then prove that it respects the four requirements for being a convex combination in the Hamming space presented in Section 3.3.

Definition 8 (three-parent mask-based crossover family). Given three parents a, b, and c in $\{0,1\}^n$, generate randomly a crossover mask of length n with symbols from the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Build the offspring o filling each position with the bit from the parent appearing in the crossover mask at the corresponding position.

The weights w_a, w_b, and w_c of the convex combination indicate, for each position in the crossover mask, the probability of having the symbols a, b, or c.

Theorem 8 (geometricity of three-parent mask-based crossover). *Any three-parent mask-based crossover is a geometric crossover under Hamming distance.*

Proof. We prove it by showing that any three-parent mask-based crossover can be decomposed as a sequence of two simple mask-based crossovers. Since the simple mask-based crossover is geometric, this theorem is a simple corollary of the multiparental geometric decomposition theorem (Theorem 2).

Let m_{abc} be the mask to recombine a, b, and c, producing the offspring o. Let m_{ab} be the mask obtained by substituting all occurrences of c in m_{abc} with b, and let m_{bc} be the mask obtained by substituting all occurrences of a in m_{abc} with b. First, recombine a and b using m_{ab} obtaining b'. Then, recombine b' and c using m_{bc} where the b's in the mask stand for alleles in b'. The offspring produced by the second crossover is o, so the sequence of the two simple crossovers is equivalent to the three-parent crossover. This is because the first crossover passes to the offspring all genes it needs to take from a according to m_{abc}, and the rest of the genes are all from b; the second crossover corrects those genes that should have been taken from parent c according to m_{abc}, but were taken from b instead. □

Theorem 9 (coherence between weights and distances). *In the weighted three-parent mask-based crossover, the distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof. We want to know the expected distance from parent p_1, p_2, and p_3 to their expected offspring o as a function of the weights w_1, w_2, and w_3. To do so, we first determine, for each position in the offspring, the probability of it being the same as p_1. Then from that, we can easily compute the expected distance between p_1 and o. We have that

$$\begin{aligned}\text{pr}\{o = p_1\} = {} & \text{pr}\{p_1 \longrightarrow o\} + \text{pr}\{p_2 \longrightarrow o\} \cdot \text{pr}\{p_1 \mid p_2\} \\ & + \text{pr}\{p_3 \longrightarrow o\} \cdot \text{pr}\{p_1 \mid p_3\},\end{aligned} \tag{7}$$

where $\text{pr}\{o = p_1\}$ is the probability of a bit of o at a certain position to be the same as the bit of p_1 at the same position; $\text{pr}\{p_1 \to o\}$, $\text{pr}\{p_2 \to o\}$, and $\text{pr}\{p_3 \to o\}$ are the probabilities that a bit in o is taken from parents p_1, p_2, and p_3, respectively (these coincide with the weights of the convex combination w_1, w_2, and w_3); $\text{pr}\{p_1 \mid p_2\}$ and $\text{pr}\{p_1 \mid p_3\}$ are the probabilities that a bit taken from p_2 or p_3 coincides with the one in p_1 at the same location. These last two probabilities equal the number of common bits in p_1 and p_2 (and p_1 and p_3) over the length of the strings n. So $\text{pr}\{p_1 \mid p_2\} = 1 - H(p_1, p_2)/n$ and $\text{pr}\{p_1 \mid p_3\} = 1 - H(p_1, p_3)/n$, where $H(\cdot, \cdot)$ is the Hamming distance. So (7) becomes

$$\begin{aligned}\text{pr}\{o = p_1\} = {} & w_1 + w_2(1 - H(p_1, p_2)/n) \\ & + w_3(1 - H(p_1, p_3)/n).\end{aligned} \tag{8}$$

Hence, the expected distance between the parent p_1 and the offspring o is $E(H(p_1, o)) = n \cdot (1 - \text{pr}\{o = p_1\}) = w_2 H(p_1, p_2) + w_3 H(p_1, p_3)$.

Notice that this is a decreasing function of w_1 because increasing w_1 forces w_2 or w_3 to decrease since the sum of the weights is constant, hence, $E(H(p_1, o))$ decreases. Analogously, $E(H(p_2, o))$ and $E(H(p_3, o))$ are decreasing functions of their weights w_2 and w_3, respectively. □

In summary, in this section, we have introduced the weighted multiparent mask-based crossover and shown that it is a convex combination operator satisfying the four requirements of a metric convex combination for the Hamming space: convex weights (Definition 8), convexity (geometricity, Theorem 8), coherence (Theorem 9), and symmetry (self evident).

5. GEOMETRIC PSO FOR OTHER REPRESENTATIONS

Before looking into how we can extend GPSO to other solution representations, we will discuss the relation between the three-parental geometric crossover and the symmetry requirement for a convex combination.

For each of the spaces considered in Section 4, we have first considered (or defined) a three-parental recombination and then proven that it is a three-parental geometric crossover by showing that it can actually be decomposed into two sequential applications of a geometric crossover for the specific space.

However, we could have skipped the *explicit definition* of a three-parental recombination altogether. In fact, to obtain the three-parental recombination, we could have used two sequential applications of a known two-parental geometric crossover for the specific space. This composition is indeed a three-parental recombination (it combines three parents) and it is decomposable by construction. Hence, it is a three-parental geometric crossover. This, indeed, would have been simpler than the route we took.

The reason we preferred to define explicitly a three-parental recombination is that the requirement of symmetry of the convex combination is true by construction: if the roles of any two parents are swapped by exchanging in the three-parental recombination both positions and the respective recombination weights, the resulting recombination operator is equivalent to the original operator.

The symmetry requirement becomes harder to enforce and prove for a three-parental geometric crossover obtained by two sequential applications of a two-parental geometric crossover. We illustrate this in the following.

Let us consider three parents a, b, and c with positive weights w_a, w_b, and w_c which add up to one. If we have a symmetric three-parental weighted geometric crossover ΔGX, the symmetry of the recombination is guaranteed by the symmetry of the operator. So $\Delta GX((a, w_a), (b, w_b), (c, w_c))$ is equivalent to $\Delta GX((b, w_b), (a, w_a), (c, w_c))$. Hence, the requirement of symmetry on the weights of the convex combination holds. If we consider a three-parental recombination defined by using twice a two-parental genetic crossover GX, we have

$$\begin{aligned}\Delta GX((a, w_a), (b, w_b), (c, w_c)) \\ = GX((GX((a, w'_a), (b, w'_b)), w_{ab}), (c, w'_c))\end{aligned} \tag{9}$$

with the constraint that w'_a and w'_b are positive and add up to one, and w_{ab} and w'_c are positive and add up to one. Notice the inherent asymmetry in this expression: the weights w'_a and w'_b are not directly comparable with w'_c because they are relative weights between a and b. Moreover, there is the extra weight w_{ab}. This asymmetry makes the requirement of symmetry problematic to meet: given the desired w_a, w_b, and w_c, what values of w'_a, w'_b, w_{ab}, and w'_c should we choose to obtain an equivalent symmetric three-parental weighted recombination expressed as a sequence of two two-parental geometric crossovers?

For the Euclidean space, it is easy to answer this question using simple algebra as follows:

$$\begin{aligned}\Delta GX &= w_a \cdot a + w_b \cdot b + w_c \cdot c \\ &= (w_a + w_b)\left(\frac{w_a}{w_a + w_b} \cdot a + \frac{w_b}{w_a + w_b} \cdot b\right) + w_c \cdot c.\end{aligned} \tag{10}$$

Since the convex combination of two points in the Euclidean space is $GX((x, w_x), (y, w_y)) = w_x \cdot x + w_y \cdot y$, and $w_x, w_y > 0$ and $w_x + w_y = 1$, then

$$\begin{aligned}&\Delta GX((a, w_a), (b, w_b), (c, w_c)) \\ &= GX\left[\left(GX\left(\left(a, \frac{w_a}{w_a + w_b}\right), \left(b, \frac{w_b}{w_a + w_b}\right)\right), w_a + w_b\right), (c, w_c)\right].\end{aligned} \tag{11}$$

However, the question may be less straightforward to answer for other spaces, although, we could use the equation above as a rule-of-thumb to map the weights of ΔGX and the weights in the sequential GX decomposition to obtain a nearly symmetric convex combination.

Where does this discussion leave us in relation to the extension of GPSO to other representations? We have seen that there are two alternative ways to produce a convex combination for a new representation: (i) explicitly define a symmetric three-parental recombination for the new representation and then prove its geometricity by showing that it is decomposable into a sequence of two two-parental geometric crossovers (*explicit definition*), or (ii) use twice the simple geometric crossover to produce a symmetric or nearly symmetric three-parental recombination (*implicit definition*). The second option is also very interesting because *it allows us to extended automatically GPSO to all representations we have geometric crossovers for* (such as permutations, GP trees, and variable-length sequences, to mention but a few), *and virtually to any other complex solution representation.*

6. EXPERIMENTAL RESULTS FOR EUCLIDEAN, MANHATTAN, AND HAMMING SPACES

We have run two groups of experiments: one for the continuous version of the GPSO (`EuclideanPSO`, or EPSO for short, and `ManhattanPSO`, or MPSO), and one for the binary version (`HammingPSO`, or HPSO).

For the Euclidean and Manhattan versions, we have compared the performances with those of a continuous PSO (`BasePSO`, or BPSO) with constriction and inertia, whose parameters are as in Table 1. We have run the experiments for dimensions 2, 10, and 30 on the following five-benchmark functions: *F1C Sphere* [19], *F2C Rosenbrock* [19], *F3C Ackley* [20], *F4C Griewangk* [21], and *F5C Rastrigin* [22]. The Hamming version has been tested on the De Jong's test suite [19]: *F1 Sphere (30)*, *F2 Rosenbrock (24)*, *F3 Step (50)*, *F4 Quartic (240)*, and *F5 Shekel (34)*, where the numbers in brackets are the dimensions of the problems in bits. All functions in the test bed have global optimum 0 and are to be maximized.

Since there is no equivalent GPSO with $\phi_1 = \phi_2 = 2.05$ ($\phi_1 + \phi_2 > 1$, which does not respect the conditions in Theorem 3), we have decided to set w_1, w_2, and w_3 proportional to ω, ϕ_1, and ϕ_2, respectively, and summing up to one (see Table 2).

For the binary version, the parameters of population size, number of iterations, and w_1, w_2, and w_3 have been tuned on the sphere function and are as in Table 3. From the parameters tuning, it appears that there is a preference for values of w_1 close to zero. This means that there is a bias towards

Table 1: Parameters for BPSO.

Population size	20, 50 particles
Stop condition	200 iterations
$V_{\max} = X_{\max}$	Max_value-min_value
κ	0.729
ω	1.0
$\phi_1 = \phi_2$	2.05

Table 2: Parameters for EPSO and MPSO.

Population size	20, 50 particles
Stop condition	200 iterations
$V_{\max} = X_{\max}$	Max_value-min_value
Mutation	uniform in $[-0.5, 0.5]$
w_1	$\omega/(\omega + \phi_1 + \phi_2) = 1.0/5.10$
w_2	$\phi_1/(\omega + \phi_1 + \phi_2) = 2.05/5.10$
w_3	$\phi_2/(\omega + \phi_1 + \phi_2) = 2.05/5.10$

Table 3: Selected parameters for HPSO.

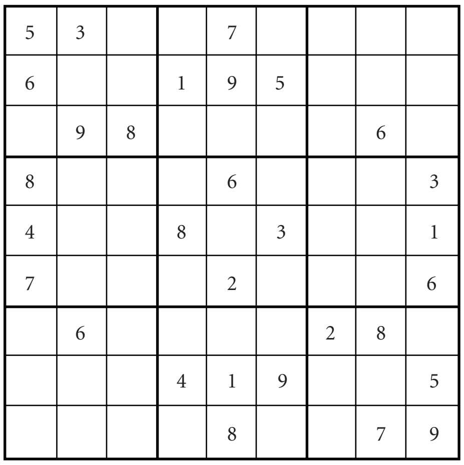
Population size	100 particles
Iterations	400
Bitwise mutation rate	$1/N$
$w_1 = 0$	$w_2 = w_3 = 1/2$
$w_1 = 1/6$	$w_2 = w_3 = 5/12$

the swarm and particle bests, and less attraction towards the current particle position.

For each set up, we performed 20 independent runs. Table 4 shows the best and the mean fitness value (i.e., the fitness value at the position where the population converges) found by the swarm when exploring continuous spaces. This table summarizes the results for the three algorithms presented, over the five test functions, for the two choices of population size, giving an immediate comparison of the performances. Overall the GPSOs (EPSO and MPSO), compare very favourably with BPSO, outperforming it in many cases. This is particularly interesting since it suggests that the inertia term (not present in GPSO) is not necessary for good performance.

In two dimensions, the results for all the functions (for PSOs both with 20 and 50 particles) are nearly identical, with BPSO and the two GPSOs both performing equally well. In higher dimensions, it is interesting to see how dimensionality does not seem to affect the quality of the results of GPSOs (while there is a fairly obvious decline in the performance of BPSO when dimension increases).

Also, EPSO's and MPSO's results are virtually identical. Let us recall from Section 4.2 the difference between Euclidean and Manhattan spaces, that is, "the difference between box recombination and linear recombination (Euclidean space) is that in the latter, the weights are randomly perturbed only once and the same weights are used for all the dimensions, whereas the former one has a different randomly perturbed version of the weights for each dimension." The results obtained show, therefore, that at least in this context,



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2: Example of Sudoku puzzle.

randomly perturbing the weights once for all dimensions, or perturbing them for each dimension, does not seem to affect the end result.

Table 5 shows the mean of the best fitness value and the best fitness value over the whole population for the binary version of the algorithm. The algorithm compares well with results reported in the literature, with HPSO obtaining near optimal results on all functions. Interestingly, the algorithm works at its best when w_1, the weight for x_i (the particle position), is zero. This corresponds to a degenerated PSO that makes decisions without considering the current position of the particle.

7. GEOMETRIC PSO FOR SUDOKU

In this section, we will put into practice the ideas discussed in Section 5 and propose a GPSO to solve the Sudoku puzzle. In Section 7.1, we introduce the Sudoku puzzle. In Section 7.2, we present a geometric crossover for Sudoku. In Section 7.3, we present a three-parental crossover for Sudoku and show that it is a convex combination.

7.1. *Description of Sudoku*

Sudoku is a logic-based placement puzzle. The aim of the puzzle is to enter a digit from 1 through 9 in each cell of a 9×9 grid made up of 3×3 subgrids (called "regions"), starting with various digits given in some cells (the "givens"). Each row, column, and region must contain only one instance of each digit. In Figure 2, we show an example of Sudoku puzzle. Sudoku puzzles with a unique solution are called proper Sudoku, and the majority of published grids are of this type.

Published puzzles often are ranked in terms of difficulty. Perhaps surprisingly, the number of "givens" has little or no bearing on a puzzle's difficulty. This is based on the relevance and the positioning of the numbers rather than the quantity of the numbers.

The 9×9 Sudoku puzzle of any difficulty can be solved very quickly by a computer. The simplest way is to use some brute force trial-and-error search employing back tracking.

Table 4: Test results for continuous versions: best and mean fitness values found by the swarm over 20 runs at last iteration (iteration 200).

		BPSO			EPSO			MPSO		
	Dim.	2	10	30	2	10	30	2	10	30
Popsize = 20	F1C best	−5.35e-14	−1.04	−59.45	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−6.54e-09	−20.75	−168.19	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	F2C best	0.00	−36.18	−1912.05	−0.71	−8.98	−28.97	−0.66	−8.96	−28.97
	mean	−97.91	−979.56	−8847.44	−1.0	−9.0	−29.0	−1.0	−9.0	−29.0
	F3C best	−3.06e-05	−8.05	−18.09	0.0	0.0	0.0	0.0	0.0	0.0
	mean	−0.00	−14.86	−20.49	0.0	0.0	0.0	0.0	0.0	0.0
	F4C best	−0.31	−1.10	−6.67	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	mean	−1.52	−2.98	−17.04	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	F5C best	−0.33	−58.78	−305.11	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−10.41	−160.98	−504.62	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
Popsize = 50	F1C best	−3.67e-13	−0.60	−53.93	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−1.11e-08	−19.09	−176.07	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	F2C best	0.00	−19.46	−1639.46	−0.57	−8.96	−28.96	−0.53	−8.95	−29.0
	mean	−56.04	−791.88	−9425.92	−1.0	−9.0	−29.0	−1.0	−9.0	−29.0
	F3C best	−1.81e-06	−6.78	−17.62	0.0	0.0	0.0	0.0	0.0	0.0
	mean	−0.00	−15.55	−20.43	0.0	0.0	0.0	0.0	0.0	0.0
	F4C best	−0.30	−1.05	−6.14	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	mean	−1.63	−2.79	−17.79	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	F5C best	−0.10	−53.67	−302.29	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−3.56	−159.76	−503.48	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0

Table 5: Test results for HPSO with selected parameters for De Jong's test suite.

	F1	F2	F3	F4	F5
$\overline{\omega} = 0.0$ best	−0.000 15	−0.000 34	−0.0	3.451 70	−1.131 83
mean	−5.515 40	−54.144 53	−2.594	−5.382 33	−142.678 53
$\overline{\omega} = \frac{1}{6}$ best	−0.000 125	−0.000 297	−0.0	3.273 980	−1.111 220
mean	−5.375 902	−85.170 099	−2.949	−6.919 343	−167.283 327

Constraint programming is a more efficient method that propagates the constraints successively to narrow down the solution space until a solution is found or until alternate values cannot otherwise be excluded, in which case, backtracking is applied. A highly efficient way of solving such constraint problems is the Dancing Links Algorithm by Donald Knuth [23].

The general problem of solving Sudoku puzzles on $n^2 \times n^2$ boards of $n \times n$ blocks is known to be NP complete [24]. This means that, unless $P =$ NP, the exact solution methods that solve very quickly the 9×9 boards take exponential time in the board size in the worst case. However, it is unknown whether the general Sudoku problem restricted to puzzles with unique solutions remains NP complete or becomes polynomial.

Solving Sudoku puzzles can be expressed as a graph coloring problem. The aim of the puzzle in its standard form is to construct a proper 9 coloring of a particular graph, given a partial 9 coloring.

A valid Sudoku solution grid is also a Latin square. Sudoku imposes the additional regional constraint. Latin square completion is known to be NP complete. A further relaxation of the problem allowing repetitions on columns (or rows) makes it polynomially solvable.

Admittedly, evolutionary algorithms and meta-heuristics in general are not the best techniques to solve Sudoku because they do not exploit systematically the problem constraints to narrow down the search. However, Sudoku is an interesting study case because it is a relatively simple problem but not trivial since is NP complete, and the different types of constraints make Sudoku an interesting playground for search operator design.

7.2. *Geometric crossover for Sudoku*

Sudoku is a constraint satisfaction problem with four types of constraints:

(1) fixed elements,
(2) rows are permutations,
(3) columns are permutations,
(4) and boxes are permutations.

It can be cast as an optimization problem by choosing some of the constraints as hard constraints that all solutions have to respect, and the remaining constraints as soft constraints that can be only partially fulfilled, and the level of fulfilment is the fitness of the solution. We consider a space with the following characteristics:

(i) *hard constraints*: fixed positions and permutations on rows,
(ii) *soft constraints*: permutations on columns and boxes,
(iii) *distance*: sum of swap distances between paired rows (row-swap distance),
(iv) *feasible geometric mutation*: swap two nonfixed elements in a row, and
(v) *feasible geometric crossover*: row-wise PMX and row-wise cycle crossover.

The chosen mutation preserves both fixed positions and permutations on rows (hard constraints) because swapping elements within a row, which is a permutation, returns a permutation. The mutation is 1-geometric under row-swap distance. Row-wise PMX and row-wise cycle crossover recombine parent grids applying, respectively, PMX and cycle crossover to each pair of corresponding rows. In case of PMX, the crossover points can be selected to be the same for all rows, or be random for each row. In terms of offspring that can be generated, the second version of row-wise PMX includes all the offspring of the first version.

Simple PMX and simple cycle crossover applied to parent permutations always return permutations. They also preserve fixed positions. This is because both are geometric under swap distance and, in order to generate offspring on a minimal sorting path between parents using swaps (sorting one parent into the order of the other parent), they have to avoid swaps that change common elements in both parents (elements that are already sorted). Therefore, also row-wise PMX and row-wise cycle crossover preserve both hard constraints.

Using the product geometric crossover Theorem 1, it is immediate to see that both row-wise PMX and row-wise cycle crossover are geometric under row-swap distance since simple PMX and simple cycle crossover are geometric under swap distance. Since simple cycle crossover is also geometric under Hamming distance (restricted to permutations), row-wise cycle crossover is also geometric under Hamming distance.

To restrict the search to the space of grids with fixed positions and permutations on rows, the initial population must be seeded with feasible random solutions taken from this space. Generating such solutions can be done still very efficiently.

The fitness function (to maximize) is the sum of the number of unique elements in each row plus the sum of the number of unique elements in each column plus the sum of the number of unique elements in each box. So for a 9×9 grid, we have a maximum fitness of $9\cdot 9 + 9\cdot 9 + 9\cdot 9 = 243$ for a completely correct Sudoku grid, and a minimum fitness little more than $9\cdot 1 + 9\cdot 1 + 9\cdot 1 = 27$ because for each row, column, and square, there is at least one unique element type.

```
Mask:1 2 2 3 1 3 2
  p1: 1 2 3 4 5 6 7
  p2: 3 5 1 4 2 7 6
  p3: 3 2 1 4 5 7 6
-------------------
   o: 1 5 3 4 2 7 6
```

Figure 3: Example of multiparental sorting crossover.

It is possible to show that the fitness landscapes associated with this space is smooth, making the search operators proposed a good choice for Sudoku.

7.3. Convex combination for Sudoku

In this section, we first define a multiparental recombination for permutations and then prove that it respects the four requirements for being a convex combination presented in Section 3.3.

Let us consider the example in Figure 3 to illustrate how the *multiparental sorting crossover* works.

The mask is generated at random and is a vector of the same length of the parents. The number of 1's, 2's, and 3's in the mask is proportional to the recombination weights w_1, w_2, and w_3 of the parents. Every entry of the mask indicates to which parent the other two parents need to be equal to for that specific position. In a parent, the content of a position is changed by swapping it with the content of another position in the parent. The recombination proceeds as follows. The mask is scanned from the left to the right. In position 1, the mask has a 1. This means that at position 1, parent p_2 and parent p_3 have to become equal to parent p_1. This is done by swapping the elements 1 and 3 in parent p_2 and the elements 1 and 3 in parent p_3. The recombination now continues on the updated parents: parent p_1 is left unchanged and the current parent p_2 and parent p_3 are the original parents p_2 and p_3 after the swap. At position 2, the mask has 2. This means that at position 2, the current parent p_1 and current parent p_3 have to become equal to the current parent p_2. So at position 2, parent p_1 and parent p_3 have to get 5. To achieve this, in parent p_1, we need to swap elements 2 and 5, and in parent p_3, we need to swap elements 2 and 5. The recombination continues on the updated parents for position 3 and so on, up to the last position in the mask. At this point, the three parents are now equal because at each position, one element of the permutation has been fixed in that position and it is automatically not involved in any further swap. Therefore, after all positions have been considered, all elements are fixed. The permutation to which the three parents converged is the offspring permutation. This recombination sorts by swaps the three parents towards each other according to the contents of the crossover mask, and the offspring is the result of this multiple sorting. This recombination can be easily generalized to any number of parents.

Theorem 10 (geometricity of three-parental sorting crossover). *Three-parental sorting crossover is a geometric crossover under swap distance.*

Proof. A three-parental sorting crossover with recombination mask m_{123} is equivalent to a sequence of two two-parental sorting crossovers: the first is between parents p_1 and p_2 with recombination mask m_{12} obtained by substituting all 3's with 2's in m_{123}. The offspring p_{12} so obtained is recombined with p_3 with recombination mask m_{23} obtained by substituting all 1's with 2's in m_{123}. So for Theorem 2, the three-parental sorting crossover is geometric. □

Theorem 11 (coherence between weights and distances). *In a weighted multiparent sorting crossover, the swap distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof. The weights associated to the parents are proportional to their frequencies in the recombination mask. The more occurrences of a parent in the recombination mask, the smaller the swap distance between this parent and the offspring. This is because the mask tells the parent to copy at each position. So the higher the weight of a parent, the smaller its distance to the offspring. □

The weighted multiparental sorting crossover is a convex combination operator satisfying the four requirements of a metric convex combination for the swap space: convex weights sum to 1 by definition, convexity (geometricity, Theorem 10), coherence (Theorem 11), and symmetry (self evident).

The solution representation for Sudoku is a vector of permutations. For the product geometric crossover theorem, the compound crossover over the vector of permutations that applies a geometric crossover to each permutation in the vector is a geometric crossover. Theorem 12 extends to the case of a multiparent geometric crossover.

Theorem 12 (product geometric crossover for convex combinations). *A convex combination operator, applied to each entry of a vector, results in a convex combination operator for the entire vector.*

Proof. The product geometric crossover theorem (Theorem 1) is true because the segment of a product space is the Cartesian product of the segments of its projections. A segment is the convex hull of two points (parents). More generally, it holds that the convex hull (of any number of points) of a product space is the Cartesian product of the convex hulls of its projections [17]. The product geometric crossover then naturally generalizes to the multiparent case. □

8. EXPERIMENTAL RESULTS FOR SUDOKU

In order to test the efficacy of the GPSO algorithm on the Sudoku problem, we ran several experiments in order to thoroughly explore the parameter space and variations of the algorithm. The algorithm in itself is a straightforward implementation of the GPSO algorithm given in Section 3.4 with the search operators for Sudoku presented in Section 7.3.

The parameters we varied were swarm sociality (w_2) and memory (w_3), each of which were in turn set to 0, 0.2, 0.4, 0.6, 0.8, and 1.0. Since the attraction to each particle's position is defined as $w_1 = 1 - w_2 - w_3$, the space of this parameter was implicitly explored. Likewise, mutation probability was set to either 0, 0.3, 0.7, or 1.0. The swarm size was set to be either 20, 100, or 500 particles, but the number of updates was set so that each run of the algorithm resulted in exactly 100 000 fitness evaluations (thus performing 5000, 1000, or 200 updates). Further, each combination was tried with ring topology, von Neumann topology (or lattice topology), and global topology, which are the most common topologies.

TABLE 6: Average of bests of 50 runs with population size 100, lattice topology, and mutation 0.0, varying sociality (vertical) and memory (horizontal).

	Memory					
Sociality	0.0	0.2	0.4	0.6	0.8	1.0
1.0	208	—	—	—	—	—
0.8	227	229	—	—	—	—
0.6	230	233	235	—	—	—
0.4	231	236	237	240	—	—
0.2	232	239	241	**242**	**242**	—
0.0	207	207	207	207	207	207

TABLE 7: Average of bests of 50 runs with population size 100, lattice topology, and mutation 0.3, varying sociality (vertical) and memory (horizontal).

	Memory					
Sociality	0.0	0.2	0.4	0.6	0.8	1.0
1.0	238	—	—	—	—	—
0.8	238	237	—	—	—	—
0.6	239	239	240	—	—	—
0.4	240	240	241	241	—	—
0.2	240	241	**242**	**242**	**242**	—
0.0	213	231	232	233	233	233

As explained in Section 5, there are two alternative ways of producing a convex combination: either using a convex combination operator or simply applying twice a two-parental weighted recombination with appropriate weights to obtain the convex combination. Both ways to produce convex combination operators, explicit and implicit, were tried on preliminary runs and turned out to produce indistinguishable results. In the end, we used the convex combination operator.

8.1. *Effects of varying coefficients*

The best population size is 100. The other two sizes we studied (20 and 500) were considerably worse. The best topology is the lattice (von Neumann) topology. The other two topologies we studied were worse (see Table 9).

From Tables 6–8, we can see that mutation rates of 0.3 and 0.7 perform better than no mutation at all. We can also see that parameter settings with w_1 (i.e., the attraction of

Table 8: Average of bests of 50 runs with population size 100, lattice topology and mutation 0.7, varying sociality (vertical) and memory (horizontal).

	Memory					
Sociality	0.0	0.2	0.4	0.6	0.8	1.0
1.0	232	—	—	—	—	—
0.8	232	240	—	—	—	—
0.6	228	241	241	—	—	—
0.4	224	**242**	**242**	**242**	—	—
0.2	219	234	**242**	**242**	**242**	—
0.0	215	226	233	233	236	236

Table 9: Success rate of various methods.

Method	Success
GA	50/50
Hillclimber	35/50
GPSO-global	7/50
GPSO-ring	20/50
GPSO-von Neumann	36/50

the particle's previous position) set to more than 0.4 generally perform badly. The best configurations generally have w_2 (i.e., sociality) set to 0.2 or 0.4, w_3 (i.e., memory) set to 0.4 or 0.6, and w_1 set to 0 or 0.2. This gives us some indication of the importance of the various types of recombinations in GPSO as applied at least to this particular problem. Surprisingly, the algorithm works at its best when the weight of the particle position (w_1) is zero or nearly zero. In the case of w_1 set to 0, GPSO, in fact, degenerates to a type of evolutionary algorithm with deterministic uniform selection, mating with the population best with local replacement between parents and offspring.

8.2. *PSO versus EA*

Table 9 compares the success rate of the best configurations of various methods we have tried. Success is here defined as the number of runs (out of 50) where the global optimum (243) is reached. All the methods were allotted the same number of function evaluations per run.

From the table, we can see that the von Neumann topology clearly outperforms the other topologies we tested, and that a GPSO with this topology can achieve a respectable success rate on this tricky noncontinuous problem. However, the best genetic algorithm still significantly outperforms the best GPSO we have found so far. (Notice that this is true only when considering GPSO as an optimizer. The approximation behavior of the GPSO is very good: with the right parameter setting, the GPSO reaches on average a fitness of 242 out of 243 (see Tables 6–8).) We believe this to be at least partly the effect of the even more extensive tuning of parameters and operators undertaken in our GA experiments.

9. CONCLUSIONS AND FUTURE WORK

We have extended the geometric framework with the notion of multiparent geometric crossover, that is, a natural generalization of two-parental geometric crossover: offspring are in the convex hull of the parents. Then, using the geometric framework, we have shown an intimate relation between a simplified form of PSO (without the inertia term) and evolutionary algorithms. This has enabled us to generalize, in a natural, rigorous, and automatic way, PSO for any type of search space for which a geometric crossover is known.

We specialized the general PSO to Euclidean, Manhattan, and Hamming spaces, obtaining three instances of the general PSO for these spaces, EPSO, MPSO, and HPSO, respectively. We have performed extensive experiments with these new GPSOs. In particular, we applied EPSO, MPSO, and HPSO to standard sets of benchmark functions and obtained a few surprising results. Firstly, the GPSOs have performed really well, beating the canonical PSO with standard parameters most of the time. Secondly, they have done so right out of the box. That is, unlike the early versions of PSO which required considerable effort before a good general set of parameters could be found, with GPSO, we have done very limited preliminary testing and parameter tuning, and yet the new PSOs have worked well. This suggests that they may be quite robust optimisers. This will need to be verified in future research. Thirdly, HPSO works at its best with only weak attraction toward the current position of the particle. With this configuration, GPSO almost degenerates to a type of genetic algorithm.

An important feature of the GPSO algorithm is that it allows one to automatically define PSOs for all spaces for which a geometric crossover is known. Since geometric crossovers are defined for all of the most frequently used representations and many variations and combinations of those, our geometric framework makes it possible to derive PSOs for all such representations. GPSO is rigorous generalization of the classical PSO to general metric spaces. In particular, it applies to combinatorial spaces.

We have demonstrated how simple it is to specify the general GPSO algorithm to the space of Sudoku grids (vectors of permutations), using both an explicit and an implicit definitions of convex combination. We have tested the new GPSO on Sudoku and have found that (i) the communication topology makes a huge difference and that the lattice topology is by far the best; (ii) as for HPSO, the GPSO on Sudoku works better with weak attraction toward the current position of the particle; (iii) the GSPO on Sudoku finds easily near-optimal solutions but it does not always find the optimum. Admittedly, GPSO is not the best algorithm for the Sudoku puzzle where the aim is to obtain the correct solution all the times, not a nearly correct one. This suggests that GPSO would be much more profitably applied to combinatorial problems for which one would be happy to find near-optimal solutions quickly.

In summary, we presented a PSO algorithm that has been quite successfully applied to a nontrivial combinatorial space. This shows that GPSO is indeed a natural and promising generalization of classical PSO. In future work, we will consider GPSO for even more challenging combinatorial spaces such as the space of genetic programs. Also, since the inertia term is a very important feature of classical PSO, we want to generalize it and test the GPSO with the inertia term on combinatorial spaces.

ACKNOWLEDGMENT

The second and fourth authors would like to thank EPSRC, Grant no. GR/T11234/01 "Extended Particle Swarms," for the financial support.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[2] M. Clerc, "Discrete particle swarm optimization, Illustrated by the traveling salesman problem," in *New Optimization Techniques in Engineering*, Springer, New York, NY, USA, 2004.

[3] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (ICSMC '97)*, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.

[4] A. Moraglio and R. Poli, "Topological interpretation of crossover," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '04)*, vol. 3102, pp. 1377–1388, Seattle, Wash, USA, June 2004.

[5] A. Moraglio and R. Poli, "Topological crossover for the permutation representation," in *Proceedings of the Workshops on Genetic and Evolutionary Computation (GECCO '05)*, pp. 332–338, Washington, DC, USA, June 2005.

[6] A. Moraglio and R. Poli, "Geometric landscape of homologous crossover for syntactic trees," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 427–434, Edinburgh, UK, September 2005.

[7] A. Moraglio, R. Poli, and R. Seehuus, "Geometric crossover for biological sequences," in *Proceedings of the 9th European Conference on Genetic Programming*, pp. 121–132, Budapest, Hungary, April 2006.

[8] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[9] A. Moraglio, J. Togelius, and S. Lucas, "Product geometric crossover for the Sudoku puzzle," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 470–476, Vancouver, BC, Canada, July 2006.

[10] T. Jones, *Evolutionary algorithms, fitness landscapes and search*, Ph.D. thesis, University of New Mexico, Albuquerque, NM, USA, 1995.

[11] M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, Springer, New York, NY, USA, 1991.

[12] A. Moraglio and R. Poli, "Product geometric crossover," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06)*, pp. 1018–1027, Reykjavik, Iceland, September 2006.

[13] A. Moraglio and R. Poli, "Geometric crossover for sets, multisets and partitions," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06)*, pp. 1038–1047, Reykjavik, Iceland, September 2006.

[14] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[15] M. Clerc, "When nearer is better," Hal Open Archive, 2007.

[16] P. M. Pardalos and M. G. C. Resende, *Handbook of Applied Optimization*, Oxford University Press, Oxford, UK, 2002.

[17] M. L. J. van de Vel, *Theory of Convex Structures*, North-Holland, Amsterdam, The Netherlands, 1993.

[18] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[19] K. De Jong, *An analysis of the behaviour of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, Ann Arbor, Mich, USA, 1975.

[20] T. Bäck, *Evolutinary Algorithms in Theory and in Practice*, Oxford University Press, Oxford, UK, 1996.

[21] A. Törn and A. Zilinskas, *Global Optimization*, vol. 350 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 1989.

[22] H. Müehlenbein, M. Schomisch, and J. Born, "The Parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, pp. 619–632, 1991.

[23] D. E. Knuth, "Dancing links," Preprint P159, Stanford University, 2000.

[24] T. Yato and T. Seta, "Complexity and completeness of finding another solution and its application to puzzles," Preprint, University of Tokyo, 2005.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 587309, 16 pages
doi:10.1155/2008/587309

Research Article

Forma Analysis of Particle Swarm Optimisation for Permutation Problems

Tao Gong and Andrew L. Tuson

Department of Computing, City University, London EC1V 0HB, UK

Correspondence should be addressed to Tao Gong, t.gong@soi.city.ac.uk

Received 20 July 2007; Revised 20 January 2008; Accepted 1 April 2008

Recommended by Riccardo Poli

Particle swarm optimisation (PSO) is an innovative and competitive optimisation technique for numerical optimisation with real-parameter representation. In this paper, we examine the working mechanism of PSO in a principled manner with forma analysis and investigate the applicability of PSO on the permutation problem domain. Particularly, our derived PSO schemes are empirically studied based on the quadratic assignment problem (QAP) benchmarks to justify its comparable performance, which in turn implies the benefits of our approach in applying PSO to the discrete problem domain.

1. INTRODUCTION

PSO was originally designed as a numerical optimisation technique based on swarm intelligence. In the literature, there are a few attempts to exploit its usage in the discrete problem domain [1–4], which are mostly performed using a binary encoding. However, research on the transformation of the working mechanism of PSO to the permutation problem domain, where the representations are highly constrained, has been relatively limited [3, 5, 6]. This limitation is mainly caused by the lack of a principled generalisation of PSO to guide its adaptation to discrete combinatorial problems [3].

In this paper, we aim to design PSO operators for permutation problems without losing the underlining principles of the original PSO. A PSO operator template will be formally defined with *forma analysis* in form of equivalence relations. Following the introduction of formal descriptions of permutations, concrete PSO operators are then formally derived based on the PSO operator template. Empirical study of the derived PSO schemes is also carried out based on the QAP benchmarks.

2. PARTICLE SWARM OPTIMISATION

PSO was initially introduced by Kennedy and Eberhart [7] as a function optimisation technique inspired by the behaviors of fish schooling and bird flocking. The PSO approach simulates the social behavior of particles/agents moving in a multidimensional search space, while each particle has its position and velocity. Each particle is treated as a potential solution to the optimisation problem. Usually, the position is represented as a vector:

$$\mathbf{X}_{it} = (x_{it}(1), x_{it}(2), \ldots, x_{it}(D)), \tag{1}$$

while the velocity is represented as another vector:

$$\mathbf{V}_{it} = (v_{it}(1), v_{it}(2), \ldots, v_{it}(D)), \tag{2}$$

where i represents the index of the particle, t is the time step, and D is the dimensionality of the search space.

For each generation, the particle compares its current position with the goal (global best/personal best) position, adjusting its velocity accordingly towards the goal with the help of the explicit memory of the best position ever found both globally and individually.

The most popular formulation of how particle adjusts its velocity and position [8, 9] is shown in

$$\begin{aligned} v_{i(t+1)}(d) = w * v_{it}(d) + c_1 r_1 [\mathrm{Pb}_i(d) - x_{it}(d)] \\ + c_2 r_2 [\mathrm{Gb}(d) - x_{it}(d)], \end{aligned} \tag{3}$$

$$x_{i(t+1)}(d) = x_{it}(d) + v_{i(t+1)}(d). \tag{4}$$

Alternatively, we may have

$$x_{i(t+1)}(d) = x_{it}(d) + wv_{it}(d) + c_1r_1[\mathrm{Pb}_i(d) - x_{it}(d)] + c_2r_2[\mathrm{Gb}(d) - x_{it}(d)]. \quad (5)$$

In the above formulation, d is the index of dimension in the search space, w represents the inertia weight of the "flying" dynamic, which describes the degree of maintaining its previous velocity vector against the attraction point. c_1 and c_2 are regarded as *cognitive* and *social* parameters for the algorithm, respectively, while r_1 and r_2 are random numbers within the interval $[0, 1]$. Pb_i is the personal best position which is recorded by particle i, while Gb is the global best position obtained by any particle in the population.

Adapting standard PSO to permutation problems has been a rather interesting task, as researchers are curious about its performance in the discrete domain. In this paper, we suggest that forma analysis gives a possible solution to achieve such task in a principled manner.

3. FORMA ANALYSIS

Forma analysis [10] is a formal but practical method that allows the problem representation and its operators to be structured in a formal manner by using *equivalence relations*. Each equivalence relation Ψ divides the search space into disjoint equivalence classes Ξ_ψ (depending on which value the solutions match), with individual equivalence classes being denoted by ξ, which gathers solutions that are equivalent under a certain equivalence relation.

The initial aim of forma analysis [10] was to codify knowledge of the problem domain using a set of *equivalence classes* (or *formae*) which is assumed to be able to cluster solutions with related performance in order to guide the search process more effectively, for example, edges if we are considering the travelling salesman problem. Since equivalence relations/classes have the ability to capture the properties of solutions, concrete operators can thus be mathematically derived with regards to how these equivalence relations are formally manipulated by the operator templates.

Figure 1 illustrates briefly the approach we adopt here in this paper, which can be explained as follows. *Given an operator template, any suitable description of the considered problem domain gives rise to a concrete operator, whose behavior and performance are related to the assumption adopted to describe the search space*. This approach effectively allows PSO to be adapted to arbitrary problem domain with the underlying working mechanism retained. Taking a step further, this approach is applicable across different problem domains and different optimisation techniques.

Some of the characteristics and operator templates related to forma analysis [10–12] are given below to facilitate our generalisation of PSO.

3.1. *Describing the search space*

The key concept is that of a basis: a set of equivalence relations that allows us to describe the search space S.

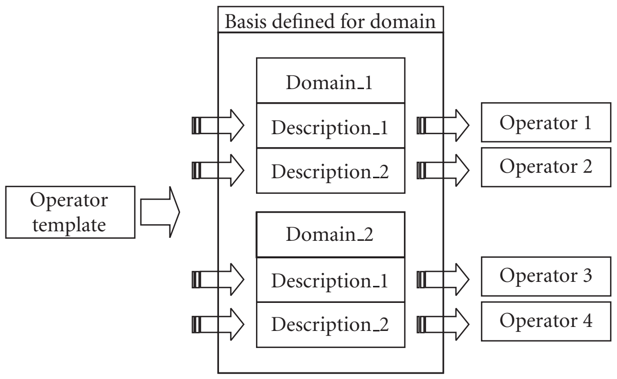


FIGURE 1: Illustration of the methodology based on forma analysis.

Definition 1. A subset Ψ of a set of equivalence relations is termed as a *basis* for the set of equivalence relations, if Ψ spans the set and Ψ is independent.

An encoding can thus be derived by taking the image of the basis equivalence classes corresponding to a particular solution in the search space.

3.2. *Domain independent operator templates*

Forma analysis can derive operators that explicitly manipulate the given equivalence relations. This is achieved by combining the basis with domain independent operators for specifying operator behavior in terms of basis.

One such (domain independent) operator template, which is related to the work presented in this paper, corresponds to the (*strict*) k-change operator template [12], formally as

$$O_k(x, k, \Psi) = \{y \in S \mid \mathrm{dis}_\Psi(x, y) = k\}, \quad (6)$$

where x and y represent the solution to be perturbed and the produced child, respectively, while $\mathrm{dis}_\Psi(x, y)$ represents the forma distance [13] between x and y under basis Ψ. This operator template can be interpreted as changing the values for k equivalence relations in x produces solution y.

The other operator template, random transmitting recombination (RTR) [10], is defined to select a child solution z out of the *dynastic potential* of the parent solutions x and y. $\mathrm{RTR}(x, y, \Psi)$ can be formally defined as

$$\mathrm{RTR}(x, y, \Psi) = \{z \in S \mid \forall \psi \in \Psi : \psi(x, z) = 1 \vee \psi(y, z) = 1\}, \quad (7)$$

where the actual child solution z is chosen from the set above uniformly at random. The underlying idea of RTR is to randomly generate child solution using parental equivalence classes (genetic material).

3.3. *Applicability*

Although the above concepts of forma analysis are developed under genetic algorithms, it has been shown that the forma analysis methodology itself is generalisable to other evolutionary optimisers based on (theoretically) any problem domain from the knowledge-based system (KBS) design standpoint [12]. Also, the underlying idea proposed by Surry [11] on defining formal representation in order to derive domain specific operators has also been justified to be rather successful. Furthermore, our previous work on adapting PSO to the binary search space following the forma analysis approach has provided some preliminary positive results and expectations. In summary, the proposed forma analysis-based operator design approach is applicable in the context of PSO and permutation problem domain, considering the previous theoretical and practical evidence.

4. FORMAL DESCRIPTIONS OF PERMUTATION

As previously studied in [12, 14], there are mainly three different representations for permutation that we can adopt to describe the permutation problems:

(1) *position*-based representation, which decides the absolute position of an item (e.g., item 5 is at position 4),

(2) *precedence*-based representation, which decides whether one task is performed before another (e.g., task 5 appears before task 4),

(3) *adjacency*-based representation, which decides whether two items are next to each other (e.g., item 4 is next to item 5).

In the following sections, formal descriptions for permutation with a set of n elements $N = \{e_1, \dots, e_n\}$ will be reviewed in form of equivalence relations, followed by some formal definitions of their induced constraints and their distance measurements [12, 15].

4.1. Position-based description of permutation

For position-based description, each position in the permutation i $(i = 1, \dots, n)$ is defined as an equivalence relation $\psi_{\text{pos}(i)}$ to form the basis of *position equivalence relations* such that

$$\Psi_{\text{pos}} = \{\psi_{\text{pos}(i)} \mid i \in N\}. \quad (8)$$

The equivalence classes (formae) for each position i are the set of n elements:

$$\Xi_{\psi_i} = \{\xi^{e_1}_{\text{pos}(i)}, \dots, \xi^{e_n}_{\text{pos}(i)}\}. \quad (9)$$

As an example, the permutation $(1, 4, 3, 2)$ can be described by the set of equivalence classes

$$\{\xi^1_{\text{pos}(1)}, \xi^4_{\text{pos}(2)}, \xi^3_{\text{pos}(3)}, \xi^2_{\text{pos}(4)}\}. \quad (10)$$

In addition, an induced feasibility constraint for this description C_{pos} needs to be added to ensure that different elements do not occupy the same position, and no two different positions can take the same element, formally as follows.

Definition 2. Given any two equivalence relations $\psi_{\text{pos}(i)}$ and $\psi_{\text{pos}(j)}$ for a permutation, the *position-based constraint* C_{pos} can be defined as

$$\forall i, j\, (i \neq j) : \psi_{\text{pos}(i)} \neq \psi_{\text{pos}(j)}. \quad (11)$$

A direct implication of this constraint is that the 1-change neighborhood structure should be prevented as this would involve placing two elements in the same position.

The distance metric for this formal description is simply the number of positions in the permutation that have different elements (i.e., the *hamming distance*) according to the definition of *forma distance* [13]. For example, the distance between $(1, 4, 3, 2)$ and $(1, 4, 2, 3)$ is 2, since they are different in two positions.

4.2. Precedence-based description of permutation

For precedence-based description, a set of basis precedence equivalence relations Ψ_{prec} between any two different elements in the permutation will be considered formally as

$$\Psi_{\text{prec}} = \{\psi_{\text{prec}(e_i, e_j)} \mid e_i, e_j \in N \wedge e_i \neq e_j\}. \quad (12)$$

However, by considering the fact (constraint) that $\psi_{\text{prec}(e_i, e_j)}$ and $\psi_{\text{prec}(e_j, e_i)}$ are reverse relations:

$$\forall e_i, e_j \in N\, (e_i \neq e_j) : \psi_{\text{prec}(e_i, e_j)} \Longleftrightarrow \neg\psi_{\text{prec}(e_j, e_i)}, \quad (13)$$

we can remove unnecessary relations by enforcing a sequence (e.g., $e_1 < e_2 < \cdots < e_n$) in the definition of the relation, such that

$$\Psi_{\text{prec}} = \{\psi_{\text{prec}(e_i, e_j)} \mid e_i, e_j \in N \wedge e_i < e_j\}. \quad (14)$$

Obviously, the equivalence classes are simply true/false for whether element e_i precedes element e_j in the permutation formally as

$$\Xi_{\psi_{\text{prec}(e_i, e_j)}} = \{\xi^0_{\text{prec}(e_i, e_j)}, \xi^1_{\text{prec}(e_i, e_j)}\}. \quad (15)$$

In addition, the feasibility constraint C_{prec} needs to be added that a valid permutation exists if and only if the relationship between the precedences is consistent (in that the transitivity condition is preserved), as shown below.

Definition 3. Given any two equivalence relations, $\psi_{\text{prec}(e_i, e_j)}$ and $\psi_{\text{prec}(e_j, e_k)}$, for a permutation, the *precedence-based constraint* C_{prec} can be defined as

$$\begin{aligned} \forall e_i, e_j, e_k \in N\, (e_i \neq e_j \neq e_k) &: \psi_{\text{prec}(e_i, e_j)} \wedge \psi_{\text{prec}(e_j, e_k)} \\ &\Longrightarrow \psi_{\text{prec}(e_i, e_k)}. \end{aligned} \quad (16)$$

The distance metric can be specified as the number of different precedence relations between two solutions. For example, the distance between permutation $(1, 2, 3, 4)$ and $(1, 4, 2, 3)$ can be obtained by comparing the following two sets:

$$\begin{aligned} &\{\xi^1_{\text{prec}(1,2)}, \xi^1_{\text{prec}(1,3)}, \dots, \underline{\xi^1_{\text{prec}(2,4)}}, \underline{\xi^1_{\text{prec}(3,4)}}\}, \\ &\{\xi^1_{\text{prec}(1,2)}, \xi^1_{\text{prec}(1,3)}, \dots, \underline{\xi^0_{\text{prec}(2,4)}}, \underline{\xi^0_{\text{prec}(3,4)}}\}. \end{aligned} \quad (17)$$

In practice, *bubble sort* will be sufficient to calculate the distance from one permutation towards another. This can be achieved by sorting one permutation towards another, when setting one as the initial permutation and the other as the goal permutation which represents the right "order." If we are looking at the previous example with two permutations, to calculate the distance from $(1,2,3,4)$ (i.e., the initial permutation) to $(1,4,2,3)$ (i.e., the goal permutation), the "priority" information can be obtained from $(1,4,2,3)$:

$$\text{order}(1)=1,\ \text{order}(4)=2,\ \text{order}(2)=3,\ \text{order}(3)=4. \tag{18}$$

Thus, the initial permutation can be represented (or more closely, implemented) in terms of $(e_i, \text{order}(e_i))$:

$$((1,1),(2,3),(3,4),(4,2)), \tag{19}$$

and it should be sorted according to the values of order(e_i) in order to "transform" to the goal permutation with a certain number of basic swap adjacency mutations.

4.3. Adjacency-based description of permutation

For adjacency-based description, a set of basis adjacency equivalence relations Ψ_{adj} is considered for any two elements to decide whether they are adjacent formally as

$$\Psi_{\text{adj}} = \{\psi_{\text{adj}(e_i,e_j)} \mid e_i, e_j \in N \wedge e_i \neq e_j\}. \tag{20}$$

Due to the fact that $\psi_{\text{adj}(e_i,e_j)}$ and $\psi_{\text{adj}(e_j,e_i)}$ are equivalent relations for *symmetric* problems (Surry presented an extensive study about *directed edge* representation for permutation in his work [11]):

$$\forall e_i, e_j \in N(e_i \neq e_j) : \psi_{\text{adj}(e_i,e_j)} \Longleftrightarrow \psi_{\text{adj}(e_j,e_i)}, \tag{21}$$

we can remove redundant relations by enforcing a sequence (e.g., $e_1 < e_2 < \cdots < e_n$) in the definition of the relation, such that

$$\Psi_{\text{adj}} = \{\psi_{\text{adj}(e_i,e_j)} \mid e_i, e_j \in N \wedge e_i < e_j\}. \tag{22}$$

As undirected edges are considered for adjacency-based description, the equivalence classes are simply true/false for whether there exists an edge between element e_i and element e_j formally as

$$\Xi_{\psi_{\text{adj}(e_i,e_j)}} = \{\xi^0_{\text{adj}(e_i,e_j)}, \xi^1_{\text{adj}(e_i,e_j)}\}. \tag{23}$$

In this case, $\xi^1_{\text{adj}(e_i,e_j)}$ represents a *positive edge* so that edge (e_i,e_j) must exists in the solution (e.g., a tour), while $\xi^0_{\text{adj}(e_i,e_j)}$ stands for a *negative edge* so that edge (e_i,e_j) cannot be included into the solution.

In addition, the feasibility constraint C_{adj} needs to be added so that each vertex of the undirected graph corresponding to the permutation can only participate in two edges and still be a valid permutation as follows.

Definition 4. Given an equivalence relation $\psi_{\text{adj}(e_i,e_j)}$ for a permutation, the *adjacency-based constraint* C_{adj} can be defined as

$$\forall e_i, e_j, e_k, e_l \in N : \psi_{\text{adj}(e_i,e_j)} \wedge \psi_{\text{adj}(e_i,e_k)} \Longrightarrow \neg\psi_{\text{adj}(e_i,e_l)}. \tag{24}$$

The distance between any two solutions in the search space under adjacency basis is thus calculated as the number of different edges that they possess. For instance, the distance between permutation $(1,2,3,4)$ and $(1,2,4,3)$ can be obtained by calculating the number of different adjacency relations between the following two sets:

$$\begin{gathered}\left\{\xi^1_{\text{adj}(1,2)}, \underline{\xi^0_{\text{adj}(1,3)}}, \underline{\xi^1_{\text{adj}(1,4)}}, \underline{\xi^1_{\text{adj}(2,3)}}, \underline{\xi^0_{\text{adj}(2,4)}}, \xi^1_{\text{adj}(3,4)}\right\},\\ \left\{\xi^1_{\text{adj}(1,2)}, \underline{\xi^1_{\text{adj}(1,3)}}, \underline{\xi^0_{\text{adj}(1,4)}}, \underline{\xi^0_{\text{adj}(2,3)}}, \underline{\xi^1_{\text{adj}(2,4)}}, \xi^1_{\text{adj}(3,4)}\right\}.\end{gathered} \tag{25}$$

However, on the "phenotypical" level this *forma distance* reduces to the number of different *positive* edges (n minus common *positive* edges), which should be 2 in this case. This is mainly because *negative* edges do not directly affect the quality of solution, although they implicitly affect the selection of positive edges through the feasibility constraints.

5. PSO OPERATOR TEMPLATE—A GENERALISATION

The generalisation of PSO is not as straightforward as some other optimisation techniques, such as the generalisation of Differential Evolution [16]. This is mainly because, besides the generalisation of *position* in a multi-dimensional space, we also need to generalise the *velocity* of particle in a multi-dimensional space.

By observing the update equation (as shown in (4)), we can easily find out that

$$v_{i(t+1)}(d) = x_{i(t+1)}(d) - x_{it}(d), \tag{26}$$

in which case the velocity is effectively the step size to perturb one solution towards another (in other words, the distance between the two solutions), which is jointly decided by the inertia component $w * v_{it}(d)$, the bias towards its personal best $c_1r_1[\text{Pb}_i(d) - x_{it}(d)]$, and the bias towards the global best $c_2r_2[\text{Gb}(d) - x_{it}(d)]$.

5.1. PSO operator template

By revealing the fact that velocity is the distance between the previous and current positions of the particle, we can define the operator template (under the basis Ψ) as follows.

Definition 5. Given a current position X_t (if one temporarily ignores the index i of this particle in the population), its position for the next time step X_{t+1} can be produced according to (27):

$$\begin{aligned}\{X_{t+1} \in S \mid\ & \text{dis}_\Psi(X_{t+1}, X_t)\\ &= w * \text{dis}_\Psi(X_t, X_{t-1}) \oplus c_1r_1 * \text{dis}_\Psi(\text{Pb}, X_t)\\ &\quad \oplus c_2r_2 * \text{dis}_\Psi(\text{Gb}, X_t)\},\end{aligned} \tag{27}$$

where $\text{dis}_\Psi(X_t, X_{t-1})$ is used to formalise the previous velocity of the particle, while $\text{dis}_\Psi(X_{t+1}, X_t)$ represents the current velocity for perturbing the current solution X_t.

5.2. *Stochastic interpretation of accumulation*

In the context of real-vectors, the accumulation of distances is straightforward to understand. However, for permutation problems the consideration of "directionality" becomes rather complex from the practical standpoint. By taking into account the fact that forma distance includes domain-specific distance magnitude and direction which cause certain difficulties in the context of permutation problems, a reasonable interpretation of the PSO operator template is required to facilitate the derivation of suitable PSO operators for permutation problems. From this perspective, the original PSO operator template (before interpretation), which abstracts how solutions are manipulated, can be regarded as an operator design guideline embedded with the PSO working mechanism. As a matter of fact, various interpretations and approximations have also been made in the previous work of forma analysis for the purpose of facilitating operator derivations [11, 12] (such as the derivation of RTR operator template to blend crossover/line recombination crossover for continuous domain).

By understanding the fact that the perturbation of the current individual is jointly decided by three components (with their degrees of influence distributed proportionally), we can give a stochastic interpretation of the PSO operator template as follows.

The perturbation $\text{dis}_\Psi(X_{t+1}, X_t)$ can be interpreted separately in three cases:

Case 1 if rand() $\leq (w/b)$, $\text{dis}_\Psi(X_t, X_{t-1})$;

Case 2 else if rand() $\leq (w/b) + (c_1 r_1/b)$, $\text{dis}_\Psi(\text{Pb}, X_t)$;

Case 3 else, $\text{dis}_\Psi(\text{Gb}, X_t)$,

where b equals $(w + c_1 r_1 + c_2 r_2)$. In this interpretation, the parameters w and c_1/c_2 are used to represent the probabilities of the current particle's commitment to each of the components. In other words, the new velocity has a probability of (w/b) to stay the same as previous value, a probability of $(c_1 r_1/b)$ to converge towards its personal best record, and a probability of $(c_2 r_2/b)$ to converge towards the global best record.

The decomposition of the flying dynamics of a particle is illustrated in Figure 2. The current particle has three options: a random k-change according to the magnitude of its previous velocity, change towards its personal best record, or change towards the global best record. (Strictly according to the original PSO working mechanism, this should not be a random k-change. Instead it should be a *directed* perturbation following the previous direction. However, since this direction is difficult to "replicate" practically, we choose to simplify this to a random k-change with the magnitude maintained only.)

However, the mixing effect of several distances with different directions is hard to represent in the context of permutation. Modelling the accumulation from a stochastic perspective helps us avoid this unnecessary complication.

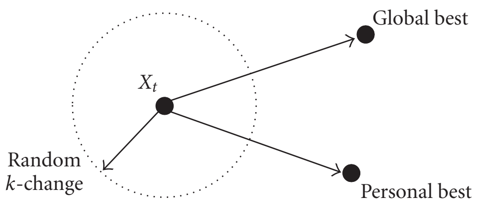


FIGURE 2: Decomposition of the flying dynamics of PSO.

5.3. *Incorporation of direction-A crossover perspective*

Given that we already have the mechanism to separate each distance component, the next question is how to incorporate direction to guide our PSO operator so that the particles can converge towards superior records.

As aforementioned, if a greedy component with superior record considered is selected to perturb the current particle, we directly have

$$\text{dis}_\Psi(X_{t+1}, X_t) \Longrightarrow \text{dis}_\Psi(\text{Pb}, X_t), \tag{28}$$

or

$$\text{dis}_\Psi(X_{t+1}, X_t) \Longrightarrow \text{dis}_\Psi(\text{Gb}, X_t), \tag{29}$$

which means that the particle should jump to the attraction point (i.e., Pb/Gb) without any reservations. However, this is the situation we should try to avoid in practice, since the search may stagnate at a very early stage dominated by the initial superior component. Thus, we desire the current particle to converge somewhere between itself and the attraction point along the right *direction*.

By taking a closer look at the guided PSO operator, we can actually find that the effect of perturbing one individual towards another is the same as making a crossover (e.g., RTR) between these two individuals *where the direction can be naturally retained*. Understanding the greedy components of PSO operator in terms of crossover makes the practical incorporation of direction in the operator much easier.

5.4. *Understanding PSO operator template*

From the above discussion, we can evolve a new interpretation of the PSO operator template. The new position X_{t+1} can be generated according to three cases separately:

Case 1 if rand() $\leq (w/b)$, $O_k(X_t, |\text{dis}_\Psi(X_t, X_{t-1})|)$;

Case 2 else if rand() $\leq (w/b) + (c_1 r_1/b)$, RTR (X_t, Pb, Ψ);

Case 3 else, RTR (X_t, Gb, Ψ),

where for the first case a random k-change, with $k = |\text{dis}_\Psi(X_t, X_{t-1})|$, is adopted to perturb the current particle, while for the second and the third cases a RTR operator template is used to perturb the current particle towards the attraction point (i.e., Pb/Gb). For the first case, only the magnitude of the previous velocity is considered to parameterise the random k-change. For the second and the third cases, the magnitude of velocity should be updated according to the difference between the new position and the previous position of the particle after the RTR "move."

5.5. Brief comparisons with geometric PSO

In this section, our forma analysis framework is compared with the geometric framework [3] which has also been proposed for generalising PSO.

5.5.1. Comparisons on the framework

On the framework level, both frameworks aim to generalise PSO based on underlying optimisation components (e.g., solution representation or distance notion) so that the abstraction of optimisers, either in the form of standard operator template RTR/RRR/RAR or in the form of line segment/ball [3], can be further generalised to arbitrary problem domain. In both cases, these abstractions need to be instantiated to concrete operators by embedding domain knowledge into the corresponding abstraction. In other words, these abstractions all carry some form of domain knowledge, either by using equivalence relations or by using distance notions, so that they can be applied to concrete problem instances.

The main difference on the design concept level lies in the choice of such abstractions and the "carrier" of domain knowledge. For our forma analysis approach, the solution representation is generalised with equivalence relations/classes so that formal representation can be defined in an unified manner, while operators that manipulate the solutions are abstracted as operator templates that process equivalence relations. In contrast, the geometric framework is more about generalising optimisers based on a *notion of distance* where different distance metrics give rise to different operators with regards to the predefined geometric operators using the notions of line segment and ball [3]. Each framework looks at operator design through abstraction and formal description (of either solutions or distance) at different levels, with potentially equivalence relations lying at a slightly lower level than distance notion as distance notion is effectively a derivative of equivalence relations once forma distance [13, 15] is defined under the forma analysis framework.

5.5.2. Comparisons on PSO generalisation

On the practical PSO generalisation level, both approaches are generally different in two places as well.

First of all, the concept of velocity has been removed from the geometric framework (thus, including the simplification of the concept of inertia as a component in geometric crossover), while a random mutation is added to the geometric PSO as a potential replacement for perturbation purposes. In our forma analysis approach, velocity has been interpreted and formulated as distance (more precisely forma distance) in the previous time step. However, velocity itself is a rather complicated concept to formulate as it involves the interpretation of both magnitude and direction which are hard to represent in the context of permutation problems. Certain simplifications and compromises have been made to maintain this concept for future research.

Secondly, the accumulation of greediness toward personal best and global best, balanced by previous velocity (or position), is interpreted differently as well. In geometric PSO, multiparental geometric crossover is used to linearly recombine these positions to produce the next position with different weights taken into account through the concept of product geometric crossover. In contrast, in our forma analysis approach, different convergence components are treated stochastically according to their different weights where higher weight represents higher probability of being treated as the convergence direction for the next time step (and vice versa). By looking at the full picture, different components (personal best, global best, or previous velocity) all share the probability of being selected to guide the next move. Standard RTR operator template is used to converge towards superior solutions with the direction of distance naturally maintained, while standard k-change operator template is used to represent the inertia component.

6. "BLENDED" PSO OPERATORS FOR PERMUTATION PROBLEMS

As mentioned earlier, the formal descriptions of permutation problems implicitly introduces some feasibility constraints to produce a valid solution. When we design PSO operators for permutation problems, these constraints must be satisfied (or handled properly) which is effectively a subproblem to solve. (Of course, these constraints only exist if we are only interested in searching feasible regions, while search techniques making use of infeasible regions are out of the scope in this discussion.) In our previous work [15], we have presented that the application of standard genetic operators for permutation problems can be viewed as a process of *constraint satisfaction*, so as to instantiate the declarative nature of forma analysis in a systematic manner and bring forma analysis to a more practical level. In this section, we will show how the PSO operators for permutation problems can be obtained procedurally based on the operator template from a constraint satisfaction problem (CSP) [17] solving perspective.

According to the aforementioned stochastic interpretation of PSO operator template, the outcome is effectively a blended operator with three different "phases": k-change according to $k = |\text{dis}_\Psi(X_t, X_{t-1})|$, RTR (X_t, Pb, Ψ), and RTR (X_t, Gb, Ψ). The separate instantiations of k-change and RTR in the context of different formal descriptions of permutation should directly give rise to the instantiations of the corresponding blended PSO operators.

6.1. Instantiations of k-change

6.1.1. Position-based k-change: $O_{k\text{-}pos}$

For the position-based description, $\text{dis}_{\Psi_{\text{pos}}}(X_{t+1}, X_t)$ can be obtained as the hamming distance between X_{t+1} and X_t

under basis Ψ_{pos}. Thus, we only need to apply a distance of k based on X_t to generate X_{t+1} such that $\text{dis}_{\Psi_{\text{pos}}}(X_{t+1}, X_t) = k$. For example, given that X_t represents a permutation $(1, 2, 3, 4)$ such that

$$X_t = \left\{\xi^1_{\text{pos}(1)}, \xi^2_{\text{pos}(2)}, \xi^3_{\text{pos}(3)}, \xi^4_{\text{pos}(4)}\right\}, \tag{30}$$

what would be the potential candidates for the permutation X_{t+1} so that they have a distance of k (e.g., $k = 2$)?

The most straightforward thought would be to randomly select k equivalence relations and change the equivalence classes they fall into. However, the feasibility constraint C_{pos} induced by position-based description must be satisfied to produce a valid permutation. In this case, the k-change operator for permutation following the position-based description should involve a *constraint satisfaction subproblem*, where C_{pos} must be satisfied to guarantee a valid permutation.

The CSP we consider here is defined as the k-change operator itself with C_{pos} satisfied. Classical CSP techniques can be directly utilised to implement the operator. Now, we will follow the above example to illustrate how CSP can be effectively used in the position-based PSO operator design.

Given X_t $(\{\xi^1_{\text{pos}(1)}, \xi^2_{\text{pos}(2)}, \xi^3_{\text{pos}(3)}, \xi^4_{\text{pos}(4)}\})$, we can first uninstantiate k (e.g., $k = 2$) equivalence relations to produce a partial permutation for a potential distance of 2. This gives us C^2_4 options from which we can uniformly select one to serve as the base (partial permutation) of X_{t+1} (e.g., $\{\xi^-_{\text{pos}(1)}, \xi^2_{\text{pos}(2)}, \xi^-_{\text{pos}(3)}, \xi^4_{\text{pos}(4)}\}$).

Then, what we need to do is simply to reinstantiate these 2 equivalence relations to suitable classes such that C_{pos} is satisfied. The first position of the partial solution $(-, 2, -, 4)$ is considered first. The domain of the first position is $\{1, 3\}$, while the domain of the third position is $\{1, 3\}$ as well. After an equivalence class (1 or 3) is chosen for the first position, the domain is reduced for the third position through constraint propagation. After instantiating all the possible solutions, X_{t+1} can be randomly selected among the whole set of feasible solutions to the CSP.

In fact, the working mechanism of $O_{k\text{-pos}}$ has a similar effect as the *scramble* mutation [18], which rearranges a certain number of positions. The only difference is that the selection of these positions should be purely random, other than inside a continuous block. Thus, $O_{k\text{-pos}}$ can be defined as the *modified scramble mutation* operator with k random positions, where k is decided by the magnitude of the current velocity $|\text{dis}_{\Psi_{\text{pos}}}(X_t, X_{t-1})|$.

6.1.2. *Precedence-based k-change: $O_{k\text{-prec}}$*

For the precedence-based description of permutation, the distance of two permutations is the number of different precedence relations $\text{dis}_{\Psi_{\text{prec}}}(P_1, P_2)$. As aforementioned, the *bubble sort* algorithm is sufficient to calculate the precedence distance of two permutations, with the distance decided as the number of adjacency-swap to sort one permutation towards another.

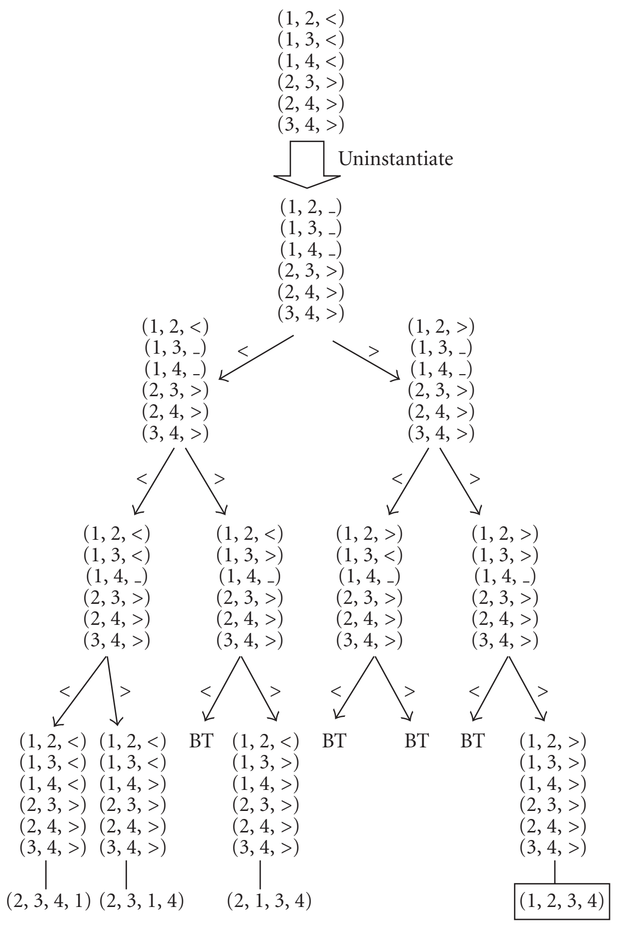


FIGURE 3: Illustration of reinstantiation of a permutation based on precedence-based description. Symbol "$(i, j, >)$" means element i is before element j, while "$(i, j, <)$" means otherwise. The framed permutation has a "strict" distance of 3 to the initial permutation.

Assuming $k = 3$ which should be applied to $X_t = (2, 3, 4, 1)$ as an example, X_{t+1} can be obtained by solving the CSP as shown in Figure 3.

Given that X_t can be represented as

$$\left\{\xi^0_{\text{prec}(1,2)}, \xi^0_{\text{prec}(1,3)}, \xi^0_{\text{prec}(1,4)}, \xi^1_{\text{prec}(2,3)}, \xi^1_{\text{prec}(2,4)}, \xi^1_{\text{prec}(3,4)}\right\}, \tag{31}$$

to apply a distance of 3 will be a reinstantiation of 3 equivalence relations. This gives us $C^3_{C^2_n}$ options from which we can uniformly select one as the partial permutation to generate X_{t+1}. For example, the partial permutation can be

$$\left\{\xi^-_{\text{prec}(1,2)}, \xi^-_{\text{prec}(1,3)}, \xi^-_{\text{prec}(1,4)}, \xi^1_{\text{prec}(2,3)}, \xi^1_{\text{prec}(2,4)}, \xi^1_{\text{prec}(3,4)}\right\}. \tag{32}$$

Reinstantiating these 3 precedence relations to suitable classes such that C_{prec} is satisfied can give us potential candidates for X_{t+1}.

As shown in Figure 3, the reinstantiation of precedence relations is carried out one after another. The domain of each relation is simply $\{0, 1\}$. Alternatively, we use "<" and

“>” to represent the precedence such that symbol “$(i, j, >)$” means element i is before element j while “$(i, j, <)$” means otherwise. X_{t+1} can be randomly selected among the set of feasible solutions to the CSP.

By observing the effect of changing a single precedence equivalence class, it is not difficult to find that a 1-change (minimal mutation) reduces to the *adjacent swap* mutation [18], which swaps two contiguous elements. Thus, $O_{k\text{-prec}}$ can be approximately regarded as equivalent to a k-iterated adjacent swap mutation with k decided by the magnitude of the current velocity $|\text{dis}_{\Psi_{\text{prec}}}(X_t, X_{t-1})|$, which can be effectively calculated through bubble sort.

6.1.3. Adjacency-based k-change: $O_{k\text{-adj}}$

Since each potential edge is defined as an equivalence relation for the adjacency-based description of permutation, the distance of two permutations corresponds to the “edge-difference” between them.

For the simplicity of illustration, we assume that $k = 4$, which should be applied to X_t to generate X_{t+1}. As an example, given the base permutation $(1, 2, 3, 4)$ being represented as

$$X_t = \left\{\xi^1_{\text{adj}(1,2)}, \xi^0_{\text{adj}(1,3)}, \xi^1_{\text{adj}(1,4)}, \xi^1_{\text{adj}(2,3)}, \xi^0_{\text{adj}(2,4)}, \xi^1_{\text{adj}(3,4)}\right\}, \tag{33}$$

we first uninstantiate 4 relations (randomly chosen from $C^4_{C^2_n}$ options) to produce the partial permutation for X_{t+1}, for example:

$$X_{t+1} = \left\{\xi^-_{\text{adj}(1,2)}, \xi^-_{\text{adj}(1,3)}, \xi^1_{\text{adj}(1,4)}, \xi^1_{\text{adj}(2,3)}, \xi^-_{\text{adj}(2,4)}, \xi^-_{\text{adj}(3,4)}\right\}. \tag{34}$$

By solving the CSP to generate X_{t+1} such that C_{adj} is satisfied, we can get a solution permutation $(1, 3, 2, 4)$ with a distance of 4 to X_t:

$$\left\{\xi^0_{\text{adj}(1,2)}, \xi^1_{\text{adj}(1,3)}, \xi^1_{\text{adj}(1,4)}, \xi^1_{\text{adj}(2,3)}, \xi^1_{\text{adj}(2,4)}, \xi^0_{\text{adj}(3,4)}\right\}. \tag{35}$$

The minimal mutation implied by the adjacency-based description is an edge 2-change mutation, because 1-change automatically violates the feasibility constraint C_{adj}. (A standard edge2-change mutation involves the “replacement” of 2 edges which in turn results in the change of 4 adjacency equivalence relations, since there are also changes involving negative edges. However, from the implementation perspective only phenotypical (or positive) edges are considered.) This operator is equivalent to the *edge-reverse mutation* in the literature [18], which reverses the positions of a segment in the permutation in such a manner that the feasibility constraint C_{adj} can be satisfied automatically. Due to the fact that any edge-reverse mutation involves the change of 2 edges, $O_{k\text{-adj}}$ can be generally approximated by a k^*-iterated edge-reverse mutation with k^* decided by k as follows:

$$k^* = \text{round}\left(\frac{k}{2}\right), \tag{36}$$

where any function that rounds $k/2$ to its nearest integer is sufficient.

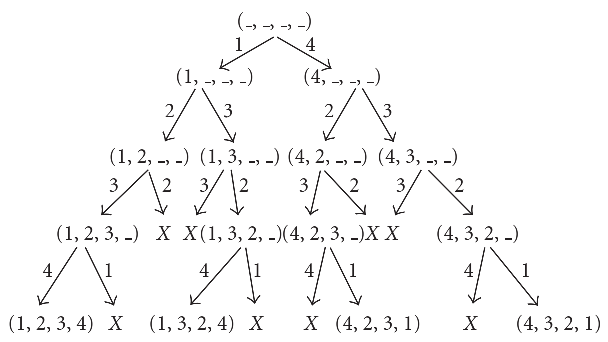


FIGURE 4: The reinstantiation of P_c for RTR_{pos}.

6.2. Instantiations of RTR

6.2.1. Position-based RTR: RTR_{pos}

For the position-based description Ψ_{pos}, its feasibility constraint C_{pos} largely reduces the number of feasible solutions to the basic CSP, where RTR is interpreted as recombination of parental equivalence classes, because not all combinations of them lead to valid permutations. By satisfying C_{pos} while instantiating P_c, we can obtained all potential candidates for RTR_{pos} as a result of solving the CSP.

For example, given $P_{p1} = (1, 2, 3, 4)$ and $P_{p2} = (4, 3, 2, 1)$, RTR_{pos} can produce potential candidate(s) for P_c by reinstantiating it (as shown in Figure 4) with corresponding equivalence classes, as shown in (37)

$$\begin{aligned} P_{p1} &= \left\{\underline{\xi^1_{\text{pos}(1)}}, \xi^2_{\text{pos}(2)}, \xi^3_{\text{pos}(3)}, \underline{\xi^4_{\text{pos}(4)}}\right\}, \\ P_{p2} &= \left\{\xi^4_{\text{pos}(1)}, \underline{\xi^3_{\text{pos}(2)}}, \underline{\xi^2_{\text{pos}(3)}}, \xi^1_{\text{pos}(4)}\right\}, \\ P_c &= \left\{\xi^1_{\text{pos}(1)}, \xi^3_{\text{pos}(2)}, \xi^2_{\text{pos}(3)}, \xi^4_{\text{pos}(4)}\right\}. \end{aligned} \tag{37}$$

To transmit position features from parents to children and interpret the feasibility constraint C_{pos} in a more natural way, we produce a fully-transmitting crossover for permutation, namely *position transmitting crossover* (PTX), by identifying the constraint satisfaction process of operator as a CSP.

In PTX, both C_{pos} and transmitting C_t have to be satisfied as an interpretation of its CSP. (Transmitting can be regarded as an additional constraint imposed by operator.) In this sense, constrained positions are clustered to function separately. For example, given two permutations

$$\begin{aligned} &\left(3, 6, \underline{5, 4, 2}, \underline{7, 8}, 1\right), \\ &\left(3, 6, \underline{2, 5, 4}, \underline{8, 7}, 1\right), \end{aligned} \tag{38}$$

we are able to construct the constraint graph implied by both C_{pos} and transmitting C_t, as shown in Figure 5 with constrained positions linked together.

The construction of the constraint graph is straightforward to understand—a value that has been taken for one position must be forbidden (constrained) for another position. For example, for position 3 (P_3 in Figure 5) either

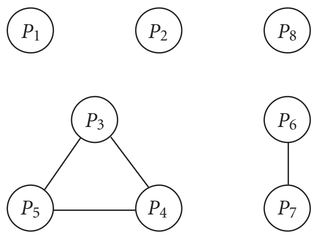


Figure 5: Illustration of the constraint graph of PTX.

5 or 2 should be chosen to achieve transmitting. However, choosing either of them will forbid another position (e.g., P_4 or P_5 in Figure 5) from taking the same value, which effectively reduces the domain for another position.

The only possibility that the value taken by one position does not constrain the value taken by another is that the parents both take the same value for that position (e.g., P_1 in Figure 5), where taking (the only) one value automatically satisfies the constraint.

Thus, as long as the constrained positions are transmitted all-together to the child, PTX always satisfies both C_{pos} and C_t, and the child solution is always a valid permutation. Following the above example, the child produced by PTX could be

$$\left(3, 6, \underline{5, 4, 2}, \underline{8, 7}, 1\right). \tag{39}$$

In fact, PTX works in an equivalent manner as *cycle crossover* [19] in the literature, which preserves absolute positions in parents and guarantees feasibility of child solutions.

6.2.2. Precedence-based RTR: RTR$_{prec}$

To achieve transmitting in precedence-based crossover, both transmitting C_t and C_{prec} should be satisfied from the CSP viewpoint. It is easy to verify that this CSP is *solvable*, since (in the worst case) taking all the equivalence classes for either parent to produce child always gives one possible solution such that constraints (C_t and C_{prec}) are satisfied.

Furthermore, precedence relation is special in that its equivalence class is either 1 or 0. This means that in the case when two parents are different for a certain equivalence relation $\psi_{\text{prec}(e_i,e_j)}$, the domain of $\psi_{\text{prec}(e_i,e_j)}$ for the child is always $\{0, 1\}$, which implicitly means that $\psi_{\text{prec}(e_i,e_j)}$ can take any value for the child. In the case when two parents are the same for $\psi_{\text{prec}(e_i,e_j)}$, the corresponding equivalence class is fixed for the child to achieve transmitting.

For example, given permutations $(3, 1, 2, 4)$ and $(4, 2, 3, 1)$ such that

$$\begin{gathered}\left\{\xi^1_{\text{prec}(1,2)}, \underline{\xi^0_{\text{prec}(1,3)}}, \xi^1_{\text{prec}(1,4)}, \xi^0_{\text{prec}(2,3)}, \xi^1_{\text{prec}(2,4)}, \xi^1_{\text{prec}(3,4)}\right\},\\ \left\{\xi^0_{\text{prec}(1,2)}, \underline{\xi^0_{\text{prec}(1,3)}}, \xi^0_{\text{prec}(1,4)}, \xi^1_{\text{prec}(2,3)}, \xi^0_{\text{prec}(2,4)}, \xi^0_{\text{prec}(3,4)}\right\},\end{gathered} \tag{40}$$

the partial permutation as a child to achieve transmitting can be

$$\left\{\xi^-_{\text{prec}(1,2)}, \xi^0_{\text{precc}(1,3)}, \xi^-_{\text{prec}(1,4)}, \xi^-_{\text{prec}(2,3)}, \xi^-_{\text{prec}(2,4)}, \xi^-_{\text{prec}(3,4)}\right\}, \tag{41}$$

while the uninstantiated relations can be reinstantiated randomly to produce the child permutation, by solving a CSP such that simply C_{prec} should be satisfied.

In fact, strictly transmitting crossover is also possible for precedence-based description. Due to the fact that the reinstantiation process of precedence relations is equivalent to the *topological sorting problems* [20], where a *partial order* needs to be completed to a linear order (in a *directed acyclic graph* (DAG) based on precedence) with a complexity of O(edges + vertices), we argue that the reinstantiation of precedence relations in the considered CSP can be solved in a deterministic polynomial-time in terms of topological sorting problems. (The partial order is defined by the equivalence relations where the parents are equivalent—the order that must be enforced for the child.)

In the literature, *precedence preservative crossover* (PPX) [18] was found to be strictly transmitting. The underlining principle in PPX is that the precedence equivalence classes of parents are passed to the child in such an order (from left to right or more specifically from the node with no incoming edges in the precedence graph) that both C_t and C_{prec} are satisfied automatically. Practically, this order is implemented using a "from" table which indicates the switching process between the two parents.

Many readers may find that this is rather similar to the most popular algorithm used for topological sorting where the order can be completed by starting from the node(s) with no incoming edges. Switching between two parents simply aims at recombining the precedence equivalence classes of the two parents.

It is also easy to find that the set of all possible solutions produced by PPX is in fact a subset of the set of solutions produced by the above CSP approach. In other words, for each of the solution produced by PPX, there is always a corresponding reinstantiation of the partial child permutation.

6.2.3. Adjacency-based RTR: RTR$_{adj}$

Regarding the adjacency-based description of permutation which has been proved to be non *g-separable* [10], literature [11, 12] pointed out that transmission cannot be achieved without sacrificing assortment. From the CSP viewpoint, this implies that C_{adj} and C_t all together may make the corresponding CSP not *strictly* solvable (if the original parental permutations are not allowed to be repeated as a child permutation). Through investigation, we can also find that it is the case when two parents are the same for some equivalence relations that makes the CSP NP-complete in "strict" sense. For those adjacency relations that two parents are different, no restriction will be applied to the child solution for that relation (as both $\{1, 0\}$ are allowed).

```
1. for all particle $i$ do
2.     initialize $P_i$ (with its velocity $V_i$) and evaluate $P_i$;
3. endfor
4. do
5.     for all particle $i$ do
6.         set $\mathrm{Pb}_i$ as best position found by particle $i$ so far;
7.         set Gb as best position found by all particles so far;
8.         if (rand() $< w/b$):
9.             modified scramble mutation according to $|V_i|$;
10.        elseif (rand() $< (w + c_1 r_1)/b$):
11.            cycle crossover CX ($P_i, \mathrm{Pb}_i$);
12.        else:
13.            cycle crossover CX ($P_i$, Gb);
14.        endif
15.        evaluate $P_i$ and update $|V_i|$;
16.    endfor
17. until (halting criterion met);
```

Algorithm 1: Position-based blended PSO scheme ($\mathrm{PSO_{pos}}$).

```
1. for all particle $i$ do
2.     initialize $P_i$ (with its velocity $V_i$) and evaluate $P_i$;
3. endfor
4. do
5.     for all particle $i$ do
6.         set $\mathrm{Pb}_i$ as best position found by particle $i$ so far;
7.         set Gb as best position found by all particles so far;
8.         if (rand() $< w/b$):
9.             $k$-iterated adjacent-swap with $k = |V_i|$;
10.        elseif (rand() $< (w + c_1 r_1)/b$):
11.            precedence preservative crossover PPX ($P_i, \mathrm{Pb}_i$);
12.        else:
13.            precedence preservative crossover PPX ($P_i$, Gb);
14.        endif
15.        evaluate $P_i$ and update $|V_i|$;
16.    endfor
17. until (halting criterion met);
```

Algorithm 2: Precedence-based blended PSO scheme ($\mathrm{PSO_{prec}}$).

Furthermore, for those edges which are absent in both parents ("negative edges" $\xi^0_{\mathrm{adj}(i,j)}$), transmitting automatically forbids them from being included in the children. In this sense, the induced CSP is equivalent to the *Hamiltonian cycle problem* [21] in an *incomplete graph*, which is NP-complete. (It should be pointed out that this Hamiltonian cycle problem is NP-complete only if the parental permutations are forbidden for the child, since the parents automatically gives 2 possible solutions. However, finding the third solution is still NP-complete.) Those edges which are common for both parents ("positive edges" $\xi^1_{\mathrm{adj}(i,j)}$) effectively enforce that some edges must be included in the Hamiltonian cycles (solutions). This is actually a constrained version of the original Hamiltonian cycle problem induced by "negative edges," which is also NP-complete.

Thus, approximation through relaxation of C_t may be required to produce a valid *new* child permutation. In the literature, *enhanced edge recombination* devised by [22] has been noticed as an effective "edge-aware" recombination operator which has a high rate (98%) of adjacency transmission. The transmitting of adjacency formae is approximated by creating an edge table with the related edge information for both parents and selecting edges in a heuristic manner to construct the child solution.

6.3. The blended PSO schemes

In summary, the derived blended PSO schemes with different formal descriptions of permutation can be described in the following Algorithms 1, 2, and 3.

7. EXPERIMENTS OF PSO SCHEMES ON QUADRATIC ASSIGNMENT PROBLEMS

To illustrate the search dynamics of the derived blended PSO schemes, we evaluated the performance of these PSO schemes on the quadratic assignment problem (QAP) benchmarks. However, due to the fact that for QAP the absolute positioning of element is more related to the quality of solution [14], it is estimated that the PSO scheme with position-based description (i.e., $\mathrm{PSO_{pos}}$) should perform better than the other PSO schemes.

After a brief description of the problem formulation, we show both the experiment configurations and the experimental results, followed by a few discussions to help the understanding of the benefits of our approach.

7.1. Problem formulation of QAP

The quadratic assignment problem (QAP) is an important problem in both practice and theory. Many practical problems can be formulated as QAPs [23–25]. The QAP can be described as the problem of assigning a set of facilities to a set of locations with the given distances between the locations and the given flows between the facilities. The goal is to place the facilities on locations such that the sum of the product between flows and distances is minimised. Formally, given n facilities and n locations, two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{rs}]$, where a_{ij} is the distance between locations i and j, and b_{rs} is the flow between facilities r and s, the QAP can be formally defined as

$$\mathrm{Min}(\psi)\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}a_{\psi_i\psi_j}. \tag{42}$$

The QAP is a class of NP-hard optimisation problems. It is considered as one of the hardest optimisation problems as general instances of size larger than 20 cannot be solved to optimality. Therefore, to practically solve QAP with high quality solutions various heuristic algorithms have been proposed [26–28] trying to find high quality solution with limited computational resources.

```
1. for all particle $i$ do
2.     initialize $P_i$ (with its velocity $V_i$) and evaluate $P_i$;
3. endfor
4. do
5.     for all particle $i$ do
6.         set $\text{Pb}_i$ as best position found by particle $i$ so far;
7.         set Gb as best position found by all particles so far
8.         if (rand() < $w/b$):
9.             $k$-iterated edge-reverse mutation with $k = \text{round}(|V_i|/2)$
10.        elseif (rand() < $(w + c_1 r_1)/b$):
11.            enhanced edge crossover EX ($P_i, \text{Pb}_i$);
12.        else:
13.            enhanced edge crossover EX ($P_i$, Gb);
14.        endif
15.        evaluate $P_i$ and update $|V_i|$;
16.    endfor
17. until (halting criterion met);
```

Algorithm 3: Adjacency-based blended PSO scheme (PSO_{adj}).

7.2. Benchmarks and experimental settings

The benchmarks for this experimental study are acquired from QAP-LIB [29]. It has been pointed out [30] that there are four types of QAP instances: unstructured, randomly generated instances, unstructured instances with grid-distances, real-life instances, real-life like instances. We select 2 instances for each type of QAP as our benchmarks, with size no smaller than 20. For type 1, we choose `Tai20a` and Tai40a. For type 2, we choose Nug20 and Sko56. For type 3, we choose Bur26a and Ste36a. For type 4, we choose Tai20b and Tai40b.

For each of the instances, fine tuning is carried out for each of the algorithms to reach its best performance among different combinations of parameter settings with equal number of generations. The parameter settings with the best performance over 20 independent runs for each algorithm will be used to get the execution results. (The performance is evaluated by considering both its average best solution found and its average number of generations to reach its best solution. The number of generation to reach its best solution is only considered when two parameter settings have the same average best solution.)

The original free parameters for each instances are W and C, where W represents the parameter to control the inertia weight and C represents the value taken by c_1 and c_2 (as in literature c_1 and c_2 often adopt the same value). By taking a closer look, it is not difficult to realise that in our PSO schemes for QAP, the probability of taking each branch can be simplified to

$$\frac{(W/C)}{(W/C)+r_1+r_2}, \quad \frac{r_1}{(W/C)+r_1+r_2}, \quad \frac{r_2}{(W/C)+r_1+r_2}. \tag{43}$$

The above simplification means that the essential element controlling the search dynamics is (W/C). This implies that (W/C) is effectively the *only* parameter we need to deal with for our PSO scheme, where a relatively larger (W/C) would encourage a more explorative search, while a relatively smaller (W/C) would favour a greedier search towards either one of the superior attraction points (i.e., Pb/Gb).

The population size is fixed to be an appropriate number (100) for each instance through observations. Since we do not have preknowledge about the discrete PSO operators, the tuning process is only carried out in a coarse manner, where the options for W and C are both formatively set to be $\{1, 2, 3, \ldots, 9, 10\}$ (although only W/C matters).

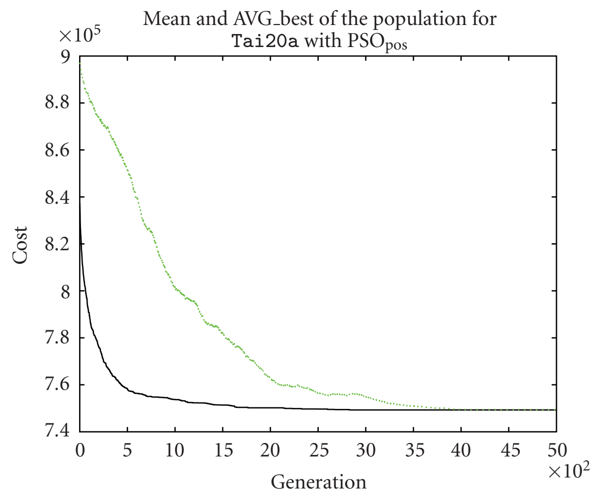
7.3. Observations and experimental results

According to the tuned parameter settings, 50 independent runs were executed for each PSO scheme under each instance to produce experimental results.

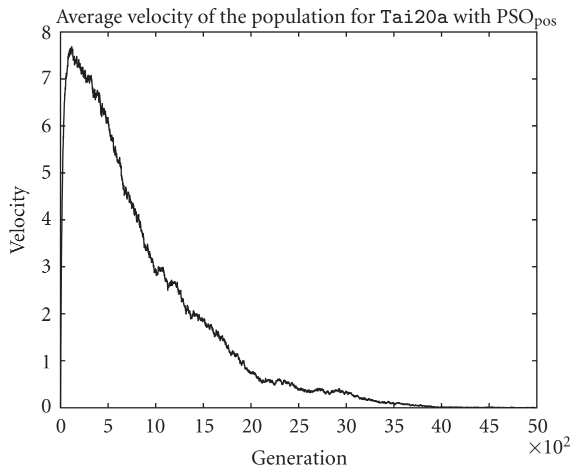
7.3.1. Comparisons among PSO schemes

To examine the search behavior of the proposed PSO schemes, we track three components of each scheme that are felt to be essential to the search dynamics of PSO. These three components are: the *mean cost* of the population; the *average best* of the population; the *average velocity* of the population. The mean cost and average best of the population are plotted together to reveal the convergence pattern, while the average velocity is plotted to track the exploration power. Only the search patterns for `Tai20a` are illustrated here (as shown in Figure 6), since the patterns for all the other instances are quite similar. In addition, the execution results are also shown in Table 1.

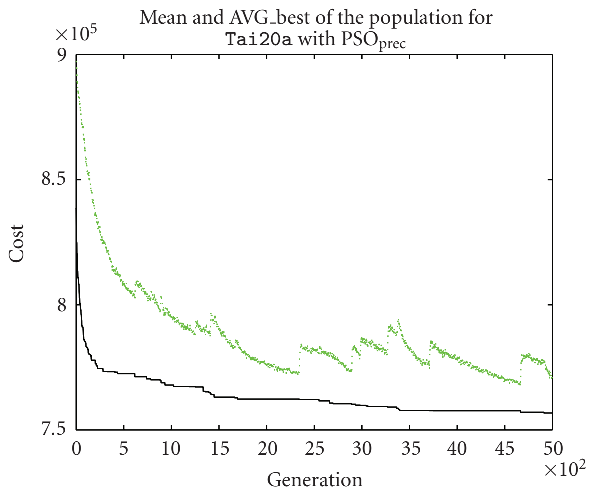
Through observation shown in Figure 6(a), we can find that for position-based PSO (PSO_{pos}) both the mean cost and the average velocity of population are changing throughout the search process smoothly, as the whole population is constantly converging towards the attraction point(s). The pattern of the corresponding velocity (as shown in Figure 6(b)) implies that the exploration power of the whole population initially increases to a relatively



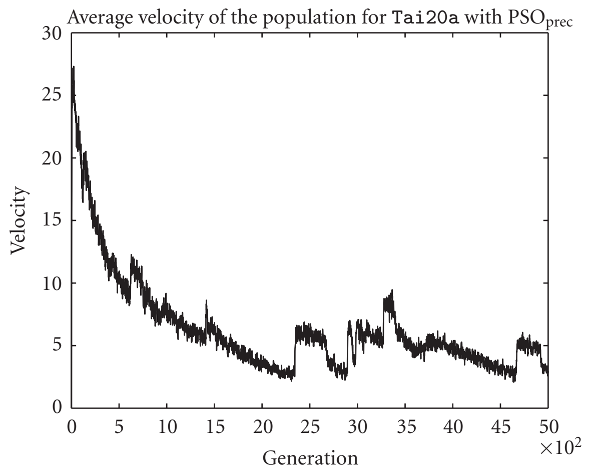
(a) Tai20a-mean and best with PSO$_{pos}$



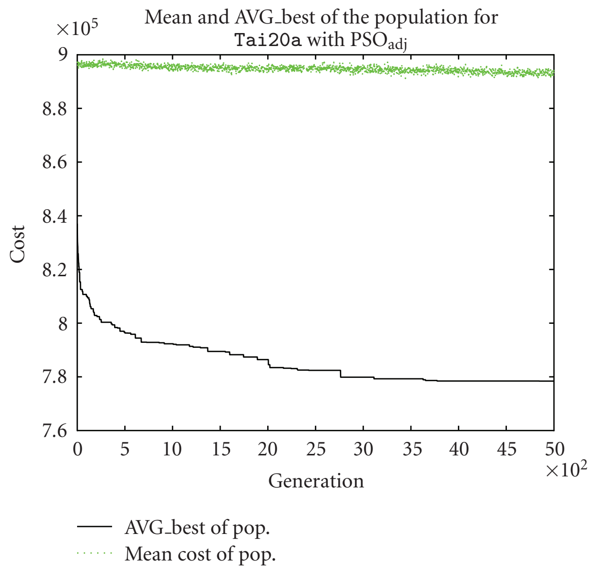
(b) Tai20a-velocity with PSO$_{pos}$



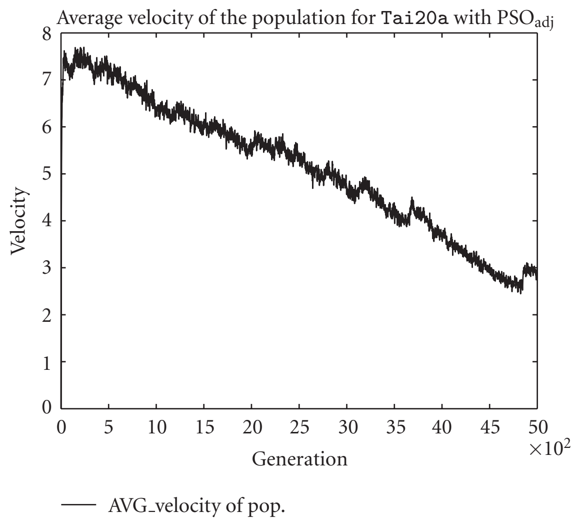
(c) Tai20a-mean and best with PSO$_{prec}$



(d) Tai20a-velocity with PSO$_{prec}$



(e) Tai20a-mean and best with PSO$_{adj}$



(f) Tai20a-velocity with PSO$_{adj}$

FIGURE 6: The mean cost, average best cost, and average velocity of the population over generation for Tai20a with different PSO schemes (with global topology).

Table 1: Experimental results over 50 independent runs for different PSO schemes.

Tai20a	SIZE	GEN #	POP_size	Tai40a	SIZE	GEN #	POP_size
	20	5000	100		40	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS	**749292.1871**		**12172.79199**	PSO_POS	**3332304**		**27611.47461**
PSO_PREC	756756.8125		6750.588867	PSO_PREC	3475673		23905.56437
PSO_ADJ	778401.8125		10900.51661	PSO_ADJ	3547892		17521.19283
Nug20	SIZE	GEN #	POP_size	Sko56	SIZE	GEN #	POP_size
	20	5000	100		65	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS	**2680.120117**		**34.834686**	PSO_POS	**35955.64063**		**309.11557**
PSO_PREC	2717.847864		27.325028	PSO_PREC	36078.96543		257.68971
PSO_ADJ	2746.896391		25.121398	PSO_ADJ	38423.78542		375.67864
Bur26a	SIZE	GEN #	POP_size	Ste36a	SIZE	GEN #	POP_size
	26	5000	100		36	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS	**5445906.1**		**8688.298828**	PSO_POS	**10808.51953**		**436.841125**
PSO_PREC	5462411.9		8622.741697	PSO_PREC	11356.24567		553.488941
PSO_ADJ	5489783.6		7989.741787	PSO_ADJ	11667.04213		604.555.718
Tai20b	SIZE	GEN #	POP_size	Tai40b	SIZE	GEN #	POP_size
	20	5000	100		40	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS	**124803488**		**1190690.75**	PSO_POS	**696782272**		**28186792**
PSO_PREC	125760656		1622274.93	PSO_PREC	706244099		17986929
PSO_ADJ	128727848		1379135.25	PSO_ADJ	722001585		18978233

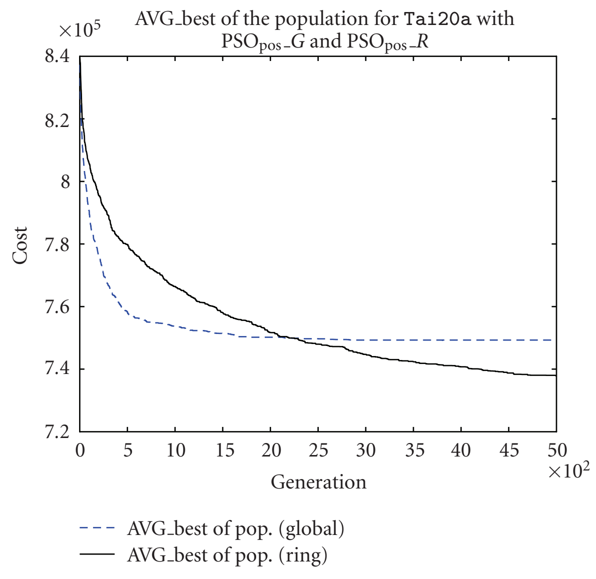
higher level, before it decreases constantly as the search goes to a later stage. These searching patterns are generally very similar as those in the traditional PSO in the real-vector space.

The situation for PSO_{prec} is however slightly different from PSO_{pos} according to the patterns shown in Figures 6(c) and 6(d). Although the mean quality of the whole population still converge towards the superior individuals (i.e., the global/local best solutions) and the average velocity decreases with the generation number, it is observed that there are some “resistances” in its convergence process. The situation is even worse for PSO_{adj} (as shown in Figures 6(e) and 6(f)), as the mean quality of the whole population *hardly* converges, and the average velocity of the population decreases more slowly relative to the average velocity for PSO_{pos}. Presumably, the degree of such resistance is the main cause for performance loss/gain, considering the fact that both PSO_{prec} and PSO_{adj} are outperformed by PSO_{pos} (as shown in Table 1).

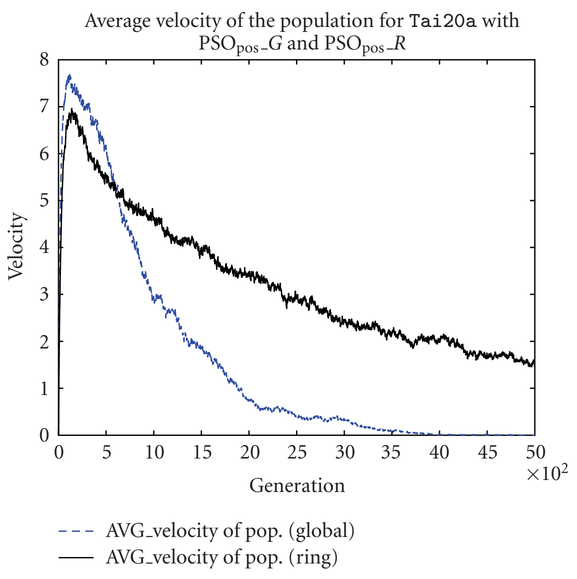
This can be mainly explained by the different information transfer efficiencies with different descriptions for QAP. The position-based PSO scheme (PSO_{pos}) is the most efficient scheme among the three since absolute positioning is being processed for its corresponding position-based description, which is most related to the quality of solution for QAP. From the implementation standpoint, the “recombination” of the position-based equivalence classes from both the current individual and its superior records serves as the main drive for convergence.

However, the information processing of absolute positioning is disrupted by both precedence description and adjacency description to different degrees. This is also quite obvious from the implementation standpoint, since the “recombination” of precedence/adjacency information certainly will not produce the convergence of solution quality efficiently in terms of absolute positioning. The degree of such “disruption/deviation” is mainly decided by its correlations to the positional description. This can be further illustrated by the fact that precedence-based PSO performs better than adjacency-based PSO. As a matter of fact, precedence relations are more correlated to positional relations, which can be easily understood by inspecting the shift operator—as the number of precedences changed by the shift operator increases, so does the number of positions in a smooth progression. In contrast, adjacency relations are found to be poorly correlated with positional relations, since the number of adjacency relations changed by edge-reverse mutation is poorly correlated with the changes in the absolute positioning of permutations.

The above results also reflect the main argument we are making for the methodology in this paper: the search behavior and performance of the derived operator depend on the description for the specific problem. Further estimations can be that PSO_{prec} should perform well in those problems where precedence information processing in permutation is essential (e.g., JSSP), while PSO_{adj} should perform well in those problems where adjacency/edge information is related to solution quality (e.g., TSP).



(a) Tai20a-AVG_best(global versus ring)



(b) Tai20a-AVG_velocity (global versus ring)

FIGURE 7: Comparison of average best cost and velocity of the population over generation for Tai20a between the global and ring structured PSO$_{pos}$ schemes.

TABLE 2: Experimental results over 50 independent runs for PSO schemes with different topologies against a standard GA.

Tai20a	SIZE	GEN #	POP_size	Tai40a	SIZE	GEN #	POP_size
	20	5000	100		40	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS_G	749292.1871		12172.79199	PSO_POS_G	3332304		27611.47461
PSO_POS_R	737858.2491		9983.63476	PSO_POS_R	3307909		28703.17578
GA	**721765.0625**		**6995.33252**	GA	**3232640**		**14233.77148**
Nug20	SIZE	GEN #	POP_size	Sko56	SIZE	GEN #	POP_size
	20	5000	100		65	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS_G	2680.120117		34.834686	PSO_POS_G	35955.64063		309.11557
PSO_POS_R	2629.800049		24.587896	PSO_POS_R	35452.19922		179.47287
GA	**2606.199951**		**21.662531**	GA	**35040.23828**		**184.824081**
Bur26a	SIZE	GEN #	POP_size	Ste36a	SIZE	GEN #	POP_size
	26	5000	100		36	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS_G	5445906.1		8688.298828	PSO_POS_G	10808.51953		436.841125
PSO_POS_R	**5434604.5**		**3932.839611**	PSO_POS_R	10252.40039		235.487991
GA	5436641.5		5282.220703	GA	**10061.67969**		**219.714996**
Tai20b	SIZE	GEN #	POP_size	Tai40b	SIZE	GEN #	POP_size
	20	5000	100		40	10000	100
Algorithms	Mean best		Std Dev	Algorithms	Mean best		Std Dev
PSO_POS_G	124803488		1190690.75	PSO_POS_G	696782272		28186792
PSO_POS_R	**123274304**		**783373.25**	PSO_POS_R	**662772480**		**13308939**
GA	125778728		5234746.11	GA	679525504		18660756

7.3.2. Extended comparisons

In addition, another PSO scheme for QAP is also implemented with *ring* topology based on PSO$_{pos}$ to illustrate the benefits of changing topology. As shown clearly in Figure 7(a), the position-based PSO scheme with ring topology (PSO$_{pos}$_R) outperforms the position-based PSO scheme with global topology (PSO$_{pos}$_G) by enhancing the exploration power through the *implicit search control* introduced by ring topology. In order to compare the exploration power of both PSOs, the average velocities of both schemes are plotted in Figures 7(b). From Figure 7(b), we can easily find that the ring structured PSO (PSO_{pos}_R) has a greater persistence in maintaining its velocity level than the global PSO (PSO_{pos}_G), which in turn enables a better exploration of the search space. Another observation is that, by the end of the execution, the average velocity of PSO_{pos}_R has not decreased to 0 yet, which means that given a larger number of generation an even better performance can be expected.

The mean-best and standard deviation produced by our PSO schemes for each instance are presented in Table 2 under algorithms PSO_{pos}_G (for global structure) and PSO_{pos}_R (for ring structure). The results are also compared against a steady state standard GA using cycle crossover, with swap mutation (mutation rate = 0.1).

From the results, we can see that PSO_{pos}_R with ring topology overall outperforms PSO_{pos}_G with global topology, while generally GA still performs slightly better than the PSO schemes we have for some cases. We understand that this inferiority is mainly caused by the simplification of the inertia component in the PSO operator template, since a random k-change is not good enough to represent a directed k-change which should be parameterised by its previous velocity. This drawback requires our further research on how to replicate and apply distance, so that the previous distance/velocity can be retained and applied during the next generation. If we are able to implement this, the PSO should display a better convergence pattern. Also, further tuning and exploration of PSO options will inevitably lead to improved performance. However, we should point out that the main aim of this paper is certainly not to produce sophisticated PSO schemes with competitive results. Instead, the intent of this work is to generalise PSO in a formal manner, adapt its working mechanism to the permutation problem domain with reasonably good performance, and hopefully show some future research directions on generalising PSO. In this case, performance comparison at a competitive level is not performed due to the fact that most of those results were produced by highly tuned/adapted hybrid algorithms [5, 31] where the separate contribution of each component is usually hard to justify, and comparisons against them would be of limited value.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented how the original PSO operator can be generalised in a formal manner to the permutation problem domain using forma analysis, with both the formal descriptions of permutation and a stochastic PSO operator template defined. By considering the application of operators as a process of constraint satisfaction, we derived several concrete PSO schemes for permutation problem, each of which involves a different assumption made on the description of the search space. Through observations of the search patterns of the derived PSO schemes together with the ring structured extension of position-based PSO on the QAP benchmarks, it is clear that the description choice is a critical issue in operator design, and the position-based PSO scheme for QAP achieves a certain degree of convergence towards the optimum in a similar manner as the traditional PSO for real-vector space, with results comparable to a standard GA.

More importantly, we have presented in this paper a principled approach to formally derive algorithms with regard to the actual problem domain, in which case the behaviors and the performance of the derived algorithms are directly related to the assumption we make to describe the search space.

In the future, efforts on the improvement of these discrete PSO schemes are possible by considering additional issues (e.g., topological search control, local search, and even parameter selections). Application of our methodology to a wider range of problems and optimisation techniques can also be explored. In addition, the interpretation of applying a directed k-change (e.g., distance replication) in the context of permutation problems should be studied in the future to give a better understanding of the working mechanism of PSO for those problems.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 4104–4108, IEEE Press, Orlando, Fla, USA, October 1997.

[2] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '99)*, vol. 3, Washington, DC, USA, July 1999.

[3] A. Moraglio, C. D. Chio, and R. Poli, "Geometric particle swarm optimization," in *Proceedings of the 10th European Conference Genetic Programming (EuroGP '07)*, pp. 125–136, Valencia, Spain, April 2007.

[4] T. Gong and A. L. Tuson, "Binary particle swarm optimization: a forma analysis approach," in *Proceedings of 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, p. 172, ACM, London, UK, July 2007.

[5] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.

[6] M. Clerc, *Particle Swarm Optimization*, ISTE, London, UK, 2006.

[7] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, WA, Australia, November-December 1995.

[8] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE Congress on Evolutionary Computation (ICEC '98)*, pp. 69–73, IEEE Press, Anchorage, Alaska, USA, May 1998.

[9] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th Annual Conference on Evolutionary Programming (EP '98)*, pp. 591–600, San Diego, Calif, USA, March 1998.

[10] N. J. Radcliffe, "The algebra of genetic algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 10, no. 4, pp. 339–384, 1994.

[11] P. D. Surry, *A prescriptive formalism for constructing domain-specific evolutionary algorithm*, Ph.D. thesis, University of Edinburgh, Edinburgh, Scotland, UK, 1998.

[12] A. L. Tuson, *No optimization without representation: a knowledge based systems view of evolutionary/neighbourhood search optimization*, Ph.D. thesis, University of Edinburgh, Edinburgh, Scotland, UK, 1999.

[13] T. Gong and A. L. Tuson, "Formal descriptions of real parameter optimisation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 2119–2126, IEEE Press, Vancouver, BC, Canada, July 2006.

[14] C. Bierwirth, D. C. Mattfeld, and H. Kopfer, "On permutation representations for scheduling problems," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN '96)*, pp. 310–318, Springer, Berlin, Germany, September 1996.

[15] T. Gong and A. L. Tuson, "Enhanced forma analysis of permutation problems," in *Proceedings of 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pp. 923–930, ACM, London, UK, July 2007.

[16] T. Gong and A. L. Tuson, "Differential evolution for binary encoding," in *Proceedings of the 11th Online World Conference on Soft Computing in Industrial Applications (WSC '06)*, A. Saad, E. Avineri, K. Dahal, M. Sarfraz, and R. Roy, Eds., vol. 39 of *Advances in Soft Computing*, Springer, September-October 2006.

[17] E. P. K. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, UK, 1993.

[18] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics, Bristol, UK, 2000.

[19] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Proceedings of the 2nd International Conference on Genetic Algorithms on Genetic Algorithms and Their Application (ICGA '87)*, J. J. Grefenstette, Ed., Lawrence Erlbaum Associates, Cambridge, Mass, USA, July 1987.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Mass, USA, 2nd edition, 2001.

[21] M. DeLeon, "A study of sufficient conditions for hamiltonian cycles," Tech. Rep., Department of Mathematics and Computer Science, Seton Hall University, South Orange, NJ, USA, 2000.

[22] T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, and D. Whitley, "A comparison of genetic sequencing operators," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91)*, R. Belew and L. Booker, Eds., Morgan Kaufmann, San Diego, Calif, USA, July 1991.

[23] L. Steinberg, "The backboard wiring problem: a placement algorithm," *SIAM Rview*, vol. 3, no. 1, pp. 37–50, 1961.

[24] A. Elshafei, "Hospital layout as a quadratic assignment problem," *Operational Research Quarterly*, vol. 28, no. 1, pp. 167–179, 1977.

[25] A. Geoffrion and G. Graves, "Scheduling parallel production lines with changeover costs: practical application of a quadratic assignment/lp approach," *Operations Research*, vol. 24, no. 4, pp. 595–610, 1976.

[26] T. Stützle and M. Dorigo, "ACO algorithms for the quadratic assignment problem," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds., pp. 33–50, McGraw-Hill, London, UK, 1999.

[27] R. Battiti and G. Tecchiolli, "The reactive tabu search," *ORSA Journal on Computing*, vol. 2, no. 6, pp. 126–140, 1994.

[28] R. E. Burkard and F. Rendl, "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *European Journal of Operational Research*, vol. 17, no. 2, pp. 169–174, 1984.

[29] R. E. Burkard, S. Karisch, and F. Rendl, "QAPLIB-A quadratic assignment problem library," *European Journal of Operational Research*, vol. 55, no. 1, pp. 115–119, 1991.

[30] E. D. Taillard, "Comparison of iterative searches for the quadratic assignment problem," *Location Science*, vol. 3, no. 2, pp. 87–105, 1994.

[31] M.-H. Lim, Y. Yuan, and S. Omatu, "Extensive testing of a hybrid genetic algorithm for solving quadratic assignment problems," *Computational Optimization and Applications*, vol. 23, no. 1, pp. 47–64, 2002.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 126970, 9 pages
doi:10.1155/2008/126970

Research Article

A Strongly Interacting Dynamic Particle Swarm Optimization Method

S. Kok and J. A. Snyman

Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria 0002, South Africa

Correspondence should be addressed to S. Kok, schalk.kok@up.ac.za

Received 24 July 2007; Accepted 18 December 2007

Recommended by Riccardo Poli

A novel dynamic interacting particle swarm optimization algorithm (DYN-PSO) is proposed. The algorithm can be considered to be the synthesis of two established trajectory methods for unconstrained minimization. In the new method, the minimization of a function is achieved through the dynamic motion of a strongly interacting particle swarm, where each particle in the swarm is simultaneously attracted by all other particles located at positions of lower function value. The force of attraction experienced by a particle at higher function value due to a particle at a lower function value is equal to the difference between the respective function-values divided by their stochastically perturbed position difference. The resultant motion of the particles under the influence of the attracting forces is computed by solving the associated equations of motion numerically. An energy dissipation strategy is applied to each particle. The specific chosen force law and the dissipation strategy result in the rapid collapse (convergence) of the swarm to a stationary point. Numerical results show that, in comparison to the standard particle swarm algorithm, the proposed DYN-PSO algorithm is promising.

1. INTRODUCTION

A new direct search method using only function values is proposed for finding a local minimizer $\mathbf{x}^*$ with associated function value f^* of a real valued function $f(\mathbf{x})$, $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in R^n$. The method proposed may be considered as the synthesis of two unconventional trajectory methods for the unconstrained minimization of a multivariable function. The first is the *dynamic* method of Snyman [1, 2], and the second method is the particle swarm optimization (PSO) method of Eberhardt and Kennedy [3]. In the dynamic method, also known as the leap-frog algorithm, the minimum of the function is sought by considering the dynamic motion of a single particle of unit mass in an n-dimensional force field, where the potential energy of the particle is represented by the function to be minimized. In the computation of the numerical trajectory (by means of the leap-frog integration scheme of Greenspan [4]), an interfering strategy is applied to the motion of the particle by extracting kinetic energy whenever it moves "uphill" along its trajectory. In this way, the particle is forced to converge to a local minimum. This method requires the availability of the gradient vector of the function, denoted ∇f, the negative of which represents the force acting on the particle. The particle's acceleration is therefore proportional to $-\nabla f$. This is in contrast to classical gradient-based optimization, where position updates (or parts thereof) are proportional to $-\nabla f$. In the PSO method, the motion of a swarm of loosely interacting particles is considered. In this method, each particle is attracted to the best location (lowest function value position) along its path, as well as to the globally overall best position over all the particle trajectories to date. This method requires no gradient information and may therefore be considered a direct search method.

In the new method proposed here, the minimization of a function is achieved through the dynamic motion of a *strongly interacting* particle swarm, where *each particle* in the swarm is *simultaneously* attracted by all other particles located at positions of lower function value. The specific force law for the interaction between the individual particles within the swarm dictates that the force of attraction experienced by a particle at higher function value position due to a particle at a lower function value position be equal to the difference between the respective function values divided by their distance of separation. The resultant motion of the particles under the influence of the attracting forces is computed by solving the associated equations of motion numerically by using again the leap-frog numerical integration scheme. An energy dissipation strategy, similar to that used in the original dynamic method, is applied to each particle by extracting kinetic energy from a particle whenever it moves "uphill". The specific chosen force law and the dissipation strategy result in the rapid collapse (convergence) of the swarm to a stationary point. To prevent the collapse of the swarm to a minimum in a subspace of R^n, the computed components of the respective attracting forces are individually stochastically perturbed in computing the respective particle trajectories. Because of the strong interaction, the resulting algorithm converges rapidly to a local minimum, as does the original dynamic algorithm, but now without the need of explicit gradient information. Note that the method proposed here is not similar to the method proposed by Engelbrecht [5], where the position of *only* the global best particle is adjusted using the dynamic method, with the remainder of the swarm using a standard PSO algorithm.

The proposed dynamic particle swarm optimization (DYN-PSO) algorithm promises to be an extremely reliable, robust, and easy-to-use method. As a preliminary test, it was decided to evaluate its performance against that of both trajectory methods it was constructed from, namely, a standard PSO algorithm [6] and the dynamic method of Snyman. The choice of a standard PSO algorithm is a deliberate one. The intention with this paper is to determine whether or not DYN-PSO deserves further investigation, not to suggest that this is a superior algorithm. A comparison to a standard PSO rather than a highly refined PSO variant serves this purpose better.

Before proceeding, the constrained molecular dynamics PSO by Poli and Stephens [7] deserves to be mentioned. In essence, their method is similar is spirit, but the details differ. They embed the cost function (the height above the search space) as an artificial coordinate, and then constrain the particles "to be on the fitness landscape" via an equality constraint. Their general formulation allows for different types of forces, one of which is a gravity-like force. In our implementation, this gravity-like force is the only external force. Poli and Stephens also consider particle-interaction forces. In particular, they discuss the case of particles connected by springs and energy dissipation that occur via friction/viscous damping. They then also numerically integrate the resulting differential equation, but elect to use the forward Euler method. Their method still requires cost function gradients, as well as second-order derivatives, which they approximate numerically. Limited numerical testing was performed.

2. DESCRIPTION OF THE DYNAMIC-PSO METHOD

2.1. Computation of particle trajectories

The DYN-PSO method is started by generating at time $t = 0$, a swarm of np particles, each of unit mass, with random initial positions denoted by $\mathbf{x}\{i\}(0) = \mathbf{x}^0\{i\}$, $i = 1, 2, \ldots, \text{np}$, within the region ("box") of interest in R^n. Initially, at $t = 0$, these particles all have zero velocities, that is, $\mathbf{v}\{i\}(0) = \mathbf{v}^0\{i\} = \mathbf{0}$, $i = 1, 2, \ldots, \text{np}$. We now postulate that at time t each particle i experiences a force $\mathbf{a}\{i\}(t)$, which is to be a function of the positions $\mathbf{x}\{j\}(t)$ and corresponding function values $f(\mathbf{x}\{j\}(t))$, $j = 1, 2, \ldots, \text{np}$ of all the particles at time t. The explicit analytical form of the force law giving $\mathbf{a}\{i\}(t)$ will be discussed in the next subsection. Thus, the trajectories $\mathbf{x}\{i\}(t)$ of the particles are given by the solution to the system of *initial value problems*

$$\ddot{\mathbf{x}}\{i\}(t) = \mathbf{a}\{i\}(t) \tag{1}$$

with initial conditions

$$\begin{aligned} \mathbf{x}\{i\}(0) &= \mathbf{x}^0\{i\}, \\ \dot{\mathbf{x}}\{i\}(0) &= \mathbf{v}\{i\}(0) = \mathbf{v}^0\{i\} = \mathbf{0}; \end{aligned} \tag{2}$$

for $i = 1, 2, \ldots, \text{np}$.

In practice, these equations are solved numerically by discretizing the time interval into time steps δ, and computing for $i = 1, 2, \ldots, \text{np}$ approximations $\mathbf{x}^k\{i\}$ to $\mathbf{x}\{i\}(t_k)$ and $\mathbf{v}^k\{i\}$ to $\mathbf{v}\{i\}(t_k)$ at discrete time mesh points $t_k = k\delta$, $k = 0, 1, 2, \ldots$, by some suitable numerical integration scheme. Here, we use the simple leap-frog numerical integration scheme of Greenspan [4].

Given $\mathbf{x}^0\{i\}$ and $\mathbf{v}^0\{i\}$ for $i = 1, 2, \ldots, \text{np}$, then for iterations $k = 0, 1, 2, \ldots$, compute for $i = 1, 2, \ldots, \text{np}$:

$$\begin{aligned} \mathbf{v}^{k+1}\{i\} &= \mathbf{v}^k\{i\} + \mathbf{a}^k\{i\}\delta, \\ \mathbf{x}^{k+1}\{i\} &= \mathbf{x}^k\{i\} + \mathbf{v}^{k+1}\{i\}\delta, \end{aligned} \tag{3}$$

where $\mathbf{a}^k\{i\}$ denotes the resultant force on particle i due to the individual forces of all the other particles at respective positions $\mathbf{x}^k\{j\}$, $j = 1, 2, \ldots, \text{np}$, $j \neq i$ (see next subsection).

This scheme has been found to be stable for sufficiently small time steps δ. It is also approximately energy conserving in the absence of energy dissipating forces, and was successfully used in the original single-particle dynamic method of Snyman [1].

2.2. The interacting force law

In the original dynamic method of Snyman [1, 2], a single particle is considered. The force $\mathbf{a}$ acting on this particle is equal to the negative of the function gradient. Since we propose a direct search method, we no longer have gradient information available. Rather, we will use information available in the swarm to generate the particle forces.

We now postulate that at iteration k each particle i in the swarm is simultaneously attracted by all other particles located at *current* computed positions of lower function value. No force is exerted on particle i by particles at higher function value positions. The explicit force law that we assume here dictates that the force of attraction experienced by a particle at higher function value due to a particle at a lower function value position is equal to the difference between the respective function values, divided by their distance of separation.

If the particle positions are randomly distributed, this force law can be viewed as a coarse finite difference computation, with the finite difference step equal to the distance of separation. As the particle positions become biased, the analogy between this proposed force law and a finite difference computation breaks down. This breakdown occurs since multiple forces in the same direction add up, opposed to standard finite difference perturbations which form an orthogonal set. Even considering generalized finite differences [8], the gradient components due to multiple perturbations in the same direction are averaged, rather than added together as in our proposed force law. A stochastic element is introduced into the proposed force law by randomly perturbing the direction of action of the attracting force. More explicitly and precisely, at each iteration k the resultant force exerted on particle i at *current* position $\mathbf{x}^k\{i\}$ with function value $f(\mathbf{x}^k\{i\})$ by the other $\text{np}-1$ particles at current positions $\mathbf{x}^k\{j\}$, $j=1,2,\ldots,\text{np}$, $j\neq i$ with corresponding function values $f(\mathbf{x}^k\{j\})$ is given by the force $\mathbf{a}^k\{i\}$ with components

$$a_m^k\{i\} = \frac{\sum_{j=1,\,j\neq i}^{\text{np}}\left[x_m^k\{j\}-x_m^k\{i\}\right]c_j 2(\text{rd})}{\left\|\mathbf{x}^k\{j\}-\mathbf{x}^k\{i\}\right\|} \tag{4}$$

for $m=1,2,\ldots,n$. Here $c_j=\max[0, f(\mathbf{x}^k\{i\})-f(\mathbf{x}^k\{j\})]$ and rd is, for each j, an independent random number in [0, 1]. Note that the multiplication of a uniform random number in [0,1] by 2 results in a scheme that on average assigns unit weight to each of the computed components. If this stochastic element is omitted, the proposed algorithm suffers from premature convergence due to collapse of the swarm to a subspace in R^n. Similar observations have been made for the linear PSO [9].

This proposed force law is in contrast to the standard PSO, where particles are attracted to historic positions (personal and global bests) rather than the current position of other particles. In addition, this force law automatically results in a dynamic neighborhood, where it is possible that completely different particles exert forces on particle i in iteration $k+1$, as compared to iteration k. In a given iteration k, the complete range of interaction is also covered. One extreme is the particle with the current worst function value, which experiences forces from all other particles. The other extreme is the particle with the current best function value, which experiences no force and travels at constant velocity.

2.3. Energy dissipation strategy

In computing the trajectories of the particles $\mathbf{x}^k\{i\}$, $i=1,2,\ldots,\text{np}$, for $k=0,1,2,\ldots$, the function values $f_k(i)=f(\mathbf{x}^k\{i\})$ at $\mathbf{x}^k\{i\}$ are monitored at each iteration k so that the best (lowest) function value $f_b\{i\}$ and the corresponding best position $\mathbf{x}^b\{i\}$ along each trajectory i are recorded. The current overall globally best function value $f_g=\min_i(f_b\{i\})$ and the corresponding position $\mathbf{x}^g$ are also recorded.

The following energy dissipation strategy is now applied to ensure local descent of a particle and the overall collapse of the swarm to a local minimum. Whenever a particle i moves "uphill" at iteration k, that is, when $f_{k+1}\{i\}>f_k\{i\}$, then set $\mathbf{x}^{k+1}\{i\}:=[2\mathbf{x}^k\{i\}+\mathbf{x}^b\{i\}+\mathbf{x}^{k+1}\{i\}]/4$ and set $\mathbf{v}^{k+1}\{i\}:=[\mathbf{v}^{k+1}\{i\}+\mathbf{v}^k\{i\}]/4$. Notice that the current velocity $\mathbf{v}^{k+1}\{i\}$ is recomputed to be half the average velocity over the past two time steps. This interference will normally result in a decrease in kinetic energy and together with the "backward" adjustment of the position $\mathbf{x}^{k+1}\{i\}$ will initiate controlled motion of the particle towards a position of lower function value. The trajectory of particle i is assumed to have converged if the relative function value difference from iteration k to $k+1$, that is, $|f_{k+1}\{i\}-f_k\{i\}|/(1+|f_{k+1}\{i\}|)$, is less than some prescribed tolerance ε. The computation of the trajectories are continued until a sufficient number (npc) of the particles have converged to a stationary point. In practice, we choose $\text{npc}=\min[n,\text{np}]$.

A formal presentation of the basic DYN-PSO algorithm is presented in Algorithm 1. For the sake of clarity and simplicity of presentation of the algorithm, the function value computation and monitoring procedure, and the recording of best local $f_b\{i\}$ and global f_g function values and corresponding best local and global positions, $\mathbf{x}^b\{i\}$ and $\mathbf{x}^g$ are not explicitly listed, but are implicitly assumed to be done in the execution of Algorithm 1. Also the computation of the forces $\mathbf{a}^k\{i\}$, $i=1,2,\ldots,\text{np}$, according to (4), is assumed to have been done as the need for the $\mathbf{a}^k\{i\}$ arises in Algorithm 1.

2.4. Selection of suitable integration time step

An outstanding matter is the selection of an appropriate time step δ to be used in the leap-frog integration scheme. This value can be chosen arbitrarily, but if chosen too large may result in individually erratic and unstable trajectories that fail to converge because of very large zigzagging steps being taken in space. On the other hand, if δ is too small, the steps in space will be correspondingly small and the collapse of the swarm may be very slow, requiring an excessively large number of iterations for convergence.

Many different schemes may be proposed to automatically select and control the time step so that acceptable convergence rates are obtained. Here, we firstly select an initial time step which guarantees sufficiently large trajectory steps in space for all the particles. Secondly, after the trajectories are initiated, for each iteration k and for each particle i, the magnitude of the actual step taken in space is monitored; and if larger than some specified step limit, the time step associated with the particular trajectory is reduced, that is, we now allow for different time steps $\delta\{i\}$, $i=1,2,\ldots,\text{np}$, for the different particles. The details of these additional automatic time step control procedures follow below.

In Step 1 of Algorithm 1, after the generation of the np random particles, compute the average magnitude $\overline{a}$ of the forces acting on the particles $\overline{a}=\sum_i\|\mathbf{a}\{i\}\|/\text{np}$. An associated average computed initial step size follows from the leap-frog scheme as $\Delta x=\overline{a}\delta^2$. Requiring initially, on average, a step size of $\Delta x=D$, where D is the diameter of the initial variable "box", we initially select as sufficiently large initial time steps $\delta\{i\}=\delta=\sqrt{D/\overline{a}}$, $i=1,2,\ldots,\text{np}$, which is now inserted in Step 1 just after the generation of the initial random particle positions. However, for a particular particle, this initial choice for the time step may still be too large and result

```
Given the following:
  (i) function $f(\mathbf{x})$, $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in R^n$;
  (ii) np = number of particles;
  (iii) $\mathbf{y}$ = central point of region ("box") of interest;
  (iv) range($i$) = range of $i$th dimension of variable "box";
  (v) $\delta$ = integration time step;
  (vi) $\varepsilon$ = tolerance for convergence of a trajectory on
       function value;
  (vii) npc = number of particle trajectories required to
       converge before termination (default: npc = min[$n$,
       np]);
  (viii) max = maximum number of iterations allowed,
then perform the following steps.
Step 1.
  (i) For $i = 1, 2, \ldots,$ np, generate random particle
      starting positions $\mathbf{x}^0\{i\}$ within the specified box
      with components: $x_j^0\{i\} = y_j +$ range($j$)(rd $- 1/2$);
      $j = 1, 2, \ldots, n$, where rd is an independent random
      number in the interval $[0, 1]$;
  (ii) set initial velocities equal to zero: $\mathbf{v}^0\{i\} = \mathbf{0}$ and set
      iteration number $k := 0$; and convergence counter
      ic := 0.
Step 2.
(i) Compute trajectory step $k$ to $k + 1$ for each particle
    $i = 1, 2, \ldots,$ np, using the leap-frog integration scheme:
    for $i = 1, 2, \ldots,$ np,
        $\mathbf{v}^{k+1}\{i\} := \mathbf{v}^k\{i\} + \mathbf{a}^k\{i\}\delta$,
        $\mathbf{x}^{k+1}\{i\} := \mathbf{x}^k\{i\} + \mathbf{v}^{k+1}\{i\}\delta$,
    end for
Step 3.
(i) for $i = 1, 2, \ldots,$ np:
        if $f_{k+1}\{i\} > f_k\{i\}$, then
            $\mathbf{x}^{k+1}\{i\} := [2\mathbf{x}^k\{i\} + \mathbf{x}^b\{i\} + \mathbf{x}^{k+1}\{i\}]/4$,
            $\mathbf{v}^{k+1}\{i\} := [\mathbf{v}^{k+1}\{i\} + \mathbf{v}^k\{i\}]/4$
        end if
        if $|f_{k+1}\{i\} - f_k\{i\}|/(1 + |f_{k+1}\{i\}|) < \varepsilon$ and $k > 0$, then
            set ic = ic + 1;
        if ic = npc, then set $\mathbf{x}^* := \mathbf{x}^g$ and $f^* := f_g$ and stop.
        if $k$ = max, then stop.
        end for
(ii) set $k := k + 1$; ic := 0 and go to Step 2.
```

Algorithm 1: Basic DYN-PSO algorithm.

in erratic behavior along the computed trajectory. Thus, as a further control measure, the magnitude of the step taken by each particle is monitored at each iteration, and if it exceeds a specified limit $x_{\lim}$, the current velocity is scaled down by a factor $x_{\lim}/\|\mathbf{x}^{k+1}\{i\} - \mathbf{x}^k\{i\}\|$, the particle's time step for subsequent steps is reduced by a factor α, and the step to $\mathbf{x}^{k+1}\{i\}$ is recomputed using the rescaled velocity and new time step. More specifically, we introduce at the end of Step 2 of Algorithm 1, for each iteration k and for each particle i, the additional procedure given in Algorithm 2.

In practice, good choices for the additional parameters introduced here are $x_{\lim} = D/2$ and $\alpha = 0.5$. These values are used in the numerical tests that are reported here.

```
if $\|\mathbf{x}^{k+1}\{i\} - \mathbf{x}^k\{i\}\| > x_{\lim}$ then
    $\delta\{i\} := \alpha\delta\{i\}$
    $\mathbf{v}^{k+1}\{i\} := x_{\lim}\mathbf{v}^{k+1}\{i\}/\|\mathbf{x}^{k+1}\{i\} - \mathbf{x}^k\{i\}\|$
    $\mathbf{x}^{k+1}\{i\} := \mathbf{x}^k\{i\} + \mathbf{v}^{k+1}\delta\{i\}$
end if
```

Algorithm 2: Adjustment of time step δ.

One complication of the above strategy arises if the nature of the cost function changes dramatically as the search proceeds. The initial time steps are appropriate during the initial stages of the search. However, the average force acting on the particles often decreases over time. Hence, the average particle step size also decreases over time. This seems appropriate for swarm convergence, but the step size decrease may be excessive if the average magnitudes of the forces computed by (4) decrease dramatically. This typically happens in the neighborhood of a local minimum of highly nonlinear functions. To overcome this drawback, we simply recompute the time steps $\delta\{i\} = \delta = \sqrt{D/\overline{a}}$, $i = 1, 2, \ldots,$ np, whenever appropriate, using the recomputed average force $\overline{a}$. For all the results presented in this paper, the time steps are recomputed every 100 iterations.

3. NUMERICAL PERFORMANCE OF THE DYN-PSO METHOD

3.1. Illustrative two-dimensional trajectories

To illustrate the mechanics of the DYN-PSO algorithm, we compute the trajectories for $f(\mathbf{x}) = x_1^2 + 2x_2^2$ with 3 particles, that is, the case $n = 2$ with np = 3. As starting points, we select the vertices $(40, 40)$; $(-40, 0)$; $(40, -40)$. Figure 1 depicts the three computed trajectories (using diameter $D = 100$) up to the 30th iteration at which point $f_g = 9.48 \times 10^{-4}$, with $x_1^g = -0.0201$; $x_2^g = 0.0165$.

3.2. Choice of number of particles

Throughout the numerical experiments done here, the number of particle trajectories required to converge before termination of the algorithm is taken as npc = min[n, np]. It is now required to obtain an indication of what is a good or an optimum choice for the number of particles np to be used for "normal" problems where one expects a single unique global minimum in the region of interest. To get an indication of what it should be, some experiments were performed (with function value tolerance $\varepsilon = 10^{-8}$) on the extended homogeneous quadratic test function (see list of test problems) for $n = 10$ and $n = 20$, using different values for np and determining the average number of functions evaluations (over 100 independent runs) required for successful convergence in each case. The variation of the number of function evaluations against np is shown in Figure 2. Note that for $n = 20$, the number of function evaluations appears to be almost insensitive to np over the range np = 15 to np = 25, with a

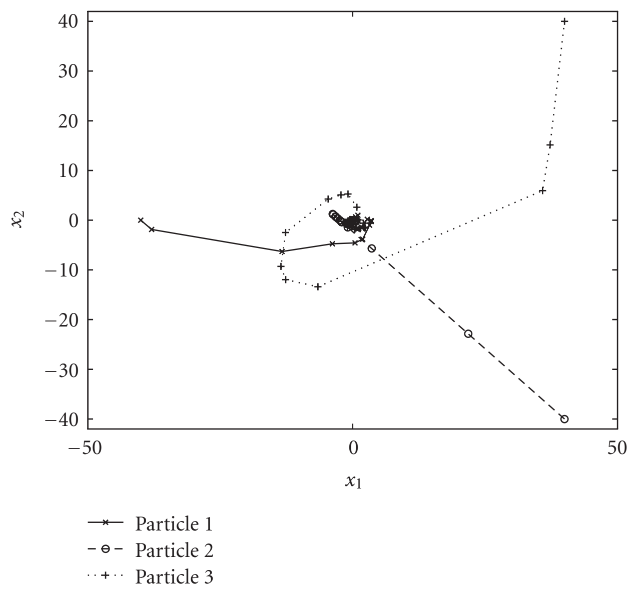


FIGURE 1: Example of dynamic PSO particle trajectories for the 2D function $f(x_1, x_2) = x_1^2 + 2x_2^2$.

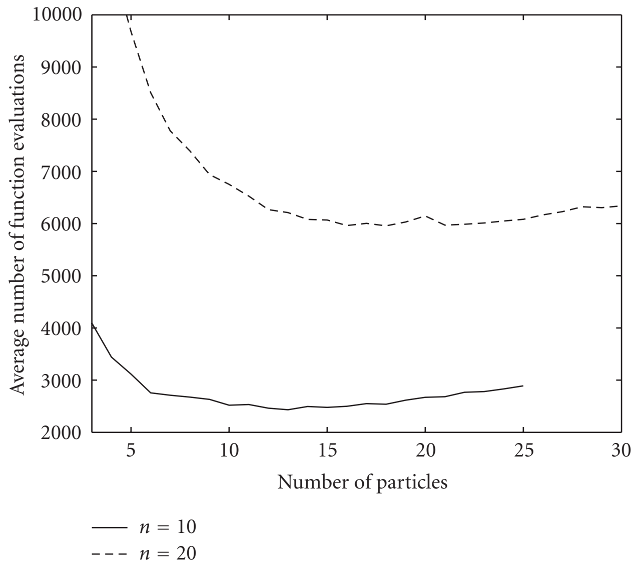


FIGURE 2: Average number of function evaluations at convergence versus number of particles.

variation of less than 4 percent being recorded. The corresponding variation of average function value at convergence with np is depicted in Figure 3. The results appear to indicate that in general a choice of np $= n + 1$ (i.e., with the positions of the particles defining the vertices of an n-dimensional simplex in R^n) is probably a good one. Consequently, the choice np $= n + 1$ is used throughout the experiments performed in the next subsection. Since this guideline is based on a very narrow test, some future effort should be directed towards improved guidelines to determine the number of particles. It is anticipated that highly multimodal problems might benefit from an increased number of particles.

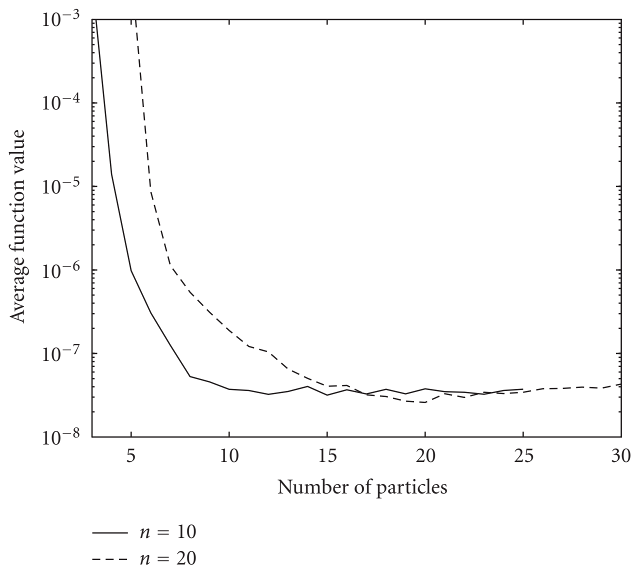


FIGURE 3: Average function value at convergence versus number of particles.

3.3. Performance on a set of test functions

The newly proposed DYN-PSO algorithm is tested on the following eight test problems.

(i) Homogeneous quadratic (unimodal)

$$f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2 \tag{5}$$

subject to $-5 < x_i < 5$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

(ii) Oren's power function (unimodal) [1]

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} i x_i^2\right)^2 \tag{6}$$

subject to $-10 < x_i < 10$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

(iii) Extended Rosenbrock (multimodal) [10]

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \tag{7}$$

subject to $-2.048 < x_i < 2.048$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [1, 1, \ldots, 1]^T$.

(iv) Neumaier 3 (multimodal) [10]

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \tag{8}$$

subject to $-n^2 < x_i < n^2$ for $i = 1, 2, \ldots, n$; $f^* = -n(n + 4)(n - 1)/6$, at $x_i^* = i(n + 1 - i)$.

(v) Extended Manevich (unimodal) [11]

$$f(\mathbf{x}) = \sum_{i=1}^{n} \frac{(1 - x_i)^2}{2^{i-1}} \tag{9}$$

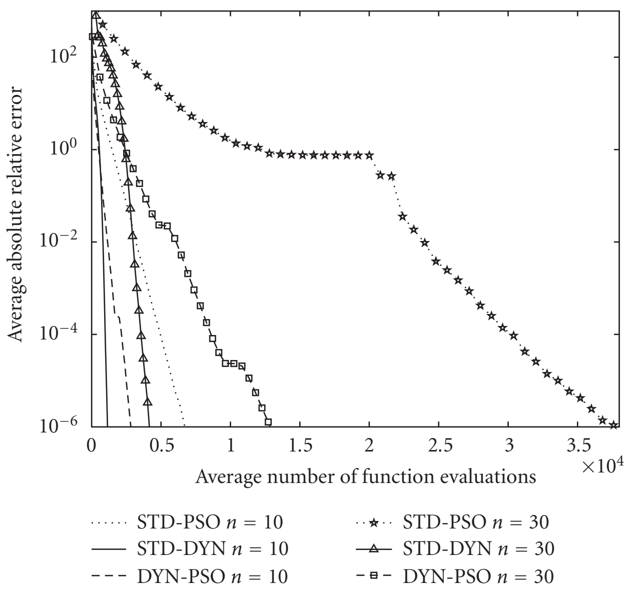


Figure 4: Convergence history on the homogeneous quadratic test function.

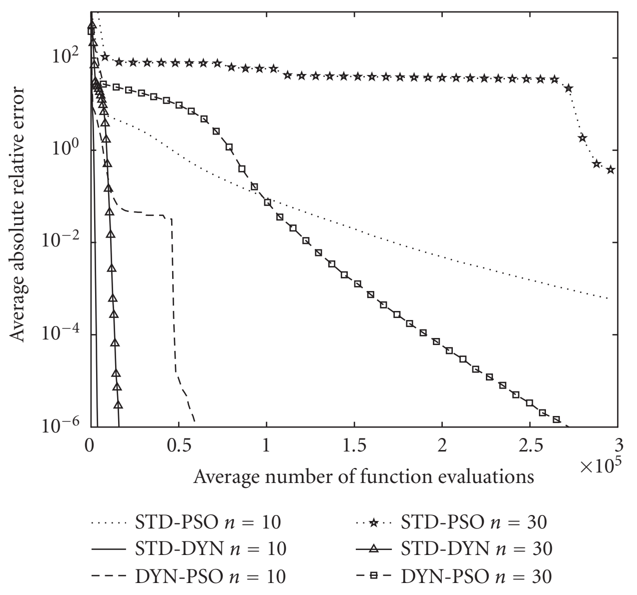


Figure 6: Convergence history on the Rosenbrock test function.

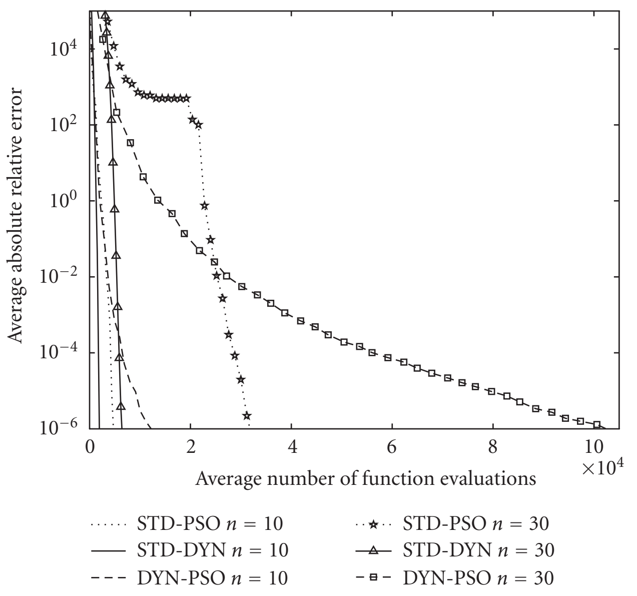


Figure 5: Convergence history on Oren's power test function.

subject to $-10 < x_i < 10$ for $i = 1, 2, \dots, n$; $f^* = 0$, at $\mathbf{x}^* = [1, 1, \dots, 1]^T$.

(vi) Zakharov (unimodal) [12]

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4 \tag{10}$$

subject to $-5 < x_i < 10$ for $i = 1, 2, \dots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \dots, 0]^T$.

(vii) Griewank (multimodal) [10]

$$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{11}$$

subject to $-600 < x_i < 600$ for $i = 1, 2, \dots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \dots, 0]^T$.

(viii) Rastrigin (multimodal) [10]

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \tag{12}$$

subject to $-5.12 < x_i < 5.12$ for $i = 1, 2, \dots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \dots, 0]^T$.

Each test problem is solved 100 times for problem dimension $n = 10$ and 30. The DYN-PSO algorithm is compared to the dynamic method of Snyman [1, 2] (STD-DYN) and a standard PSO algorithm (STD-PSO) (http://www.particleswarm.info/Standard_PSO_2006.c), making use of settings proposed by Eberhart and Shi [6] ($c_1 = c_2 = 1.49445$, constant inertia weight of 0.729 and swarm size of 20). The gradients required by the dynamic method are computed using a forward difference method, using a perturbation size of 10^{-6}. These additional function evaluations are taken into consideration in the results of STD-DYN presented below. The convergence histories of the STD-DYN, DYN-PSO, and STD-PSO algorithms are depicted in Figures 4 to 11, for the eight test functions. In these graphs, at each iteration $k = 0, 1, 2, \dots$, the absolute relative error

$$e_r = \frac{|f_g - f^*|}{1 + |f^*|}, \tag{13}$$

averaged over the 100 runs, is plotted against the corresponding average number of function evaluations.

Two common scenarios exist to terminate optimization algorithms. First, some convergence criteria are satisfied, which indicates that no substantial improvement is likely and search can terminate. Alternatively, a maximum number of function evaluations may be specified. For the test problems considered here, both these scenarios occur. The many local

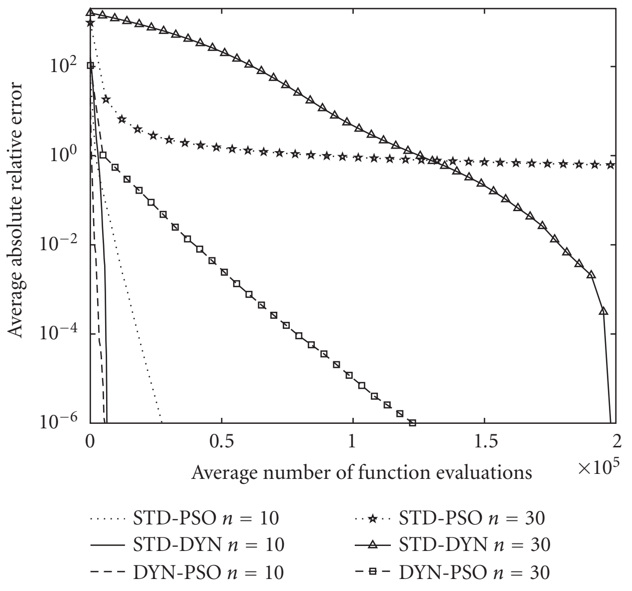


FIGURE 7: Convergence history on the Neumaier 3 test function.

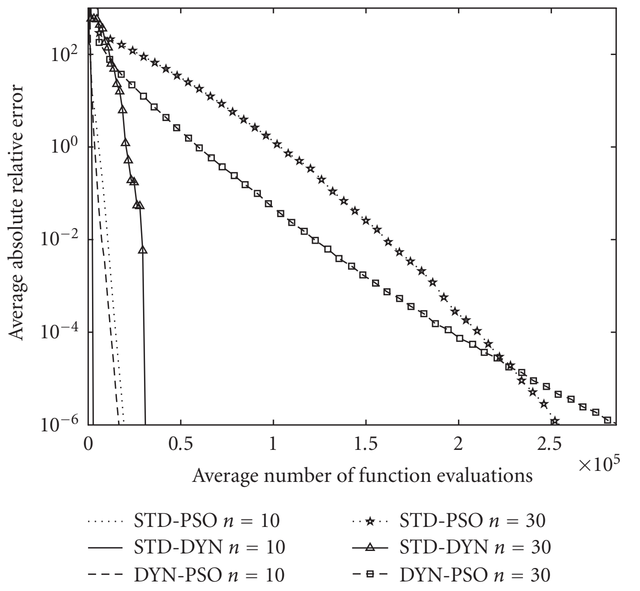


FIGURE 9: Convergence history on the Zakharov test function.

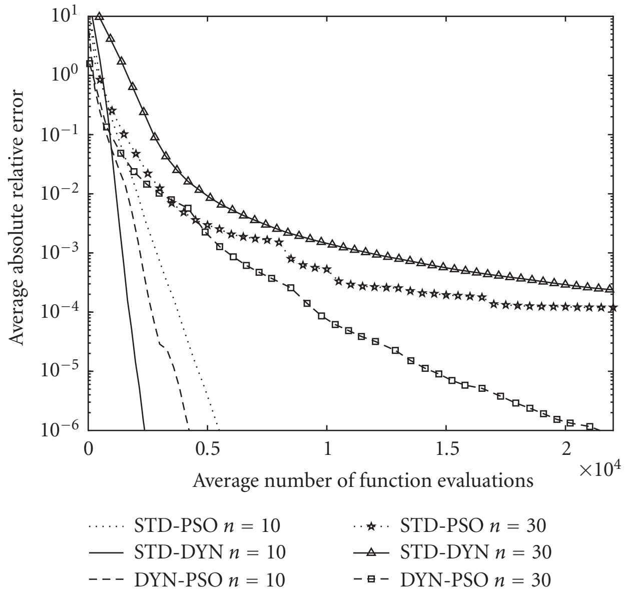


FIGURE 8: Convergence history on the Manevich test function.

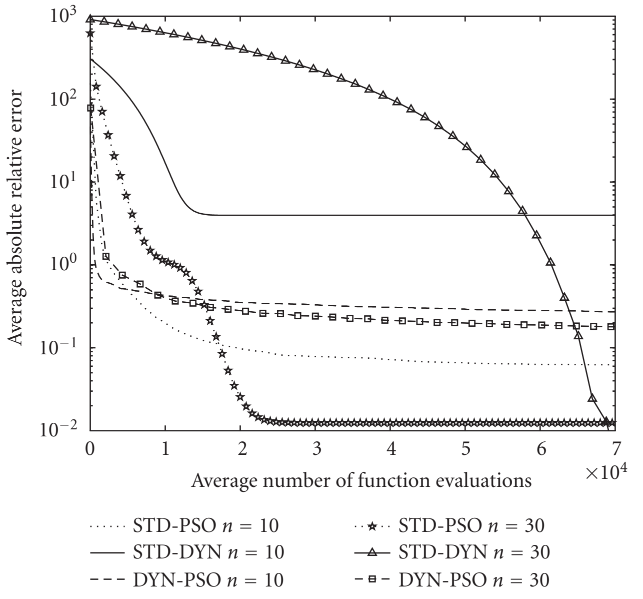


FIGURE 10: Convergence history on the Griewank test function.

minima present in the multimodal Griewank and Rastrigin test functions prevent the majority of particle trajectories to converge, hence these problems are terminated based on a maximum allowable number of function evaluations. The proposed convergence criteria work well for all the remaining test functions, where we used a convergence tolerance of $\varepsilon = 10^{-8}$.

The STD-DYN and DYN-PSO algorithms always locate the global minimum for the homogeneous quadratic, Oren's power function, Neumaier 3, Manevich and Zakharov problems. The STD PSO fails to do so for the 30D Neumaier problem. Also note that out of these 5 test problems, the STD-DYN algorithm is most efficient on the homogeneous quadratic, Oren's power function, and the Zakharov function, all of which are unimodal. The DYN-PSO algorithm performs best on the Neumaier 3 problem and the 30D Manevich function. STD PSO is only more efficient than the DYN-PSO algorithm for Oren's power function.

In the case of the Rosenbrock test function, the DYN-PSO algorithm does not always locate the global minimum. In those cases in which the global minimum is not found, the algorithm converges to the only other local minimum, as reported by Shang and Qiu [13]. In the Rosenbrock experiments, 89 and 96 of the 100 runs converged to the global minimum, for $n = 10$ and 30, respectively. The STD-DYN method locates the global minimum 83 and 88 times, for

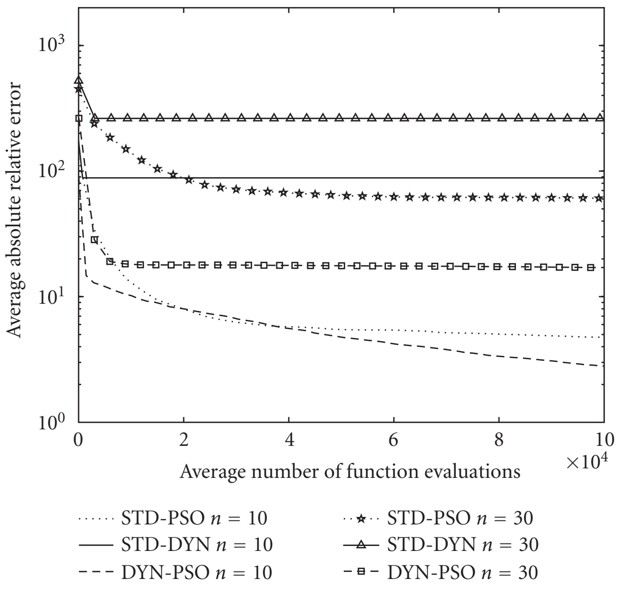


Figure 11: Convergence history on the Rastrigin test function.

$n = 10$ and 30, respectively. Note however that the number of function evaluations is more than a factor 10 less compared to the DYN-PSO algorithm. In the case of the STD-PSO algorithm, the majority of runs do not converge to either the global or local minimum. The number of runs that has a global best less than 1 after 3×10^5 function evaluations is 98 and 4, respectively, for n=10 and 30. The absolute relative error in Figure 6 is computed by only averaging over those runs that converge to the global minimum (DYN PSO and STD DYN) or those that have a global best less than unity after 3×10^5 function evaluations (STD PSO).

Finally, the DYN-PSO algorithm never locates the global minimum for the multimodal Griewank and Rastrigin test functions. The dynamic method, which is the backbone of the DYN-PSO algorithm, was developed as a local minimizer. However, the energy dissipation strategy we opted in the DYN-PSO algorithm is unchanged from that of the original dynamic method. This strategy is quite severe on "uphill" moves and hence energy is lost quickly if many local minimizers are present, such as in the Griewank and Rastrigin test functions. Nevertheless, compared to the performance of the STD-DYN method, the DYN PSO is vastly superior on the Rastrigin test function. The STD-DYN method simply locates the first strong local minimum and cannot escape it. On the Griewank function, DYN PSO is superior to STD DYN for 10D, and vice versa for 30D. The STD PSO however far outperforms both STD DYN and DYN PSO for the Griewank function. However, in our experience, the performance of the DYN-PSO algorithm is still comparable to that of more traditional global optimization algorithms. The current algorithm does demonstrate the ability to escape some local minima, similar to the standard PSO algorithm. The performance on the 30D Rastrigin function is especially noteworthy, with a mean function value less than 20 after only 5 000 function evaluations. This is obtained with the standard settings proposed here, with no tuning of parameters to suit the problem.

In summary, the DYN-PSO algorithm seems to inherit the desired properties from both its ancestors. It can efficiently solve unimodal functions, sometimes even more efficiently than the gradient-based local minimizer it is based on, for example, the 30D Manevich problem. This is achieved without making use of gradient information. Therefore, the DYN-PSO method might even work for discontinuous problems. This efficient local character is blended with nonlocal behavior, where the swarm provides sufficient information to solve multimodal problems, illustrated best on the Rastrigin function.

4. CONCLUSIONS

We have proposed a novel dynamic interacting particle swarm optimization algorithm. The algorithm compares well to a standard PSO implementation, especially in terms of efficient solution of high-dimensional problems containing few local minimizers. Based on these promising results, the DYN-PSO algorithm deserves further development.

A number of outstanding issues remain. The importance of recomputing appropriate time-step sizes is already recognized, but more refined criteria should be developed. A guideline for the number of particles for efficient search is already proposed, but since it is based on a very narrow test, additional experiments are required. Also, the convergence criterion can be refined in order to also work for functions containing a very large number of local minimizers. The mechanism that probably requires the most attention is the energy dissipation scheme, which can be modified to increase the probability of convergence to the global minimum in the case of multiple local minima. This could be achieved by less aggressive energy dissipation during "uphill" moves, but will necessarily retard convergence.

The initial results indicate that the proposed DYN-PSO algorithm shows much promise as an alternative direct search method for solving large scale unconstrained optimization problems. The algorithm seems capable of solving unimodal problems economically, and it also has competitive performance on functions containing many local minima.

REFERENCES

[1] K. W. Kolodziej and J. Hjelm, *Local Positioning Systems: LBS Applications and Services*, CRC Press, Boca Raton, Fla, USA, 2006.

[2] A. LaMarca, Y. Chawathe, S. Consolvo, et al., "Place lab: device positioning using radio beacons in the wild," in *Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE '05)*, vol. 3468, pp. 116–133, Munich, Germany, May 2005.

[3] J. Letchner, D. Fox, and A. LaMarca, "Large-scale localization from wireless signal strength," in *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (AAAI '05)*, pp. 15–20, Pittsburgh, Pa, USA, July 2005.

[4] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, "Zero-configuration, robust indoor localization: theory and experimentation," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[5] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, vol. 2, pp. 775–784, Tel Aviv, Israel, March 2000.

[6] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for WLAN location fingerprinting," in *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS '04)*, pp. 14–23, Boston, Mass, USA, August 2004.

[7] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '01)*, vol. 5, pp. 2926–2931, San Antonio, Tex, USA, November 2001.

[8] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 1734–1743, San Francisco, Calif, USA, March-April 2003.

[9] R. Peng and M. L. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *Proceedings of the 3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '06)*, vol. 1, pp. 374–382, Reston, Va, USA, September 2006.

[10] R. L. Moses, D. Krishnamurthy, and R. M. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 4, pp. 348–358, 2003.

[11] D. Niculescu and B. Nath, "Position and orientation in ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 2, pp. 133–151, 2004.

[12] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[13] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed Ad-Hoc wireless sensor networks," in *Proceedings of USENIX Annual Technical Conference*, pp. 317–327, Monterey, Calif, USA, June 2002.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 861275, 15 pages
doi:10.1155/2008/861275

Research Article

The Generalized PSO: A New Door to PSO Evolution

J. L. Fernández Martínez and E. García Gonzalo

Department of Mathematics, University of Oviedo, C/Calvo Sotelo S/N, 33007 Oviedo, Spain

Correspondence should be addressed to J. L. Fernández Martínez, jlfm@uniovi.es

Received 20 July 2007; Revised 15 January 2008; Accepted 24 February 2008

Recommended by Riccardo Poli

A generalized form of the particle swarm optimization (PSO) algorithm is presented. Generalized PSO (GPSO) is derived from a continuous version of PSO adopting a time step different than the unit. Generalized continuous particle swarm optimizations are compared in terms of attenuation and oscillation. The deterministic and stochastic stability regions and their respective asymptotic velocities of convergence are analyzed as a function of the time step and the GPSO parameters. The sampling distribution of the GPSO algorithm helps to study the effect of stochasticity on the stability of trajectories. The stability regions for the second-, third-, and fourth-order moments depend on inertia, local, and global accelerations and the time step and are inside of the deterministic stability region for the same time step. We prove that stability regions are the same under stagnation and with a moving center of attraction. Properties of the second-order moments variance and covariance serve to propose some promising parameter sets. High variance and temporal uncorrelation improve the exploration task while solving ill-posed inverse problems. Finally, a comparison is made between PSO and GPSO by means of numerical experiments using well-known benchmark functions with two types of ill-posedness commonly found in inverse problems: the Rosenbrock and the "elongated" DeJong functions (global minimum located in a very flat area), and the Griewank function (global minimum surrounded by multiple minima). Numerical simulations support the results provided by theoretical analysis. Based on these results, two variants of Generalized PSO algorithm are proposed, improving the convergence and the exploration task while solving real applications of inverse problems.

1. INTRODUCTION

Particle swarm optimization (PSO) is an evolutionary computation technique [1] for optimization which is inspired by the social behavior of individuals in groups in nature. The particle swarm algorithm applied to optimization problems is very simple.

(1) Individuals, or particles, are represented by vectors whose length is the number of degrees of freedom of the optimization problem.

(2) To start, a population of particles is initialized with random positions ($\mathbf{x}_i^0$) and velocities ($\mathbf{v}_i^0$). A misfit or cost function is evaluated for each particle.

(3) As time advances, the positions and velocities of each particle are updated as a function of its own history of misfits, and the misfit of its neighbors. At time-step $k+1$, positions ($\mathbf{x}_i^{k+1}$) and velocities ($\mathbf{v}_i^{k+1}$) of the individuals are calculated as follows:

$$\begin{aligned}\mathbf{v}_i^{k+1} &= \omega\mathbf{v}_i^k + \phi_1\cdot 1\cdot(\mathbf{g}^k-\mathbf{x}_i^k) + \phi_2\cdot 1\cdot(\mathbf{l}_i^k-\mathbf{x}_i^k),\\ \mathbf{x}_i^{k+1} &= \mathbf{x}_i^k + \mathbf{v}_i^{k+1}\cdot 1\end{aligned} \tag{1}$$

with

$$\phi_1 = r_1 a_g,\quad \phi_2 = r_2 a_l,\quad r_1, r_2 \longrightarrow U(0,1),\quad w, a_l, a_g \in \mathbb{R}, \tag{2}$$

where $\mathbf{l}_i^k$ is the ith particle best position, $\mathbf{g}^k$ is the global best position found so far, ϕ_1, ϕ_2 are the global and local accelerations, and ω is a real constant, called inertia.

The real scalar 1 in (1) stands for the time step necessary to make this algorithm dimensionally correct, as corresponds to the relationship between trajectories, velocities, and accelerations. The random numbers, r_1 and r_2, affect the local and global acceleration terms, a_l and a_g, causing the particle trajectory to oscillate at each iteration around its center [2], named

$$\mathbf{o}_i^k = \frac{\phi_1 \mathbf{g}^k + \phi_2 \mathbf{l}_i^k}{\phi_1 + \phi_2}. \tag{3}$$

Thus, relative values of ϕ_1 and ϕ_2 affect the balance between local and global search. PSO algorithm has been also adapted to incorporate a different kind of topologies concerning the "informant group."

Convergence properties and PSO trajectories were analyzed, and helped to clarify some numerical aspects of the PSO algorithm; see [2–6]. Also, based on these analysis, some parameter selection strategies were proposed (see, e.g., [5]).

Stability analysis of the particle dynamics has also been done in presence of randomness by Kadirkamanathan et al. [7], using Lyapunov techniques. As result of this analysis, they found a sufficient condition on (ω, ϕ) to achieve convergence:

$$\left\{(\omega, \phi) : |\omega| < 1,\ w \neq 0,\ \phi < \frac{1 - 2|w| + w^2}{1 + w}\right\}. \tag{4}$$

Nevertheless, this condition is very restrictive, and this region does not include the best parameters found in the literature.

Physical analogy with a damped mass-spring oscillator was introduced by Brandstätter and Baumgartner [8] to optimize electrical engineering problems, nevertheless, this analogy was not completely exploited in their work. Recently, Fernández Martínez et al. [9] used this analogy to derive the PSO continuous model and to present a systematic study of PSO trajectories by means of stability analysis of the deterministic PSO difference equations involved and fixed point techniques. The PSO convergence can then be explained in terms of some combined criteria: trajectory attenuation, trajectory oscillation, and center attraction potential, explaining the success in achieving convergence of some parameter sets found in the literature.

Interdisciplinary approaches, from classical Newtonian mechanics to molecular dynamics theory, have been recently proposed by Mikki and Kishk [10] to study the thermodynamic behavior of PSO, providing new criterion to analyze the algorithm convergence. Also, stagnation analysis [11] and statistical approaches [12] were used to characterize the sampling distribution of PSO in presence of stochasticity and to provide criteria for the PSO parameter selection.

In this paper, we present a generalized form of the particle swarm optimization (PSO), which is derived from a continuous version of PSO. This idea has been partially presented in [9, 10, 13], but in this manuscript is fully developed.

(1) The deterministic and stochastic stability regions and the asymptotic velocities of convergence of the GPSO algorithm are analyzed as a function of the time step, the local and global accelerations, and the inertia value.

(2) Generalized and continuous PSO models are compared in terms of attenuation and oscillation properties as a function of the time step and the PSO parameters. As expected, GPSO properties tend to those of the continuous PSO model, as the time step goes to zero.

(3)The sampling distribution of the GPSO algorithm is also analyzed and helps to study the effect of stochasticity on the stability of trajectories. The stability region for the second-, third-, and fourth-order moments depends on inertia, local, and global accelerations and time discretization step. These regions are smaller in size than the corresponding deterministic stability region for the same time step. Regions increase in size as time step decreases. Also, for a given time step, the maximum size of the second-order stability region occurs when local and global accelerations are equal. Properties of the second-order moments—variance, covariance, variogram, and correlogram—are also explored with a moving center of attraction, assuming that $l(t)$ and $g(t)$ exhibit a deterministic behavior. This analysis serve to propose some promising parameter sets. High variance and temporal uncorrelation improve the exploration task needed to solve ill-posed inverse problems. A similar idea has been also proposed by Clerc [11] under stagnation.

(4) Finally, a comparison is made between PSO and the GPSO by means of numerical experiments using well-known benchmark functions supporting the theoretical results. Based on these results two variants of generalized PSO algorithm are proposed, improving the convergence and the exploration task while solving real applications in inverse problems modeling.

2. THE PSO DIFFERENCE EQUATIONS

In the PSO algorithm, the following vectorial difference equation is involved for each particle in the swarm:

$$\begin{gathered}\mathbf{x}_i^{k+1} + (\phi - \omega - 1)\mathbf{x}_i^k + \omega \mathbf{x}_i^{k-1} = \phi o_i^k = \phi_1 \mathbf{g}^k + \phi_2 \mathbf{l}_i^k,\\ \mathbf{x}_i^0 = \mathbf{x}_{i0},\\ \mathbf{x}_i^1 = \mathbf{x}_i^0 + w\mathbf{v}_{i0} + \phi(\mathbf{o}_i^0 - x_i^0),\end{gathered} \tag{5}$$

where $\phi = \phi_1 + \phi_2$ is a random variable with trapezoidal (or triangular if $a_l = a_g$) distribution, and i is the particle number.

Let us introduce the variables $\xi_i^k = \mathbf{x}_i^k - \mathbf{o}_i^k$, that is, particle's positions in iteration k are referred to their oscillation center $\mathbf{o}_i^k$ (3). Then, the following second-order difference equation is involved:

$$\begin{gathered}\xi_i^{k+1} + (\phi - \omega - 1)\xi_i^k + \omega \xi_i^{k-1} = \beta(\mathbf{o}), \quad k \in \mathbb{N},\\ \beta(\mathbf{o}) = \mathbf{o}_i^k - \mathbf{o}_i^{k+1} + w(\mathbf{o}_i^k - \mathbf{o}_i^{k-1}).\end{gathered} \tag{6}$$

When the oscillation center $\mathbf{o}_i^k$ stabilizes (stagnation case), then the PSO algorithm involves the following vectorial stochastic difference equation [6]:

$$\begin{gathered}\xi_i^{k+1} + (\phi - \omega - 1)\xi_i^k + \omega \xi_i^{k-1} = \mathbf{0}, \quad k \in \mathbb{N},\\ \xi_i^0 = \xi_{i0},\\ \xi_i^1 = (1 - \phi)\xi_{i0} + w\mathbf{v}_{i0}.\end{gathered} \tag{7}$$

In this model, trajectories are considered as random functions.

Let us introduce the PSO deterministic case which is modeled by difference equation (7) and ϕ is in this case is a real constant. As we will show on the stochastic stability analysis, this model describes the mean behavior of random trajectories, $E(\xi_i^{k+1})$, when $\phi = \overline{\phi} = (a_g + a_l)/2$ is in this case the average value of random variable ϕ.

The deterministic stability region of the PSO is the part of the space $(\omega, \overline{\phi})$, where the roots of the characteristic equation

$$\lambda^2 + (\overline{\phi} - \omega - 1)\lambda + w = 0 \tag{8}$$

are in the unit circle. This region turns to be [4]

$$S_D = \{(\omega, \overline{\phi}) : |\omega| < 1,\ 0 < \overline{\phi} < 2(w+1)\}. \tag{9}$$

The border line between complex and real roots of the characteristic equation (8) is the parabola

$$(\overline{\phi} - w - 1)^2 = 4w, \quad 0 \le w < 1. \tag{10}$$

Model (7) has been generalized, for any time t, to the so-called PSO-discrete functional model,

$$\begin{gathered}\xi(t+1) + (\overline{\phi} - \omega - 1)\xi(t) + \omega\xi(t-1) = 0, \quad t \in \mathbb{R}, \\ \xi(0) = \xi_0, \\ \xi(1) = \xi_0(1 - \overline{\phi}) + w\mathbf{v}_0\end{gathered} \tag{11}$$

that provides the continuous analytical function that overlaps the discrete PSO points, and allows us to perform a systematic study of the PSO trajectories under stagnation [9]. Four different zones have been differentiated (see Figure 1) showing a different behavior, depending on the character of the eigenvalues of the characteristic equation (8) associated to the PSO difference equation.

The PSO algorithm can also be written in terms of the absolute position and velocity $(\mathbf{x}^k, \mathbf{v}^k)$ in each iteration, as the stochastic dynamical system

$$\begin{pmatrix}\mathbf{x}^{k+1}\\ \mathbf{v}^{k+1}\end{pmatrix} = M_{\text{PSO}}\begin{pmatrix}\mathbf{x}^{k}\\ \mathbf{v}^{k}\end{pmatrix} + \mathbf{b}_{\text{PSO}}, \tag{12}$$

where

$$\begin{gathered}M_{\text{PSO}} = \begin{pmatrix} w & 1-\phi \\ w & -\phi\end{pmatrix}, \\ \mathbf{b}_{\text{PSO}} = \begin{pmatrix}\phi_1 g^k + \phi_2 l^k \\ \phi_1 g^k + \phi_2 l^k\end{pmatrix}.\end{gathered} \tag{13}$$

The same considerations about random variables ϕ, ϕ_1, ϕ_2, and the use of fixed point techniques applied to (12), allow us to determine the deterministic stability region for the $(\mathbf{x}^{k+1}, \mathbf{v}^{k+1})$ system. The deterministic stability region with moving center of attraction turns to be S_D. Also, this methodology allows us to obtain information about the asymptotic velocity of convergence, $v_c(w, \overline{\phi})$, as a function the spectral radius ρ of the iteration matrix M_{PSO} [9]:

$$v_c(w, \overline{\phi}) = -\ln \rho(M_{\text{PSO}}). \tag{14}$$

This velocity tends to infinite on the vertex of the parabola (10), that is, the point $(w, \overline{\phi}) = (0, 1)$.

3. THE PSO CONTINUOUS MODEL

Let us introduce the following stochastic differential equation:

$$\begin{gathered}x''(t) + (1-\omega)x'(t) + \phi x(t) = \phi_1 g(t) + \phi_2 l(t), \quad t \in \mathbb{R}, \\ x(0) = x_0, \\ x'(0) = v_0,\end{gathered} \tag{15}$$

where $l(t)$ and $g(t)$ are respectively the trajectories of the global and local best trajectories associated to each particle in the swarm, and ϕ, ϕ_1, ϕ_2 are random variables.

This can be considered the PSO continuous model, describing the continuous movement of each coordinate of any particle in the swarm. Referred to the oscillation center,

$$o(t) = \frac{\phi_1 g(t) + \phi_2 l(t)}{\phi}, \tag{16}$$

the continuous model becomes

$$\begin{gathered}\xi''(t) + (1-\omega)\xi'(t) + \phi\xi(t) = -(1-w)o'(t) - o''(t), \quad t \in \mathbb{R}, \\ \xi(0) = \xi_0, \\ \xi'(0) = \xi'_0.\end{gathered} \tag{17}$$

As mentioned before, this last differential equation becomes homogeneous when the oscillation center is stable (stagnation), helping to simplify the PSO analysis.

Let us consider for each of the ith particle coordinates, a centered discretization in acceleration and a regressive schema in velocity in time $t = k \in \mathbb{N}$, and a unit discretization time step, $\Delta t = 1$:

$$\begin{aligned}x'(k) &\simeq \frac{x(k) - x(k-1)}{1}, \\ x''(k) &\simeq \frac{x(k+1) - 2x(k) + x(k-1)}{1}.\end{aligned} \tag{18}$$

It is straightforward to arrive at the PSO-discrete functional model (5).

The PSO differential model (15) has been deduced from a mechanical analogy [9]: a damped mass-spring system of unit mass, damping factor, $b = 1 - w$, and stiffness constant, $k = \phi$. The knowledge of the trajectories of both the continuous and the discrete model helped to explain why some zones of the parameter space (w, ϕ) provide better convergence, in terms of some combined criteria: trajectory attenuation, trajectory oscillation, and center attraction potential. The same conclusions are true for the GPSO as it will be demonstrated.

4. THE GENERALIZED PSO

The idea we propose here is to generalize the PSO algorithm for any time step, Δt. Without loss of generality, the theoretical development is done in one dimension, as if we were reasoning separately for each particle coordinate.

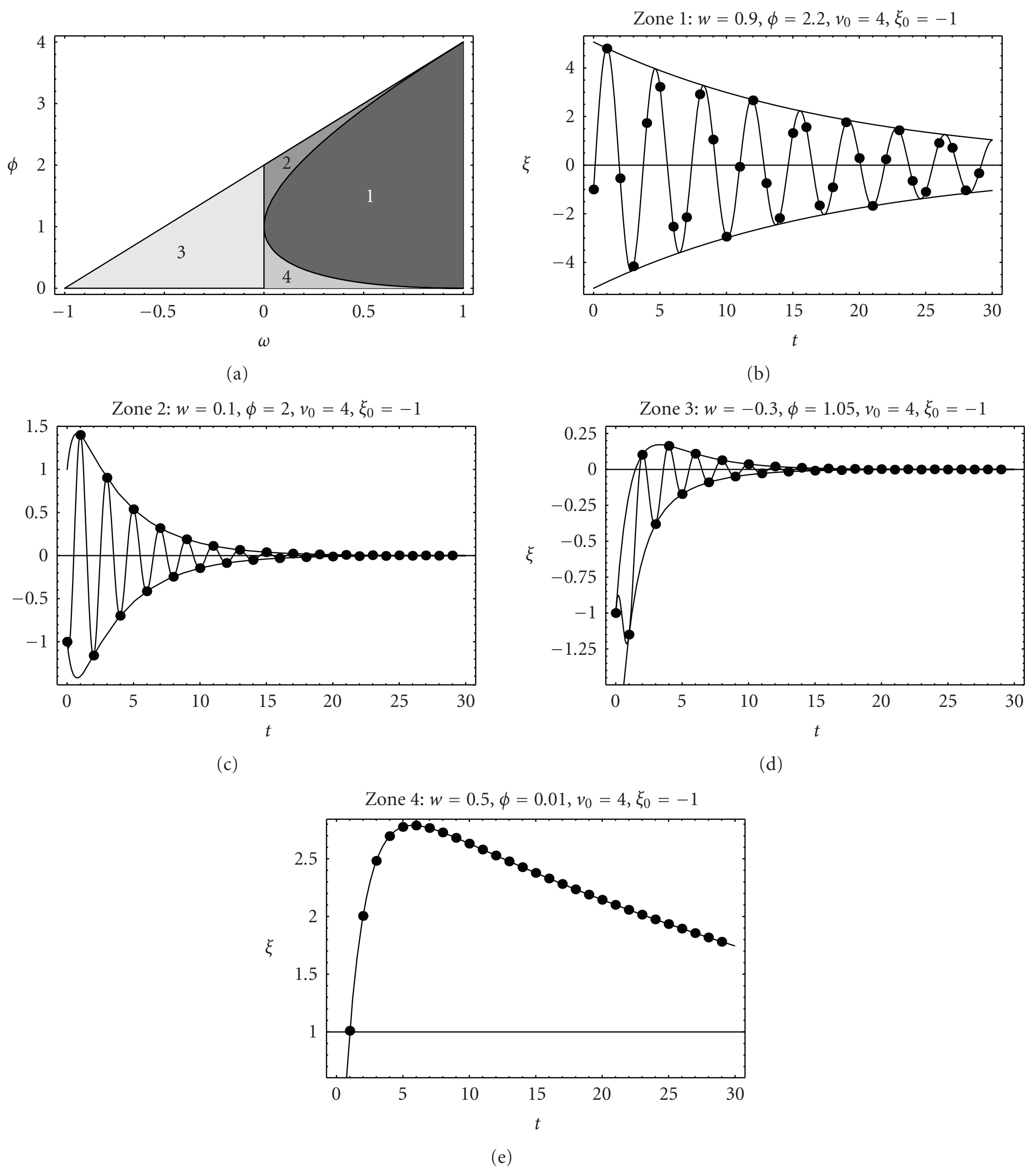


Figure 1: Trajectories for particles in different zones of the PSO deterministic stability region. ZONE 1: complex zone, ZONE 2: real symmetrical oscillating zone, ZONE 3: real asymmetrical oscillating zone, ZONE 4: real nonoscillating zone. The points represent the PSO values. Continuous line is the solution of the PSO-discrete model at real times. Envelopes curves for trajectories are also shown.

Let us consider the same discretization schemes, mentioned above, for velocity and acceleration at time $t \in \mathbb{R}$,

$$x'(t) \simeq \frac{x(t) - x(t-\Delta t)}{\Delta t},$$
$$x''(t) \simeq \frac{x(t+\Delta t) - 2x(t) + x(t-\Delta t)}{\Delta t^2}, \tag{19}$$

and introduce them on the PSO continuous model (15). Then, the following second-order difference equation is obtained for any real times t and $t + \Delta t$:

$$x(t+\Delta t) - Ax(t) - Bx(t-\Delta t) = o(t)\phi\Delta t^2, \tag{20}$$

where

$$A = 2 - (1-w)\Delta t - \phi\Delta t^2,$$
$$B = (1-w)\Delta t - 1. \tag{21}$$

Stochastic functional equation (20) corresponds to the so-called generalized PSO algorithm.

The GPSO algorithm can be written in terms of the absolute position and velocity $(x(t), v(t))$ as follows:

$$v(t+\Delta t) = (1-(1-\omega)\Delta t)v(t) + \phi\Delta t(o(t) - x(t)),$$
$$x(t+\Delta t) = x(t) + v(t+\Delta t)\Delta t. \tag{22}$$

Introducing $\Delta x(t+\Delta t) = v(t+\Delta t)\Delta t$, the algorithm becomes

$$\begin{aligned}\Delta x(t+\Delta t) &= (1-(1-\omega)\Delta t)\Delta x(t) + \phi\Delta^2 t((o(t)-x(t))),\\ x(t+\Delta t) &= x(t) + \Delta x(t+\Delta t).\end{aligned} \tag{23}$$

This model is valid for any of the particles of the swarm in any dimension. Considering $t = k \in \mathbb{N}$, $\Delta t = 1$, we arrive at the PSO algorithm. Also, taking into account expression (1), it is easy to show that this algorithm corresponds to a modified PSO with inertia

$$w_{\Delta t} = 1 - (1-\omega)\Delta t, \tag{24}$$

and total acceleration

$$\phi_{\Delta t} = (\phi_1 + \phi_2)\Delta t^2 = \phi\Delta t^2, \tag{25}$$

and thus, the effect of decreasing Δt (PSO corresponds to $\Delta t = 1$) is to lower the total acceleration according to Δt^2, and increasing the inertia proportionally to Δt. Thus, the unit-mass damped spring system involved has a damping factor

$$b_{\Delta t} = 1 - w_{\Delta t} = (1-\omega)\Delta t = b\Delta t, \tag{26}$$

instead of $b = 1 - w$, and a stiffness constant

$$k_{\Delta t} = \phi_{\Delta t} = (\phi_1 + \phi_2)\Delta^2 t = \phi\Delta^2 t, \tag{27}$$

instead of $k = \phi$. This means that the swarm movement becomes more elastic and less damped for $\Delta t \to 0$. In fact, the pair $(w_{\Delta t}, \phi_{\Delta t})$ follows the parabola

$$\phi_{\Delta t} = \frac{\phi}{(1-w)^2}(1 - w_{\Delta t})^2, \tag{28}$$

approaching to the point $(w_{\Delta t}, \phi_{\Delta t}) = (1, 0)$, as Δt decreases.

4.1. GPSO deterministic stability analysis

This approach allows us also to calculate the deterministic stability region of the generalized PSO.

4.1.1. Stagnation case

In case of stagnation, the GPSO difference equation referred to the oscillation center becomes

$$\begin{aligned}\xi(t+\Delta t) - A\xi(t) - B\xi(t-\Delta t) &= 0,\\ \xi^0 &= \xi_0,\\ \xi^1 &= (1-\phi\Delta t^2)\xi_0 + (1-(1-w)\Delta t)\Delta t v_0,\end{aligned} \tag{29}$$

where A, B are given by (21).

The deterministic stability region is the part of the space $(\omega, \overline{\phi})$, where the roots of the characteristic equation

$$\lambda^{2\Delta t} + ((1-w)\Delta t - 2 + \overline{\phi}\Delta t^2)\lambda^{\Delta t} + (1-(1-w)\Delta t) = 0 \tag{30}$$

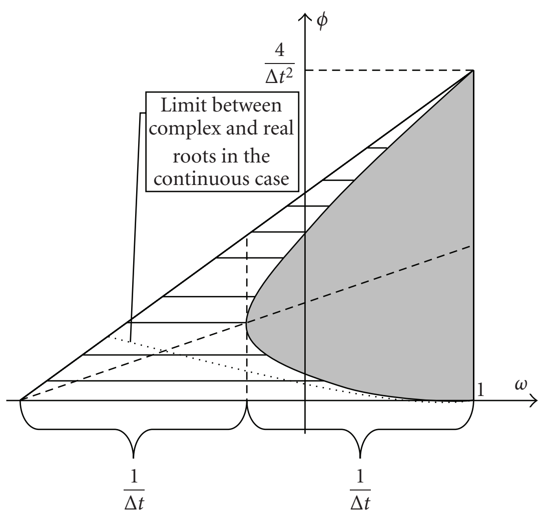


Figure 2: Deterministic stability region for the generalized PSO.

are in the unit circle. This region (see Figure 2) turns to be

$$S_{\text{GPSO}} = \left\{(\omega, \overline{\phi}) : 1 - \frac{2}{\Delta t} < \omega < 1,\right.\\ \left. 0 < \overline{\phi} < \frac{1}{\Delta t^2}(2\Delta t\omega - 2\Delta t + 4)\right\}. \tag{31}$$

The shadowed part represents the zone where roots are complex. In the rest, both roots are real. Parabola separating real and complex roots is given by

$$\begin{aligned}&(\overline{\phi}\Delta t^2 + (1-w)\Delta t - 2)^2\\ &\quad = 4(1-(1-w)\Delta t), \quad 1 - \frac{1}{\Delta t} \le w < 1.\end{aligned} \tag{32}$$

One can show that as $\Delta t \to 0$, the stability region of the generalized PSO tends to the continuous stability region, that is,

$$S_C = \{(\omega, \overline{\phi}) : \omega < 1,\ \overline{\phi} > 0\}. \tag{33}$$

The upper border line of the PSO algorithm

$$\overline{\phi} = 2(w+1), \quad |\omega| < 1 \tag{34}$$

is in fact due to the time discretization adopted for PSO ($\Delta t = 1$) and comes from a mathematical restriction to achieve stability. Fernández Martínez et al. [9] mentioned this effect, comparing the PSO stability region to its continuous counterpart. Border line (32) separating real and complex roots also follows a similar behavior. Figure 3 shows numerically, when $\Delta t \to 0$, how the discrete stability zone increases its size, approaching the continuous region (33). Figure 4 shows that, for a given parameter set (w, ϕ) on the stability zone, the GPSO trajectories tend to the continuous as Δt decreases, and thus, the GPSO sampling becomes more dense. On the opposite, as Δt increases, the stability triangle of the generalized PSO shrinks.

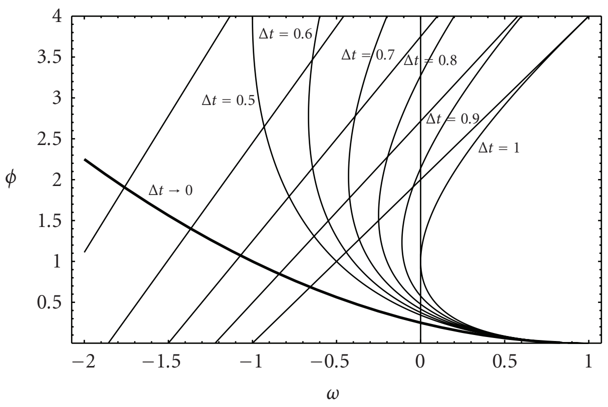


FIGURE 3: Evolution of stablity zone for different decreasing Δt, approaching the continuous case.

4.1.2. *Moving center of attraction*

The same deterministic stability region, S_{GPSO}, can be deduced applying fixed point techniques to the GPSO system:

$$\begin{pmatrix} x(t+\Delta t) \\ v(t+\Delta t) \end{pmatrix} = M_{\text{GPSO}} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \mathbf{b}_{\text{GPSO}},$$
$$M_{\text{GPSO}} = \begin{pmatrix} (1-(1-w)\Delta t)\Delta t & 1-\overline{\phi}\Delta t^2 \\ 1-(1-w)\Delta t & -\overline{\phi}\Delta t \end{pmatrix}, \qquad (35)$$
$$\mathbf{b}_{\text{GPSO}} = \begin{pmatrix} \Delta t^2(\overline{\phi}_1 g(t) + \overline{\phi}_2 l(t)) \\ \Delta t(\overline{\phi}_1 g(t) + \overline{\phi}_2 l(t)) \end{pmatrix}.$$

This proves that the GPSO deterministic stability analysis shown before is also valid when the oscillation center changes.

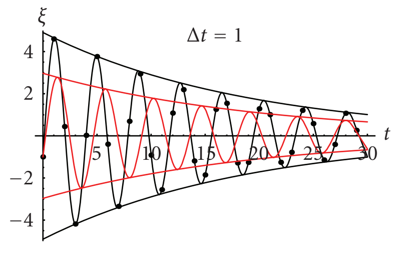
Figure 5 shows the deterministic spectral radius—associated to the asymptotic velocity of convergence—for the generalized PSO. The similarity with the PSO case ($\Delta t = 1$) is absolute, and the black zone of high velocity increases in size. The asymptotic velocity tends to infinite on the parabola vertex, $(1 - 1/\Delta t, 1/\Delta t^2)$.

5. SOME INTERESTING PROPERTIES OF THE GPSO

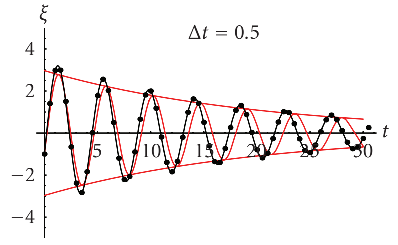
With GPSO, we have performed a numerical analysis that seeks to explain why some pairs of (w, ϕ) in certain zones of the deterministic convergence triangle work in achieving convergence and why others do not.

5.1. *Center attraction*

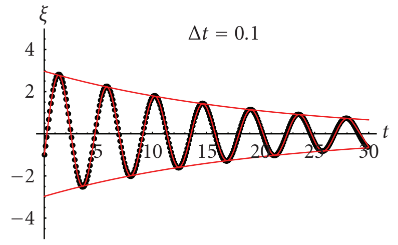
For a GPSO simplified model where $\mathbf{o}_i^k = \mathbf{0}$, without objective function, two tests have been made: one deterministic (see Figure 6(a)) and the other using random ϕ parameters (see Figure 6(b)).



(a)



(b)



(c)

FIGURE 4: Convergence of the generalized PSO trajectories to the continuous as $\Delta t \to 0$. Black points represent GPSO values.

For the deterministic one,

(1) 100 particles, $\xi_p^0 = (x_p^0, y_p^0)$, are randomly initialized over the two dimensional domain $[-100, 100] \times [-100, 100]$;

(2) a parameter grid (ω, ϕ) is defined over the region $(\omega, \phi) \in [-3, 1] \times [0, 18]$ with the following grid steps: $\Delta w = 0.06$, $\Delta\phi = 0.02$;

(3) each particle $\xi_p = (x_p, y_p)$ follows the simplified trajectory:

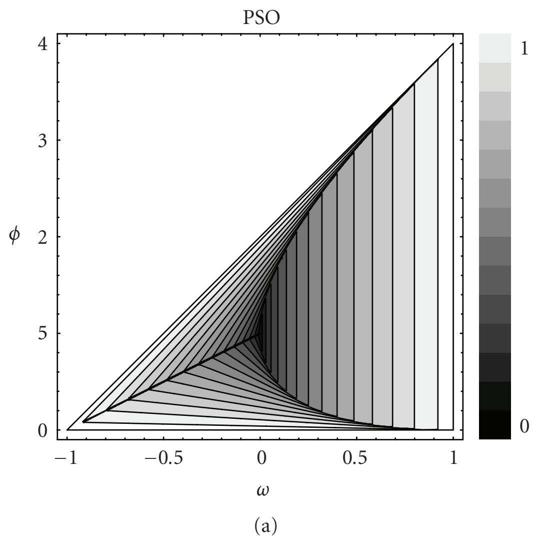
$$\mathbf{v}^{k+1} = (1-(1-\omega)\Delta t)\mathbf{v}^k - \phi\Delta t \xi^k,$$
$$\xi^{k+1} = \xi^k + \mathbf{v}^{k+1}\Delta t, \qquad (36)$$

for each point (ω, ϕ) of the parameter grid;

(4) for each (ω, ϕ), the particle nearest to the attractor point—$(0, 0)$ in this case—is found after 100 iterations. Its distance to the attractor point, $d_{\min}(\omega, \phi)$, is calculated and expressed in logarithmic scale;

(5) The contour map $\log d_{\min}(\omega, \phi)$ is plotted over the parameter region $[-3, 1] \times [0, 18]$.

In the random case, the same test has been performed, but 300 simulations have been made for each (ω, ϕ) (instead of only one simulation in the deterministic case) and the median of $d_{\min}(\omega, \phi)$ has been computed. It can be observed that



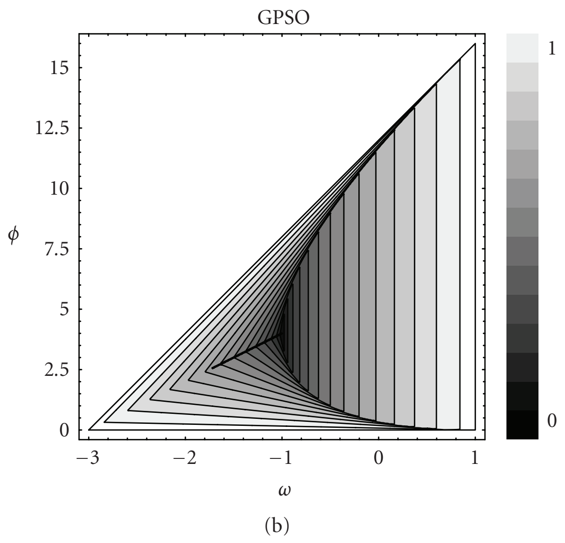
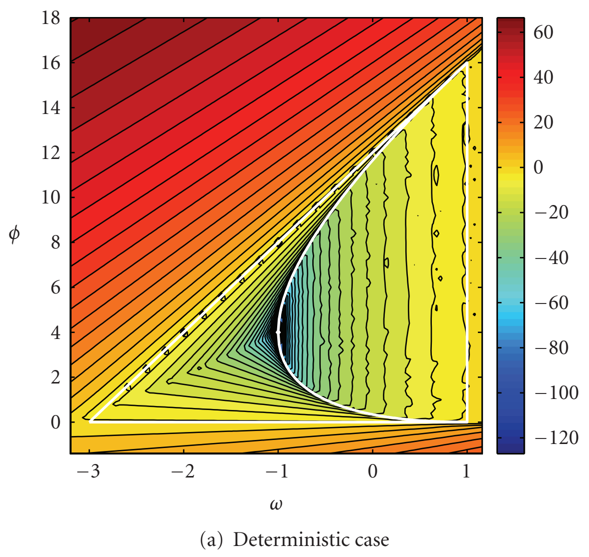


FIGURE 5: Spectral radius for PSO and for GPSO with time step $\Delta t = 0.5$. Black area close to the parabola vertex stands for null spectral radius.



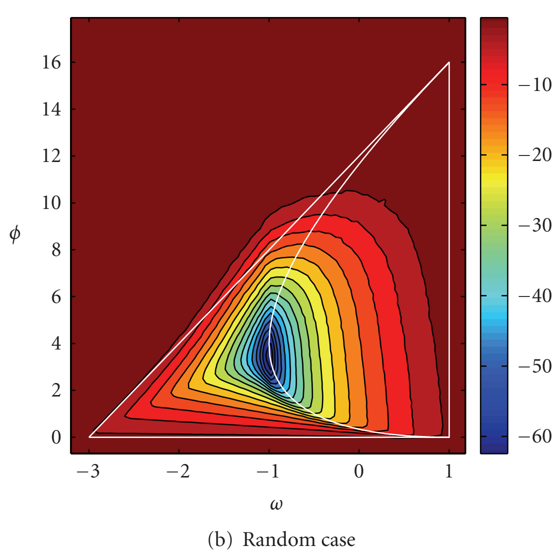


FIGURE 6: Effect of the randomness in the center attraction capability for GPSO with $\Delta t = 0.5$.

in the upper-right zone of the convergence triangle there is no convergence (see Figure 6(b)). This is due to the high amplitudes and frequencies of the trajectories in this zone.

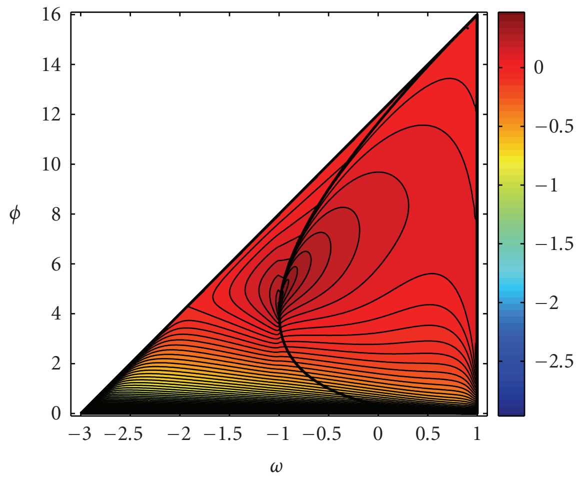
5.2. *Comparison to the PSO continuous model*

It has been observed that in zones of optimal convergence, the continuous and discrete trajectories differ a little [9]. Figure 7 shows the relative logarithmic misfit between the GPSO for $\Delta t = 0.5$ and the PSO continuous model, both using their trajectories and/or their respective envelope curves. It can be observed that as Δt decreases, the size of the zone of similarity increases. The PSO stability triangle is embedded in this zone.

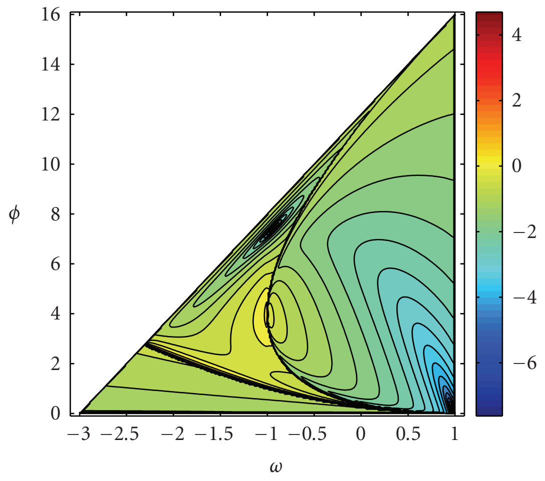
5.3. *Attenuation and oscillation*

No convergence close to the point $(w, \phi) = (1 - 1/\Delta t, 1/\Delta t^2)$, vertex of the parabola, can be explained by the quick attenuation of the trajectories amplitude and fast convergence to the oscillation center, that in the first moments of the application of the algorithm is usually far from the optimum. Figure 8 shows, for different (w, ϕ) points on the stability region, the number of iterations (expressed in logarithmic scale) needed to reduce the amplitude of the GPSO trajectories in 90% for $\Delta t = 0.5$, and the comparison to PSO and to the continuous case. As in the previous figure, similarity with the continuous increases as $\Delta t \to 0$.

The area between envelopes for the discrete trajectories has been used as a measure of the exploratory capacity of the



(a) Misfit using trajectories



(b) Misfit using envelopes

FIGURE 7: GPSO for $\Delta t = 0.5$. (a) logarithmic relative misfit L_2 error between continuous and discrete solutions; (b) logarithmic relative misfit L_2 error using envelope curves.

PSO. It can be seen that it is higher as w increases in the complex zone and near to the upper convergence limit in the real zone. Figure 9 shows the area between envelope curves for the GPSO with $\Delta t = 0.5$ and its comparison to the PSO case. These two last magnitudes (attenuation and oscillation) have been proposed as measures of the exploration capabilities associated to the deterministic PSO trajectories [9].

6. THE SAMPLING DISTRIBUTION OF THE GPSO ALGORITHM

In this section, we analyze the effect of stochasticity on the stability of GPSO trajectories using the methodology first proposed by Poli [12]. Trajectories are modeled as stochastic processes whose univariate and bivariate statistical moments must satisfy the PSO difference equations involved. The methodology consists in discretizing the PSO continuous model (15) and applying fixed point analysis to the dynamical systems deduced for the first- to fourth-order moments describing the trajectories stability. This analysis is performed under stagnation and with a moving center of attraction. We show that the stability regions are the same in both cases.

This methodology becomes, nevertheless, complicated and time consuming since it involves a linear system of the kind

$$\mathbf{y}_n(t + \Delta t) = M_n \mathbf{y}_n(t) + \mathbf{b}_n \tag{37}$$

with iteration matrix $M_n \in M(n(n + 3)/2, n(n + 3)/2)$, for controlling the stability of the n-order moments.

6.1. *Stagnation case*

6.1.1. *First- and second-order moments*

Let us consider the GPSO difference equation

$$\xi(t + \Delta t) - A\xi(t) - B\xi(t - \Delta t) = 0, \quad t, \Delta t \in \mathbb{R}, \tag{38}$$

where A, B are given by (21).

Let us also consider the vector describing the dynamics of the first- and second-order moments:

$$\begin{aligned}\mathbf{z}_2(t) = \big(&E(\xi(t)), E(\xi(t - \Delta t)),\\ &E(\xi^2(t)), E(\xi(t)\xi(t - \Delta t)),\\ &E(\xi^2(t - \Delta t))\big)^t\end{aligned} \tag{39}$$

It is straight forward to show that the stochastic GPSO dynamics can be written on algebraic form as

$$\mathbf{z}_2(t + \Delta t) = \begin{pmatrix} E(A) & B & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & E(A^2) & 2BE(A) & B^2 \\ 0 & 0 & E(A) & B & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{z}_2(t), \tag{40}$$

where

$$\begin{aligned} E(A) &= 2 - (1 - w)\Delta t - \Delta t^2 \overline{\phi}, \\ E(A^2) &= (2-(1 - w)\Delta t)^2 - 2(2-(1 - w)\Delta t)\Delta t^2 \overline{\phi} + \Delta t^4 E(\phi^2), \\ E(\phi^2) &= \frac{a_l^2}{3} + \frac{a_g^2}{3} + \frac{a_l a_g}{2}. \end{aligned} \tag{41}$$

The following results are of interest.

(1) Stability of the first-order moments only involves the matrix $M_1 = \left(\begin{smallmatrix} E(A) & B \\ 1 & 0 \end{smallmatrix}\right)$, and logically provides the same results as GPSO deterministic stability analysis.

(2) The stability of the second-order moments involves the matrix

$$M_2 = \begin{pmatrix} E(A^2) & 2BE(A) & B^2 \\ E(A) & B & 0 \\ 1 & 0 & 0 \end{pmatrix}. \tag{42}$$

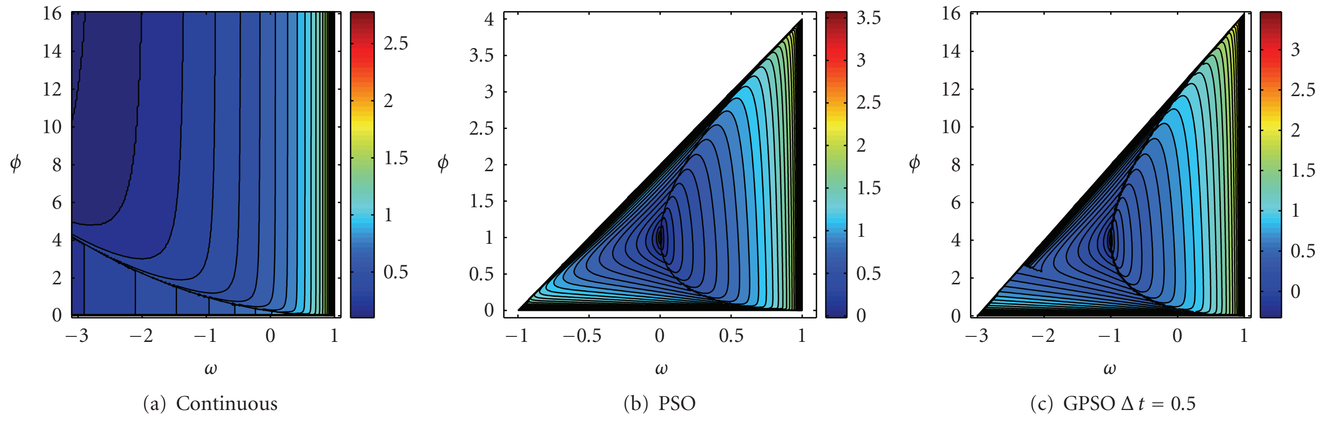


Figure 8: Number of iterations needed to decrease 90% in amplitude, expressed in logarithmic scale, with initial conditions $x_i(0) = 1$, $v_i(0) = 0$. (a) continuous case, (b) PSO, (c) GPSO with $\Delta t = 0.5$.

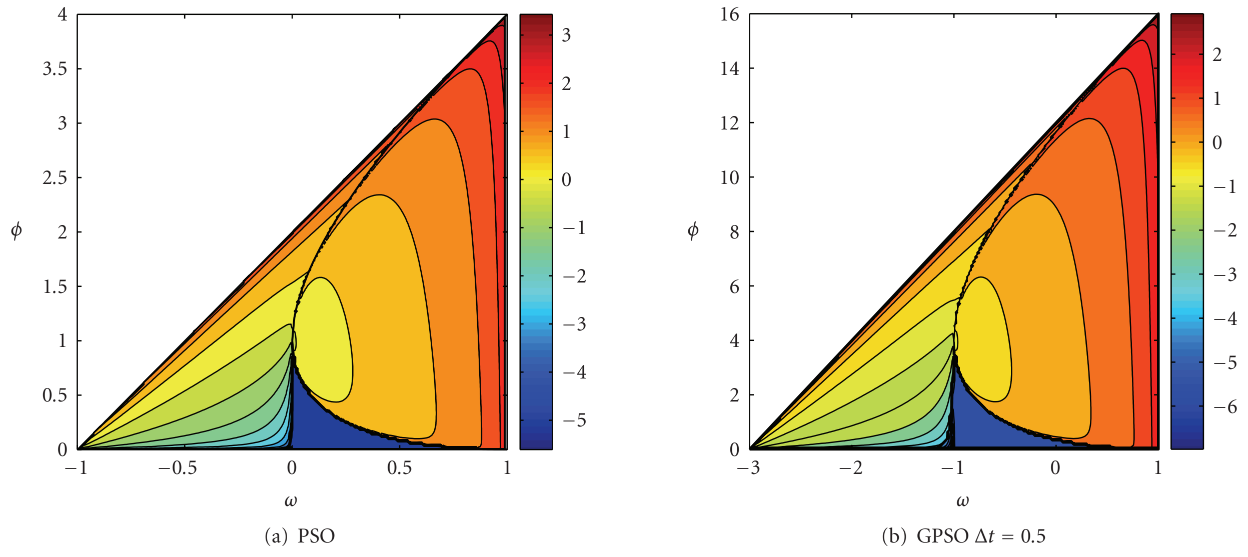


Figure 9: Area between envelope curves using logarithmic scale. (a) PSO, (b) GPSO with $\Delta t = 0.5$.

(3) Analysis of the fixed point linear system involved

$$\begin{pmatrix} p_1 \\ p_1 \\ p_2 \\ p_3 \\ p_2 \end{pmatrix} = \begin{pmatrix} E(A) & B & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & E(A^2) & 2BE(A) & B^2 \\ 0 & 0 & E(A) & B & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_1 \\ p_2 \\ p_3 \\ p_2 \end{pmatrix} \tag{43}$$

has allowed us to determine the border line of the second order stability region. Calling $\alpha = a_g/\overline{\phi}$, it is possible to show that this line is a hyperbola and has the following explicit equation:

$$\phi_h(w, \alpha, \Delta t) = \frac{12}{\Delta t}\, \frac{(1-w)(2+(w-1)\Delta t)}{4-4(w-1)\Delta t+(\alpha^2-2\alpha)(2+(w-1)\Delta t)}. \tag{44}$$

This line generalizes for any Δt, the border line of the second-order stability region deduced by Poli [12] for $\Delta t = 1$(PSO) and $\alpha = 1$ ($a_l = a_g$). The GPSO stochastic stability region turns to be

$$R_{\text{GPSO}} = \left\{(\omega, \overline{\phi}) : 1 - \frac{2}{\Delta t} < \omega < 1,\ 0 < \overline{\phi} < \phi_h(w, \alpha, \Delta t)\right\} \tag{45}$$

which is embedded on the deterministic stability region S_{GPSO}. Also, it is possible to show that this region reaches its maximum size if $\alpha = 1$ ($a_l = a_g$).

Figure 10 shows the contour lines of the spectral radius of the stochastic iteration matrix on the unit circle for $\Delta t = 0.5$ and $\alpha = 1$. As it can be observed, the isolines bend on their upper part, remembering the results shown in Figure 6(b).

Finally, the second-order stability region increases its size as Δt diminishes. Besides, GPSO algorithm with $\Delta t > 1$ introduces for the same (w, ϕ) a higher variance on the trajectories.

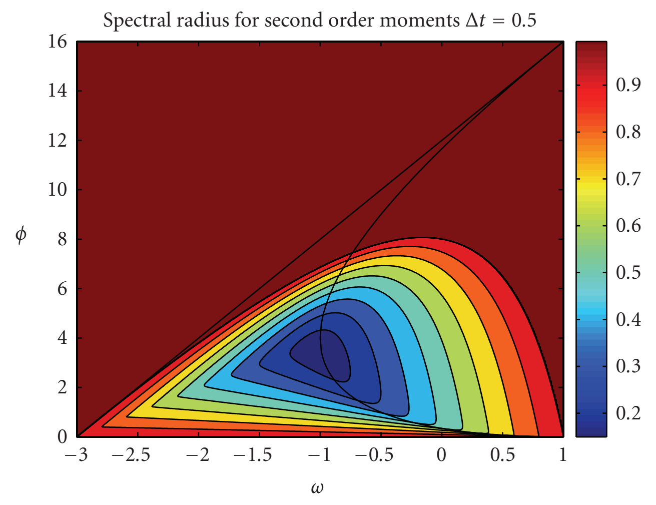


FIGURE 10: Contour plot spectral radius of iteration matrix for $a_l = a_g$ $(\alpha = 1)$ and $\Delta t = 0.5$ in the second-order stability region.

The same methodology can be applied to determine the skewness and kurtosis stability regions. Results are presented for a moving center of attraction.

6.2. *Moving center of attraction*

Let us consider the GPSO difference equation in absolute positions:

$$x(t+\Delta t) - Ax(t) - Bx(t-\Delta t) = (\phi_1 g(t) + \phi_2 l(t))\Delta t^2, \quad t, \Delta t \in \mathbb{R}, \tag{46}$$

where A, B are given by (21). In what follows we will consider $g(t)$ and $l(t)$ as deterministic functions.

Let us also consider the vector

$$\mathbf{y}_2(t) = (E(x(t)), E(x(t-\Delta t)), E(x^2(t)), E(x(t)x(t-\Delta t)), E(x^2(t-\Delta t)))^t \tag{47}$$

which describes the first- and second-order GPSO "absolute" dynamics. The GPSO system can be written on algebraic form as

$$\mathbf{y}_2(t+\Delta t) = \begin{pmatrix} E(A) & B & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 2E(AC) & 2BE(C) & E(A^2) & 2BE(A) & B^2 \\ E(C) & 0 & E(A) & B & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{y}_2(t) + \begin{pmatrix} E(C) \\ 0 \\ E(C^2) \\ 0 \\ 0 \end{pmatrix}, \tag{48}$$

where

$$C = (\phi_1 g(t) + \phi_2 l(t))\Delta t^2. \tag{49}$$

In this case, the dynamics of the second-order moments also depend on the first-order moments.

One can show the following.

(1) The stochastic stability region for the first- and the second-order moments is the same that in case of stagnation.

(2) The fix points for $\mathbf{y}_2(t+\Delta t)$ are the following:

$$E(x(t_p)) = \frac{a_g g(t_p) + a_l l(t_p)}{a_g + a_l}, \tag{50}$$

$$\operatorname{Var}(x(t_p)) = \frac{1}{4}\frac{(\alpha-2)^2\alpha^2\overline{\phi}(2+(w-1)\Delta t)\Delta t}{\Delta} \times (g(t_p) - l(t_p))^2, \tag{51}$$

$$\operatorname{Cov}(x(t_p), x(t_p+\Delta t)) = \frac{1}{4}\frac{(\alpha-2)^2\alpha^2\overline{\phi}(2+(w-1)\Delta t - \Delta t^2\overline{\phi})\Delta t}{\Delta}(g(t_p)-l(t_p))^2, \tag{52}$$

where

$$\Delta = 24(1-w) - 2\Delta t(6(w-1)^2 + \overline{\phi}(\alpha^2 - 2\alpha + 2)) - \Delta t^2\overline{\phi}(4 + 2\alpha - \alpha^2)(1-w), \tag{53}$$

and t_p stands for the time when the fixed point is reached.

(3) The variance increases its value from 0 in $\overline{\phi} = 0$, until $+\infty$ on the stochastic border line. Isolines are shown in Figure 11(a). Variance is also null for $\alpha = 2\,(a_l = 0)$, $\alpha = 0\,(a_g = 0)$.

(4) The covariance is zero in $\overline{\phi} = 0$, and in the line $\overline{\phi} = (2 + \Delta t(w-1))/\Delta t^2$, which is one of the median of the deterministic stability triangle. Covariance is negative above this line. Figure 11(b) shows the contour plot of $\operatorname{sgn}(\mathrm{Cov}) \cdot |\ln|\mathrm{Cov}||\cdot$ Covariance is also null for any Δt if $a_l = 0$ or $a_g = 0$.

(5) A useful measure for trajectories dispersion is the variogram, defined as

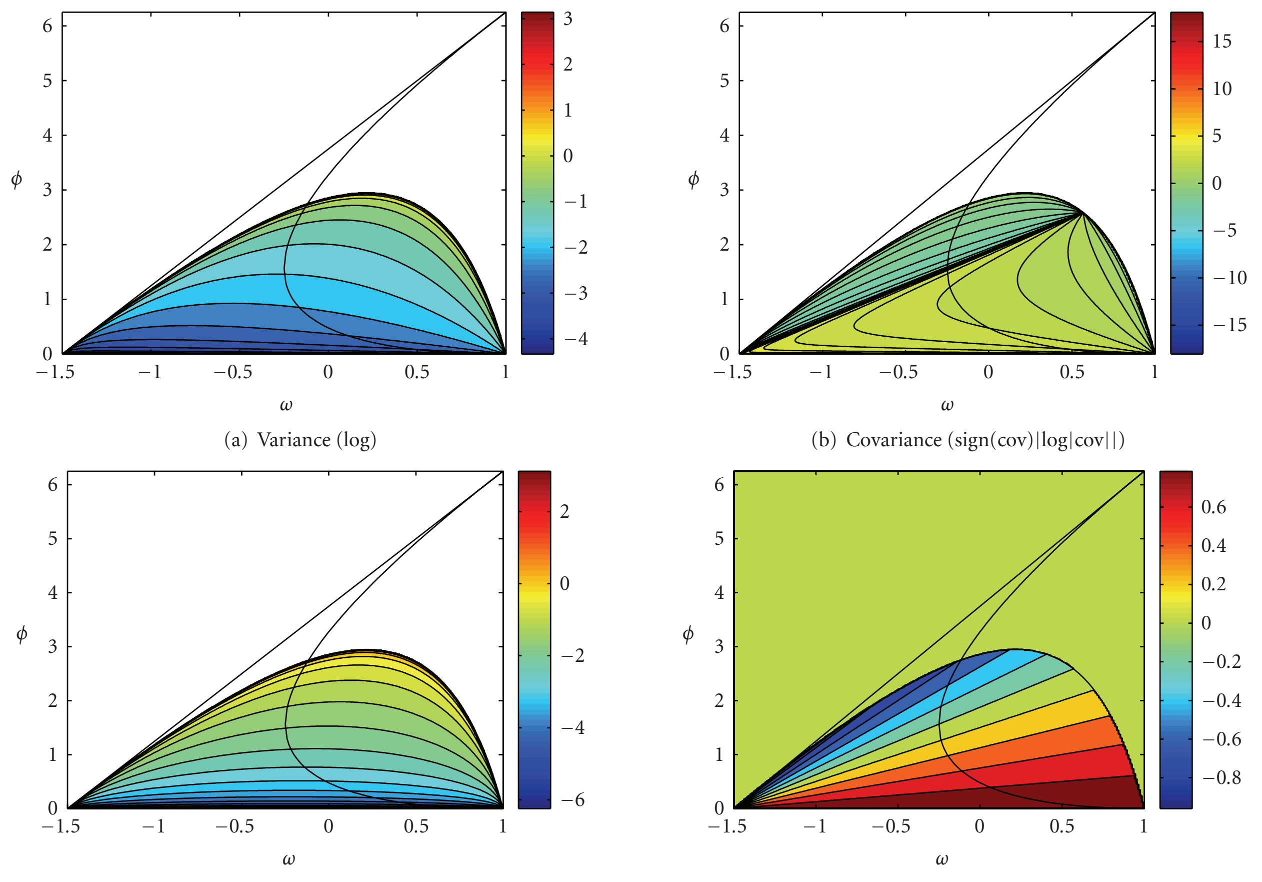
$$\gamma(x(t_p), x(t_p+\Delta t)) = E\left[(x(t_p) - x(t_p+\Delta t))^2\right] = \frac{1}{4}\frac{(\alpha-2)^2\alpha^2\overline{\phi}^2\Delta t^3}{\Delta}(g(t_p) - l(t_p))^2. \tag{54}$$

Variogram shows that dispersion increases from 0 in $\overline{\phi} = 0$, until $+\infty$ on the second-order stability border line. The variogram is also null for any Δt if $a_l = 0$ or $a_g = 0$. Isolines are shown in Figure 11(c).

(6) The correlation coefficient is

$$\rho(x(t_p), x(t_p+\Delta t)) = \frac{\operatorname{Cov}(x(t_p), x(t_p+\Delta t))}{\sqrt{\operatorname{Var}(x(t))\cdot\operatorname{Var}(x(t+\Delta t))}} = \frac{2+(w-1)\Delta t - \Delta t^2\overline{\phi}}{2+(w-1)\Delta t}, \quad \Delta t \neq 0, \tag{55}$$

for any point $(w, \overline{\phi})$ on the stability region of the second-order moments, and does not depend on α, neither on the local and global best trajectories. Correlation coefficient is 1



(a) Variance (log)

(b) Covariance (sign(cov)|log|cov||)

(c) Variogram (log)

(d) Correlation coefficient

Figure 11: Contour plots of the main second-order moments for $a_l = 2a_g$ ($\alpha = 0.5$) and $\Delta t = 0.8$: (a) variance (in logaritmic scale), (b) covariance (sign(cov)|log|cov||), (c) variogram (log), (d) correlation coefficient.

in $\overline{\phi} = 0$, and is null on the line $\overline{\phi} = (2 + \Delta t(w - 1))/\Delta t^2$. Isolines are straight lines passing in $(w, \phi) = (1 - 2/\Delta t, 0)$ (see Figure 11(d)).

(7) A priori a good parameter choice is given by the intersection by the line, $\overline{\phi} = (2+\Delta t(w-1))/\Delta t^2$ (no correlation between trajectories), with the border line of the stochastic stability region (maximum variance and dispersion). The point of intersection has the following coordinates:

$$w = \frac{-4 + 4\alpha - 2\alpha^2 + \Delta t(8 - 2\alpha + \alpha^2)}{(8 - 2\alpha + \alpha^2)\Delta t},$$
$$\overline{\phi} = \frac{12}{(8 - 2\alpha + \alpha^2)\Delta t^2}, \tag{56}$$

which in the case of PSO and $\alpha = 1$ (maximum size of the stochastic stability region) is $(w, \overline{\phi}) = (5/7, 12/7) = (0.714, 1.714)$.

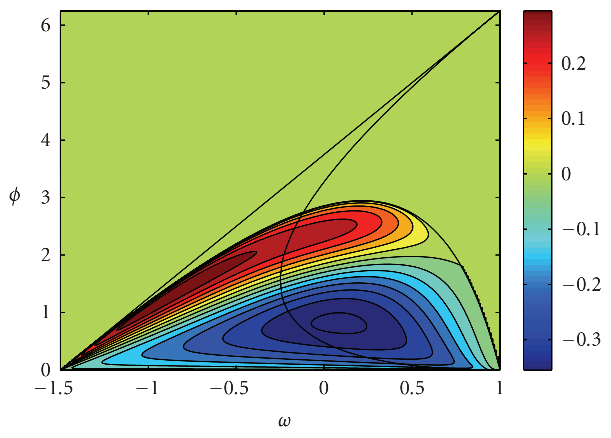
(8) This analysis can be generalized when $l(t)$ and $g(t)$ are considered as stochastic processes. (The results presented here are valid considering that the center of attraction and the trajectories are independent. Obviously, this is a wrong hypothesis. The analysis of the general case will be addressed in a future paper devoted to this important subject.) The first- and second-order stability zones do not change. The mean is, in this case,

$$E(x(t_p)) = \frac{1}{2}[\alpha E[g(t)] + (2 - \alpha)E[l(t)]], \tag{57}$$

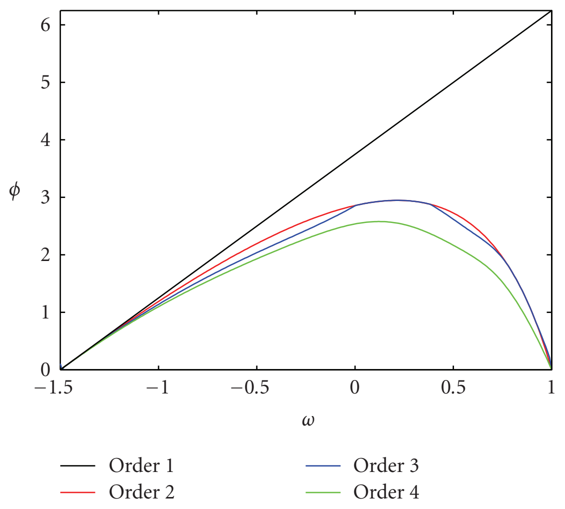
which generalizes the expression (50). The variance, covariance, and variogram show now a more complicated dependency which involves the first- and second-order moments of $l(t)$ and $g(t)$ in t_p, instead of $(g(t_p) - l(t_p))^2$. For instance, the variance and covariance become

$$\mathrm{Var}(x(t_p)) = \frac{1}{4}\frac{\overline{\phi}(2 + (w - 1)\Delta t)\Delta t}{\Delta}K_{\mathrm{gl}},$$
$$\mathrm{Cov}(x(t_p), x(t_p + \Delta t)) = \frac{1}{4}\frac{\overline{\phi}(2 + (w - 1)\Delta t - \Delta t^2\overline{\phi})\Delta t}{\Delta}K_{\mathrm{gl}},$$
$$K_{\mathrm{gl}} = 2(\alpha - 2)^2\mathrm{Var}[l(t_p)] + 2\alpha^2\mathrm{Var}[g(t_p)] - 3\alpha(\alpha - 2) \times \mathrm{Cov}[g(t_p), l(t_p)] + \alpha^2(\alpha - 2)^2(E[g(t_p)] - E[l(t_p)])^2. \tag{58}$$

These expressions simplifies to (51), (52), and (54) in case $l(t)$ and $g(t)$ are deterministic. Correlogram (55) has the same expression in both cases.



(a) Skewness



(b) Stability regions for order 1, 2, 3, and 4

FIGURE 12: (a) Contour plot of skewness for $a_l = 2a_g$ ($\alpha = 0.5$) and $\Delta t = 0.8$. (b) Stability regions of orders 1, 2, 3, and 4.

6.2.1. *Skewness*

The stability analysis for the third-order moments involves a linear system with an $\mathbb{R}^9 \times \mathbb{R}^9$ iteration matrix. The iteration matrix extra terms corresponding to the different third-order moments $(E(x^3(t)), E(x^3(t-\Delta t)), E(x^2(t)x(t-\Delta t)), E(x(t)x^2(t-\Delta t)))$ are

$$M_3 = \begin{pmatrix} E(A^3) & B^3 & 3E(A^2)B & 3E(A)B^2 \\ 1 & 0 & 0 & 0 \\ E(A^2) & 0 & 2E(A)B & B^2 \\ E(A) & 0 & B & 0 \end{pmatrix}. \tag{59}$$

Our analysis provides the following results.

(i) The region where the skewness fix point exists coincides with the region of stability of the second-order moments, nevertheless, the region of skewness stability (where the fixed point is correctly approximated by the iterative PSO algorithm) is smaller in size than the region of second-order stability. Both regions coincide for many w values on $(-1, 1)$. This result has been also pointed by Poli [12] in the PSO case, for inertia values restricted to the interval $[0, 1]$. Also, the region of skewness stability tends to the region of second-order stability as time step, Δt, goes to zero.

(ii) Skewness is null for any Δt if $\alpha = 1$ $(a_g = a_l)$. Skewness is positive in the upper zone under the second-order stability hyperbola for $\alpha < 1$ $(a_g < a_l)$, and negative in the bottom zone. For $\alpha > 1$ $(a_g > a_l)$ the sign swaps between these two zones for any Δt (see Figure 12(a)). Skewness does only depend on the sign of $g(t_p) - l(t_p)$ but not on its absolute value.

6.2.2. *Kurtosis*

Kurtosis stability analysis involves a linear system with $\mathbb{R}^{14} \times \mathbb{R}^{14}$ iteration matrix.

The main results are the following.

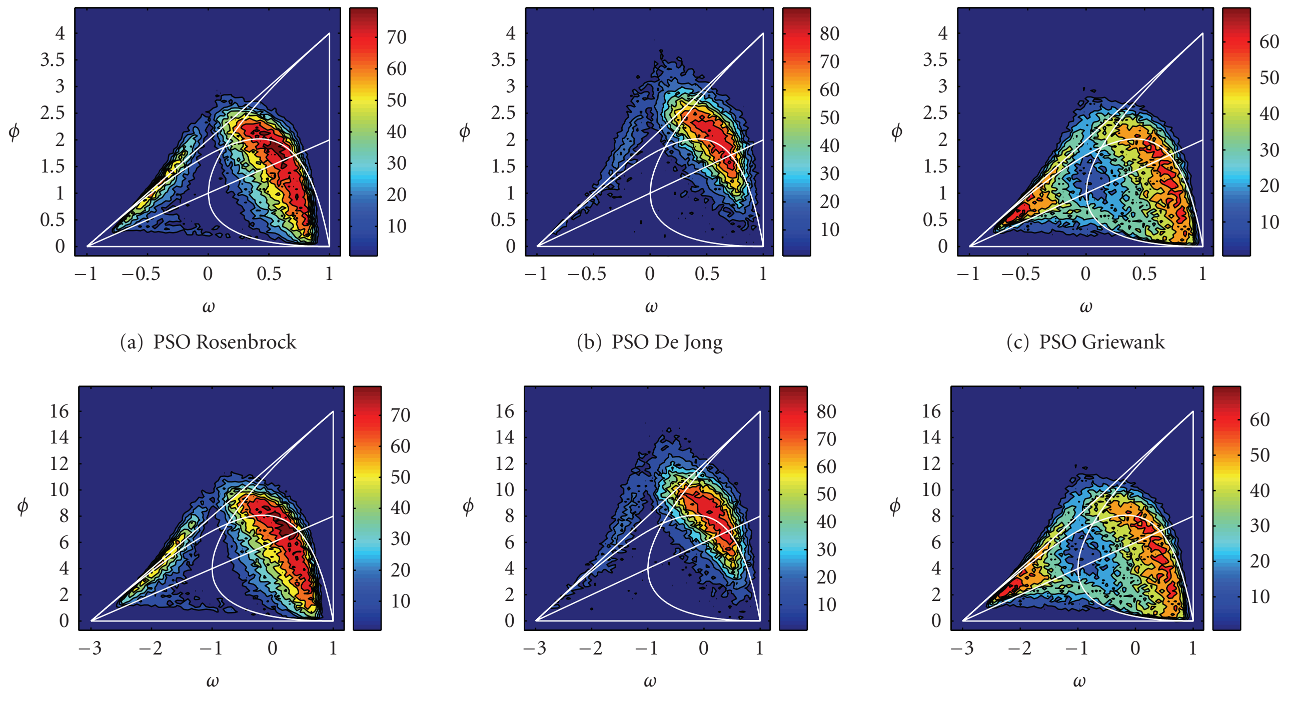
(i) The regions of orders 1, 2, 3, and 4 are nested, being the region of kurtosis stability the smallest on size (see Figure 12(b)). This imply that while mean, variance, covariance, and skewness are stable, the kurtosis grows indefinitely. This last result has been also pointed by [12] in the PSO case under stagnation.

(ii) All these regions of stability tend to coincide as Δt decreases (GPSO case with $\Delta t < 1$). In the limit $(\Delta t \to 0)$, they tend to the continuous stability region (33).

7. NUMERICAL EXPERIMENTS ON BENCHMARK FUNCTIONS

To confirm the above-mentioned theoretical results, we ran several numerical experiments on benchmark functions with two types of ill-posedness commonly found in inverse problems: the Rosenbrock and the "elongated" DeJong functions (global minimum located in a very flat area), and the Griewank function (global minimum surrounded by multiple minima). A similar analysis has been done in [9] for the PSO case, showing that as the ill-posedness increases, PSO parameters from the complex stability region, with inertia values from 0.5 to 0.9, and medium to high total accelerations $(1.5 < \overline{\phi} < 2)$ seem to give systematically very good results. Also, in case of cost functions with multiple local minima, the real stability zone of negative inertia values (around -0.55) and total acceleration from 0.6 to 1.2 give very good performance with respect to the percentage of success.

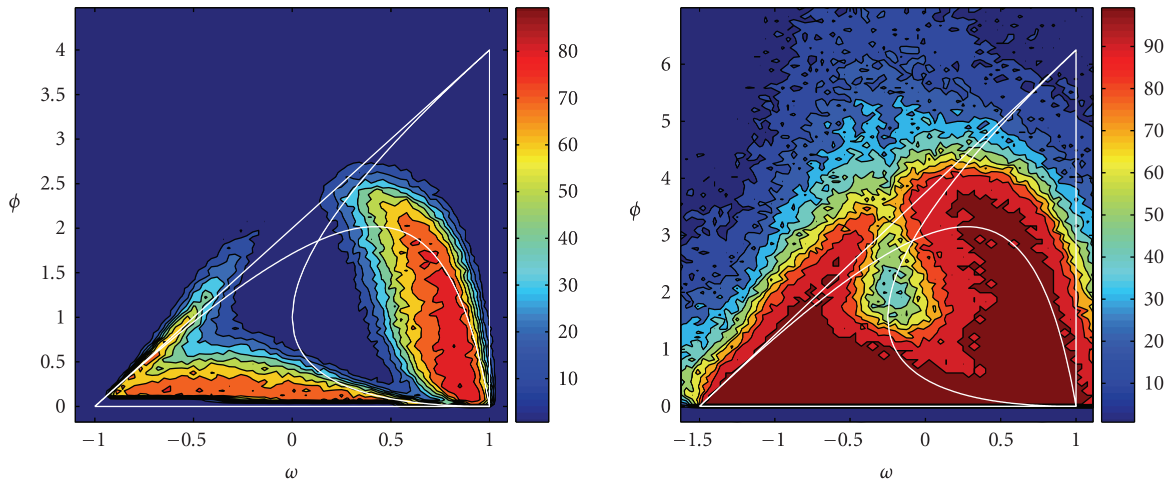
Figure 13 shows the percentage of times (over 100 simulations) that PSO and GPSO (with $\Delta t = 0.5$) arrive very close (within a tolerance of 10^{-5}) to the global optimum after 100 iterations. It can be observed that the convergence zone increases in size and the best parameters move towards decreasing inertia values and increasing accelerations, confirming our previous theoretical analysis. Good parameter sets are close to the limit of stochastic stability region (hyperbola (44)). Only for the elongated DeJong function good



(a) PSO Rosenbrock (b) PSO De Jong (c) PSO Griewank

(d) GPSO $\Delta t = 0.5$ Rosenbrock (e) GPSO $\Delta t = 0.5$ De Jong (f) GPSO $\Delta t = 0.5$ Griewank

Figure 13: Percentage of success for different functions for PSO and GPSO with $\Delta t = 0.5$.



(a) Rosenbrock PSO (b) Rosenbrock GPSO $\Delta t = 0.8$

Figure 14: Percentage of success for the Rosenbrock function in $\mathbb{R}^{10}$ for PSO and GPSO with $\Delta t = 0.8$.

parameter sets are partly above this line. This parameter region has also been pointed by Clerc [11] in the PSO case, under the stagnation hypothesis, and only for positive inertia values.

For the Griewank function parameters between the median of the deterministic triangle and the hyperbola (44), and inertia values around -0.5 (in the PSO case) give also a very good performance. In the GPSO case, the inertia values are shifted $-1/\Delta t$. Negative PSO inertia values have given good results in the PSO case for cost functions with multiple minima [9].

Figure 14 shows the same numerical analysis for the Rosenbrock function defined in $\mathbb{R}^{10}$, both in the PSO ($\alpha = 1$) and GPSO cases ($\Delta t = 0.8$ and $\alpha = 1$), under the same tolerance requirements. As the number of dimensions increases, the best parameter sets seem to fit better the limit of the second-order stability. Also, the region of negative inertia values seems to be more important in size than in dimension

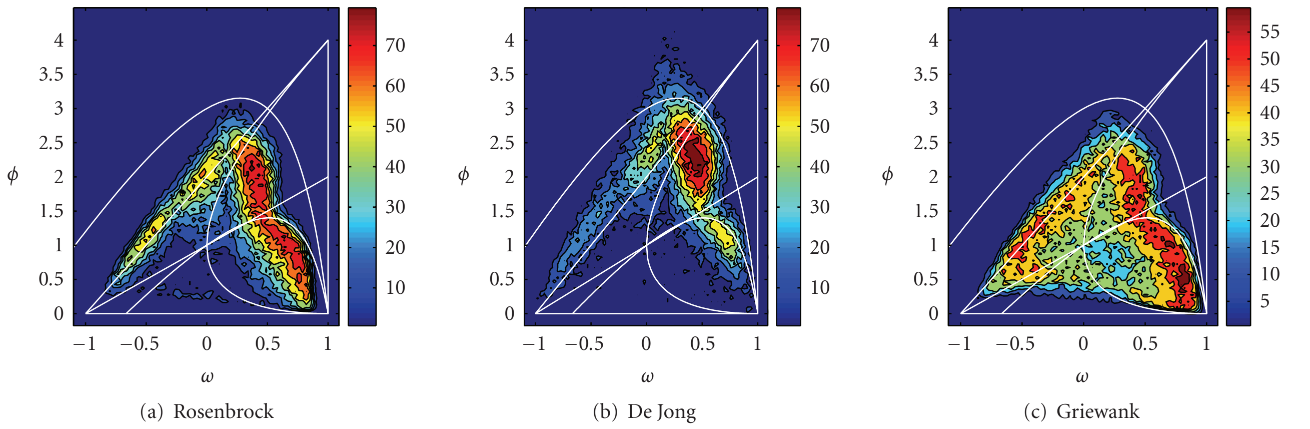


FIGURE 15: Percentage of success for the De Jong, Rosenbrock, and Griewank functions for the mixed GPSO algorithm.

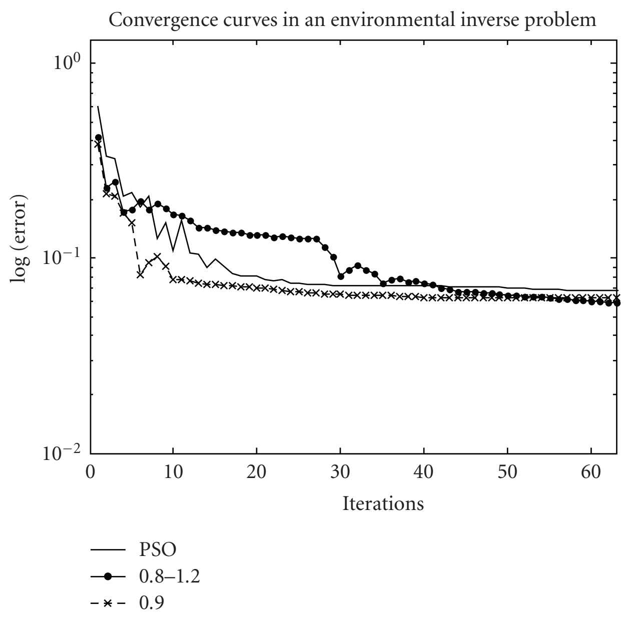


FIGURE 16: Vertical electrical sounding case (optimization in $\mathbb{R}^{11}$). Logarithmic convergence curves for $\Delta t = 0.9,\ 1.0,\ 0.8$–$1.2$.

2. In the GPSO case, the "good" parameter region increases its size.

7.1. Time-step adapted GPSO algorithms

As a result of the above-mentioned analysis and numerical experiments, we propose the following algorithm.

(1) To specify the search space limits, and to initialize swarm positions randomly within, velocities are initialized to zero and are not limited.

(2) To choose the $(w, \overline{\phi})$ parameters for the standard PSO ($\Delta t = 1$), as a general rule, a good choice is an inertia value ω in the range $(0.5, 0.8)$ and a total acceleration given by (44). For cost functions with multiple local minima, inertia values in $(-0.6, -0.5)$ and total acceleration following the same rule (44) seem to give also very good results.

(3) At the exploration stage, the local acceleration ϕ_2 should be bigger than the global term, ϕ_1 ($\alpha < 1$).

(4) At the convergence stage, we propose two different variants with respect to the PSO case.

(a) GPSO algorithm with $\Delta t < 1$

The idea is to monitor the percentage of models having at least one parameter on the lower or upper limit of the model search space as a function of iterations. If this percentage decreases significantly (which is expected mainly in the convergence stage), the time step Δt is reduced (e.g., $\Delta t = 0.8 - 0.9$) in order to adapt the PSO amplitudes to locate accurately the minima. Numerical experiments have shown that at the exploration stage it is common that some model parameters reach the search space limits. Nevertheless, this circumstance does not necessarily imply a time-step reduction. On the contrary, at the convergence stage it is important to adapt the PSO amplitudes in order to efficiently locate the optimum.

(b) Mixed GPSO algorithm

This algorithm consists in adopting $\Delta t = 1.2$ and 0.8 alternatively for the odd and even iterations. When $\Delta t = 1.2$ the exploration capabilities increase (higher variance to avoid local minima) and for $\Delta t = 0.8$ the search is done more accurately around the global best.

We ran several numerical experiments with the Rosenbrock, DeJong, and Griewank functions. Figure 15 shows the percentage of times (over 100 simulations) that mixed GPSO arrives very close (within a tolerance of 10^{-5}) to the global optimum after 100 iterations. It can be observed that good parameter regions are close to both hyperbolas of the second-order stability ($\Delta t = 1.2$ and $\Delta t = 0.8$).

8. APPLICATION TO AN ENVIRONMENTAL REAL CASE

The vertical electrical sounding (VES) is a geophysical direct current technique aiming to characterize the depth-variation of the resistivity distribution of a stratified earth. The VES methodology is as follows.

(i) On the surface, at each measuring station, two current electrodes and two potential electrodes are symmetrically laid at both sides of a fixed central point.

(ii) In successive stations, the two external current electrodes are moved apart from each other, increasing their mutual distance, while holding the two inner potential electrodes fixed in place at a much shorter distance. At each position of injection the voltage difference, ΔV, is then measured.

The inverse problem consists in estimating the conductivity and thicknesses of the stratified earth that explains the voltage measurements made on the surface. One, then, needs to define a misfit function to quantify the distance between observations and predictions issued from the earth models. This geophysical inverse problem is nonlinear and ill-posed, that is, different sets of earth models may give similar voltage predictions. In fact, the VES objective function has the minima in a very narrow and flat valley [14], similar to the Rosenbrock case in several dimensions.

Figure 16 shows the convergence curves (logarithmic error versus iterations) for a VES geophysical inverse problem having an important environmental application (salt intrusion in a coastal aquifer), using different GPSO versions. The cost function is in this case defined in $\mathbb{R}^{11}$. As it can be observed, GPSO with $\Delta t = 0.9$ performs better than PSO from the first iterations. The mixed GPSO algorithms perform more slowly, nevertheless, it is the PSO variant that reaches the lower misfit. These results obviously depend on the search space size.

9. CONCLUSIONS

A new PSO algorithm, generalized PSO, has been presented and analyzed. It comes from the continuous PSO model by considering time steps different from the unit (PSO). Its deterministic and stochastic stability regions and their respective asymptotic velocities of convergence are also a generalization of those of the PSO case. A comparison to PSO and to the continuous model is done, confirming that GPSO properties tend to the continuous as time step decreases. The size of the time steps is thus presented as a numerical constriction factor. Properties of the second-order moments—variance and covariance—serve to propose some promising parameter sets. High variance and temporal uncorrelation improve the exploration task while solving ill-posed inverse problems.

Comparisons between PSO and GPSO, by means of some numerical experiments using well-known benchmark functions and real data issued from environmental inverse problems, show that generalized PSO is more effective than PSO at accurately locating the global minimum of these cost functions. Based on this analysis, we propose two new time-step adapted PSO variants, GPSO, and mixed GPSO that improve the convergence performance of PSO in a geophysical inverse problem in $\mathbb{R}^{11}$.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, WA, Australia, November-December 1995.

[2] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[3] E. Ozcan and C. K. Mohan, "Particle swarm optimization: surfing the waves," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1939–1944, Washington, DC, USA, July 1999.

[4] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[5] Y.-L. Zheng, L.-H. Ma, L.-Y. Zhang, and J.-X. Qian, "On the convergence analysis and parameter selection in particle swarm optimisation," in *Proceedings of the 2nd International Conference on Machine Learning and Cybernetics (ICMLC '03)*, vol. 3, pp. 1802–1807, Xi'an, China, November 2003.

[6] F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[7] V. Kadirkamanathan, K. Selvarajah, and P. J. Fleming, "Stability analysis of the particle dynamics in particle swarm optimizer," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 245–255, 2006.

[8] B. Brandstätter and U. Baumgartner, "Particle swarm optimization—mass-spring system analogon," *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 997–1000, 2002.

[9] J. L. Fernández Martínez, E. García Gonzalo, and J. P. Fernández Alvarez, "Theoretical analysis of particle swarm trajectories through a mechanical analogy," *International Journal of Computational Intelligence Research*, vol. 4, no. 2, 2008.

[10] S. M. Mikki and A. A. Kishk, "Physical theory for particle swarm optimisation," *Progress in Electromagnetics Research*, vol. 75, pp. 171–207, 2007.

[11] M. Clerc, "Stagnation analysis in particle swarm optimization or what happens when nothing happens," Tech. Rep. CSM-460, Department of Computer Science, University of Essex, Colchester, UK, 2006, http://clerc.maurice.free.fr/pso/.

[12] R. Poli, "The sampling distribution of particle swarm optimisers and their stability," Tech. Rep. CSM-465, Department of Computer Science, University of Essex, Colchester, UK, 2007, http://cswww.essex.ac.uk/technical-reports/2007/csm-465.pdf.

[13] J. P. Fernández Alvarez, J. L. Fernández Martínez, E. García Gonzalo, and C. O. Menéndez Pérez, "Application of the particle swarm optimization algorithm to the solution and appraisal of the vertical electrical sounding inverse problem," in *Proceedings of the 11th Annual Conference of the International Association of Mathematical Geology (IAMG '06)*, Liège, Belgium, September 2006.

[14] J. P. Fernández Alvarez, J. L. Fernández Martínez, and C. O. Menéndez Pérez, "Feasibility analysis of the use of binary genetic algorithms as importance samplers application to a geoelectrical VES inverse problem," *Mathematical Geosciences*, 2008.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 827401, 9 pages
doi:10.1155/2008/827401

Research Article

Novel Orthogonal Momentum-Type Particle Swarm Optimization Applied to Solve Large Parameter Optimization Problems

Jenn-Long Liu and Chao-Chun Chang

Department of Information Management, College of Electrical Engineering and Information Science, I-Shou University, No. 1, Section 1, Syuecheng Road, Kaohsiung County 840, Taiwan

Correspondence should be addressed to Jenn-Long Liu, jlliu@isu.edu.tw)

Received 13 July 2007; Accepted 10 January 2008

Recommended by Jim Kennedy

This study proposes an orthogonal momentum-type particle swarm optimization (PSO) that finds good solutions to global optimization problems using a delta momentum rule to update the flying velocity of particles and incorporating a fractional factorial design (FFD) via several factorial experiments to determine the best position of particles. The novel combination of the momentum-type PSO and FFD is termed as the momentum-type PSO with FFD herein. The momentum-type PSO modifies the velocity-updating equation of the original Kennedy and Eberhart PSO, and the FFD incorporates classical orthogonal arrays into a velocity-updating equation for analyzing the best factor associated with cognitive learning and social learning terms. Twelve widely used large parameter optimization problems were used to evaluate the performance of the proposed PSO with the original PSO, momentum-type PSO, and original PSO with FFD. Experimental results reveal that the proposed momentum-type PSO with an FFD algorithm efficiently solves large parameter optimization problems.

1. INTRODUCTION

Many well-known representative evolutionary computation (EC) techniques, suchas genetic algorithms (GAs) [1–3], genetic programming (GP) [4], evolutionary programming (EP) [5, 6], evolutionary strategies (ESs) [7, 8], and particle swarm optimization (PSO) [9], have emerged as efficient computational tools for solving global optimization problems, especially for large parameter optimization problems (LPOPs). These EC techniques are sufficiently robust when coping with discontinuous, vast multimodal problems or with noisy search spaces since they do not depend on derivatives of objective functions and constraints. Among these techniques, PSO is easily implemented and computationally inexpensive as it has low memory and CPU costs. The PSO methodology involves creation of a population of particles such that each particle is capable of adjusting its flying velocity, according to its flying experience and the best experience of the group. Therefore, each particle in the PSO can be regarded as a cooperating agent.

The original PSO algorithm, developed by Kennedy and Eberhart in 1995 [9], was inspired by the swarm behaviors of flocks of birds and schools of fish. Thus, the PSO method is a swarm intelligence method [10] for solving global optimization problems, and it compares favorably to other evolutionary algorithms [11]. The original PSO is effective in determining optimal solutions in static environments, but performs poorly in locating a changing extremum. In addition, imposing a maximum value $V_{\max}$ is necessary to limit the particle velocity. Shi and Eberhart introduced a parameter called inertia weight (w), which dampens particles' velocities over time, allowing a swarm to converge efficiently and with increased accuracy [12, 13]. Furthermore, Clerc proposed using a constriction factor (K), which improves the ability of PSO to constrain and control velocities [14]. Eberhart and Shi found that K, combined with constraints on the maximum allowable velocity vector ($V_{\max}$), improved the PSO performance significantly [15]. Additionally, when utilizing PSO with a constriction factor setting, the $X_{\max}$ for each dimension is the best approach. Constriction factor (K)

implements velocity control for effectively erasing the tendency of some particles to spiral into ever-increasing velocity oscillations. Parsopoulos and Vrahatis presented an overview of recent approaches for solving global optimization problems using PSOs [16].

This work presents a novel orthogonal momentum-type PSO algorithm for locating the best position for each particle efficiently. The momentum-type PSO, proposed in previous work [17], has a delta momentum rule in the velocity-updating equation, and efficiently determines the physical motion of particles more reasonably than other PSO versions. The momentum-type PSO markedly outperformed the original PSO [9] and modified PSO [13] versions. Furthermore, this study incorporates the momentum-type PSO and well-known fractional factorial analysis, which utilize several factorial experiments, according to classical orthogonal tables, to identify intelligently the best combination of factors from two-level variables to enhance algorithmic search efficiency. The novel combination of the momentum-type PSO and fractional factorial design (FFD) is termed the momentum-type PSO with FFD herein. Ho et al. [18] and Lin [19] developed Taguchi orthogonal tables for their GA and PSO for constructing an orthogonal GA and orthogonal PSO. Their strategies markedly improved the search ability of their GA and PSO. In 1991, Bhote reported that classical fractional factorial analysis combined with evolutionary optimization is superior to the Taguchi method [20]. Consequently, this study proposes orthogonal PSO using the FFD, rather than Taguchi tables, for a velocity-updating equation of momentum-type PSO to analyze the best factor related to cognitive learning and social learning terms with the goal of enhancing algorithmic efficiency. This work also developed the original PSO with FFD, which combines the original Kennedy and Eberhart PSO and classical orthogonal arrays, to enhance the performance of the original PSO. Performance of the original PSO, momentum-type PSO, original PSO with FFD, and proposed momentum-type PSO with FFD were compared using 12 benchmark LPOPs.

2. METHODOLOGY OF THE PROPOSED PARTICLE SWARM OPTIMIZATION

2.1. *Description of an objective function*

The general form of an objective function with variable vector $\vec{x}$ is expressed as $f(\vec{x})$. The variable $\vec{x}$ represents the solution vector with N variables, and is defined as the set $\{x_i,\ i = 1, N\}$. In PSO computation, each particle is assigned a fitness value based on the calculation of objective function. Moreover, the fitness value of a particle determines the best position of each particle over time and determines which particle has the best global value in the current swarm.

2.2. *Momentum-type particle swarm optimization*

The original PSO developed by Kennedy and Eberhart [9] supposed that the ith particle flies over a hyperspace; its position and velocity are given by $\vec{x}_i$ and $\vec{v}_i$, respectively. Initial particle position and velocity are chosen randomly. At time step k, the best position of the current and previous moves for the ith particle is denoted by $p\text{best}_i$; the best particle with the best function value in the swarm is denoted by $g$best. Consequently, the next flying velocity and position of particle i at time step $k+1$ is updated using the following heuristic equations:

$$\begin{aligned}\vec{v}_i^{k+1} &= \vec{v}_i^k + c_1 \times \text{rand}() \times (p\text{best}_i - \vec{x}_i^k) + c_2 \\ &\quad \times \text{rand}() \times (g\text{best} - \vec{x}_i^k),\end{aligned} \tag{1}$$

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \vec{v}_i^{k+1}, \tag{2}$$

where c_1 and c_2 are cognitive and social learning rates, respectively. These two parameters control the relative importance of memory (position) of a particle itself and the neighborhood memory, and are both set to 2.0 [9, 12, 21] for multiplying random values by a mean of 1, such that agents would "overfly" the target about half the time [9]. Random function rand() is uniformly distributed in the range [0,1]. As in (1), the two random values are generated independently, and the velocity vector of a particle is updated based on variations on its current position, its previous best position, and the previous best position of its neighbors. After particle velocity is updated using (1), its new position is updated by adding the velocity vector to the current position. Analyses of algorithmic stability and convergence of the PSO have been examined theoretically by Clerc and Kennedy [22] and by Trelea [23] who used dynamic system theory.

The goal of the momentum-type PSO [17], which includes a delta term for the velocity vector for detecting particle-velocity variation, is to express the physical motion of particles more accurately than current PSO algorithms do. According to [17], the difference between a particle's current position $\vec{x}_i^k$ and $p\text{best}_i$ (as shown in (1)) represents the first external effect on the particle and causes it to move, and the difference between a particle's current position $\vec{x}_i^k$ and $g$best, which represents the second external effect on a particle. The two effects propelling particle i to move can be expressed in delta position forms as follows:

$$\begin{aligned} c_1 \times \text{rand}() \times (p\text{best}_i - \vec{x}_i^k) &= p \times (\Delta\vec{x}_i^k)_1, \\ c_2 \times \text{rand}() \times (g\text{best} - \vec{x}_i^k) &= q \times (\Delta\vec{x}_i^k)_2, \end{aligned} \tag{3}$$

where $p = c_1 \times \text{rand}()$ and $q = c_2 \times \text{rand}()$. Combining the two terms and using the velocity $\vec{v}_i^k$ to represent the variation of positions $(\Delta\vec{x}_i^k)$, the composed velocity $\vec{v}_i^k$ can be obtained as follows:

$$\vec{v}_i^k = p \times (\vec{v}_i^k)_1 + q \times (\vec{v}_i^k)_2. \tag{4}$$

Since $\vec{v}_i^k$ is particle-composed velocity, velocity $\vec{v}_i^{k+1}$ of particle i at next time step $k+1$ may include a delta form for velocity as $\Delta\vec{v}_i^k$, rather than the velocity $(\vec{v}_i^k)$, as employed in the original PSO, the Shi and Eberhart PSO, and many other PSOs. Generally, $\Delta\vec{v}_i^k$ denotes the difference of velocities between time steps $k+1$ and k, that is, $\Delta\vec{v}_i^k = \vec{v}_i^{k+1} - \vec{v}_i^k$. Thus,

Table 1: Orthogonal table with seven factors.

Factor	A	B	AB	C	AC	BC	ABC	Output
Factor level	$X_1^{\pm}$	$X_2^{\pm}$	$X_3^{\pm}$	$X_4^{\pm}$	$X_5^{\pm}$	$X_6^{\pm}$	$X_7^{\pm}$	
Experiment 1	−	−	+	−	+	+	−	f_1
2	+	−	−	−	−	+	+	f_2
3	−	+	−	−	+	−	+	f_3
4	+	+	+	−	−	−	−	f_4
5	−	−	+	+	−	−	+	f_5
6	+	−	−	+	+	−	−	f_6
7	−	+	−	+	−	+	−	f_7
8	+	+	+	+	+	+	+	f_8
Contribution	C_1	C_2	C_3	C_4	C_5	C_6	C_7	
Select level	+ or −	+ or −	+ or −	+ or −	+ or −	+ or −	+ or −	
Best factor	X_1^+ or X_1^-	X_2^+ or X_2^-	X_3^+ or X_3^-	X_4^+ or X_4^-	X_5^+ or X_5^-	X_6^+ or X_6^-	X_7^+ or X_7^-	

the momentum-type form for velocity updating with parameter β can be written as

$$\vec{v}_i^{k+1} = \vec{v}_i^k + \beta \times \Delta\vec{v}_i^k, \tag{5}$$

where β is a positive number ($0 \le \beta < 1$) termed the momentum constant, which controls velocity vector rate of change. The momentum constant has characteristics that are reminiscent of the stabilizing effect in backpropagation neural networks [24] and accelerating convergence in numerical algorithms [25]. Similarly, position $\vec{x}_i^{k+1}$ of particle i at time step $k+1$ can be written as the current position $\vec{x}_i^k$ plus a delta form of the position ($\Delta\vec{x}_i^k$):

$$\vec{x}_i^{k+1} = \vec{x}_i^k + \alpha \times \Delta\vec{x}_i^k. \tag{6}$$

Parameter α is another momentum constant for adjusting the rate of change of particle position. Consequently, (5) and (6) correspond to each other, and both are consistent with the physical behavior for variations in velocity and position of particle motion. Hence, (5) and (6) can be combined to form a matrix of the generalized delta rule:

$$\vec{S}_i^{k+1} = \vec{S}_i'^{k} + M\Delta\vec{S}_i^k, \tag{7}$$

where

$$\vec{S} = \begin{bmatrix} \vec{v} \\ \vec{x} \end{bmatrix}, \qquad \vec{S}' = \begin{bmatrix} \vec{V} \\ \vec{x} \end{bmatrix}, \qquad M = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{8}$$

Equation (7) allows each particle the ability of dynamic self-adaptation in the search space over time, that is, the ith particle can memorize the previous velocity variation state and automatically adjust the next velocity value during movement. The delta momentum term in the PSO algorithm stabilizes searching, and an appropriate value for $\beta = 0.1$ improves convergence and the optimal solution [17]. Parameter α was set to 1 such that variable v could be interpreted as true velocity, that is, the change between two successive particle positions [23]. Thus (7) satisfies the zero-velocity variation ($\Delta\vec{v}_i = 0$) and zero velocity ($\vec{v}_i = 0$) conditions for the particle considered.

2.3. *Proposed momentum-type PSO with fractional factorial design*

This study integrates FFD into the momentum-type PSO to determine the combination of factors associated with cognitive learning and social learning terms. This study defined the two states of orthogonal momentum-type PSO as follows:

$$\begin{aligned} \vec{X}_i^+ &= \vec{x}_k^i + \beta \times \Delta\vec{v}_i^k + p \times (\vec{v}_i^k)_1, \\ \vec{X}_i^- &= \vec{x}_k^i + \beta \times \Delta\vec{v}_i^k + q \times (\vec{v}_i^k)_2, \end{aligned} \tag{9}$$

where $\vec{X}_i^+$ and $\vec{X}_i^-$ indicate two possible new positions related to cognitive learning and social learning for particle i. Thus, updating rules for velocity and position using the proposed momentum-type PSO with FFD can be expressed as

$$\vec{v}_i^{k+1} = \left(\vec{X}_i^+ \text{ OR } \vec{X}_i^-\right) - \vec{x}_i^k, \tag{10}$$

$$\vec{x}_i^{k+1} = \left(\vec{X}_i^+ \text{ OR } \vec{X}_i^-\right). \tag{11}$$

The expression ($\vec{X}_i^+$ OR $\vec{X}_i^-$) represents the implementation of OR operator via fractional factorial analysis using variables $\vec{X}_i^+$ and $\vec{X}_i^-$. Therefore, the new flying position of particle i is determined by (11), namely, the best position of particle i on the next time $k+1$ can be obtained using FFD.

2.4. *Fractional factorial design (FFD)*

To solve the operator ($\vec{X}_i^+$ OR $\vec{X}_i^-$) shown in (10) and (11), the FFD used in this work is based on classical full factorial analysis with two-level, multivariable orthogonal tables [20]. Applying the fractional factorial analysis to the operator ($\vec{X}_i^+$ OR $\vec{X}_i^-$) yields an arrangement of two-level factors that corresponds to the two possible states of the design variables. In the classical factorial experiments, the two levels of a design variable are labeled by "−" and "+". Hence, X_i^- and X_i^+ represent the two levels of the design variable X_i in this work. Table 1 lists the level distribution of seven factors with the same number of levels "−" and "+" at each column to retain the balance of a statistically designed experiment. The detailed steps of the numerical procedure for the fractional factorial analysis are as follows.

(1) First, if N factors (termed by design variables in the PSO) each has two levels, build a table of m rows and $m-1$ columns. The value m is defined by the integer $2^{[\log_2(N+1)]}$. For example, if seven factors ($N = 7$) are associated with vector $\vec{X}$, then eight experiments ($m = 8$) are conducted for factorial analysis. As shown in Table 1, the levels "$-$" and "+" in columns A, B, and C are assigned to each cell position according to the classical full factorial principle [20]. In columns AB, AC, BC, and ABC, each level in the cells is determined from the inner products of columns A and B, columns A and C, columns B and C, and columns A, B, and C.

(2) The first experiment has a factor set to $\vec{X} = \{X_1^-, X_2^-, X_3^+, X_4^-, X_5^+, X_6^+, X_7^-\}$, obtained by combining the factors A to ABC with the assigned levels. In the following, its objective function, f_1, is computed, and put it into the first position of column "Output". Repeat the computations for the remaining seven experiments with function values $f_2, f_3, \dots,$ and f_8; then, the eight experiments for seven factors are finished.

(3) In column A, multiply the function values $f_1, f_2, \dots,$ and f_8 by the corresponding algebraic value -1 for level "$-$," and 1 for level "+," as listed in Table 1. The effect of factor A on the level is determined by adding the eight products together. The mathematical form for each factor i is $C_i = \sum_{j=1}^{8} U_{i,j} \times f_j$. Here, $U_{i,j}$ equals -1 when the cell level is at level "$-$," and equals 1 when it is at level "+". The summation C_1 is placed in the first position in the row "Contribution" which is associated with factor A. Repeat the multiplication and summation operations for the remaining six values, $C_2, C_3, \dots,$ and C_7, and put them in corresponding positions in the row "Contribution"; the effects of factors B, C,. . . , and ABC on the levels are thus obtained.

(4) Check the signs of the seven values C_1, $C_2, \dots,$ and C_7 listed in the row "Contribution"; if the sign of C_i is negative, then place the symbol "$-$" in the row "Selected level" (Table 1). Otherwise, select symbol "+". The dominant levels of the seven factors are thus determined.

(5) The best combination of X_i^* for the seven factors is determined from the factors with selected levels, presented in the row "Best factor" (Table 1), and the best value of the function is obtained by calculating the objective function $f(\vec{X}^*)$. The choice principle for each best factor X_i^* is expressed as follows:

$$X_i^* = \begin{cases} X_i^- & \text{if } \operatorname{sign}(C_i) < 0 \\ X_i^+ & \text{if } \operatorname{sign}(C_i) > 0. \end{cases} \tag{12}$$

Thus the new position x_i^{k+1} of the particle i at time step $k+1$ shown in (11) can be set equal to X_i^*.

3. RESULTS AND DISCUSSION

3.1. Effect of number of particles

This work applied FFD to several factorial experiments to determine the best position of particles. When the dimension of design variables is high, that is, a large parameter optimization problem, FFD usually required numerous function evaluations for factorial analysis. Therefore, few particles for the proposed PSO with FFD should reduce CPU cost if the good performance of searching of the PSO can be retained. To determine how many particles are appropriate when using the momentum-type PSO with FFD and original PSO with FFD algorithms, adequate size of N_{particle} is investigated by conducting the following large parameter optimization problem with low and high dimensions:

$$\text{maximize} f(\vec{x}) = -\sum_{i=1}^{N}\left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right] \quad \text{subject to } x_i \in [3, 13]. \tag{13}$$

This function is a multimodal problem and has an analytical optimum of $1.216N$. In this computation, dimensions were set at 10 and 100, and the maximum numbers of function evaluation for the two cases were set at 10000 and 100000, respectively. Moreover, the number of particles was 5, 10, 15, 20, 25, and 30 to investigate the effects of different swarm sizes of particles to orthogonal PSOs. Figures 1(a) and 1(b) reveal that the proposed momentum-type PSO with the FFD algorithm and few particles achieved a favorable convergence rate and numerical accuracy; nevertheless, many particles resulted in poor convergence. Using the proposed momentum-type PSO with FFD and few particles for $N = 10$ and $N = 100$ cases rapidly converges to optimal values 12.1598 and 121.598, respectively. In this study, numerical performance using $N_{\text{particle}} = 5$ is better than other cases using large number sizes of particles. Similar results shown in Figures 2(a) and 2(b) are observed when using the original PSO with the FFD technique. Accordingly, this work set $N_{\text{particle}} = 5$ for the momentum-type PSO with FFD and the original PSO with FFD algorithms in the following computations.

3.2. Experiments for 12 large parameter optimization problems (LPOPs)

Twelve benchmark LPOPs presented in [18] are evaluated to further examine the performance of the proposed momentum-type PSO with FFD. This study sets $N_{\text{particle}} = 5$ for the original PSO with FFD and the proposed momentum-type PSO with FFD, and $N_{\text{particle}} = 30$ for the original PSO and momentum-type PSO. The number of dimensions for all benchmark LPOPs was set to 100, and the terminated number of function evaluations for the four PSO algorithms was set to 100000. Twenty independent runs were conducted for each problem. Algorithmic performance was assessed by the best solution, averaged best solution, mean absolute error (MAE, and standard deviation. The MAE was used as the measurement of error in this work with the following form:

$$\text{MAE} = \frac{|f_{\text{opt}} - f_{\text{avg}}|}{N}. \tag{14}$$

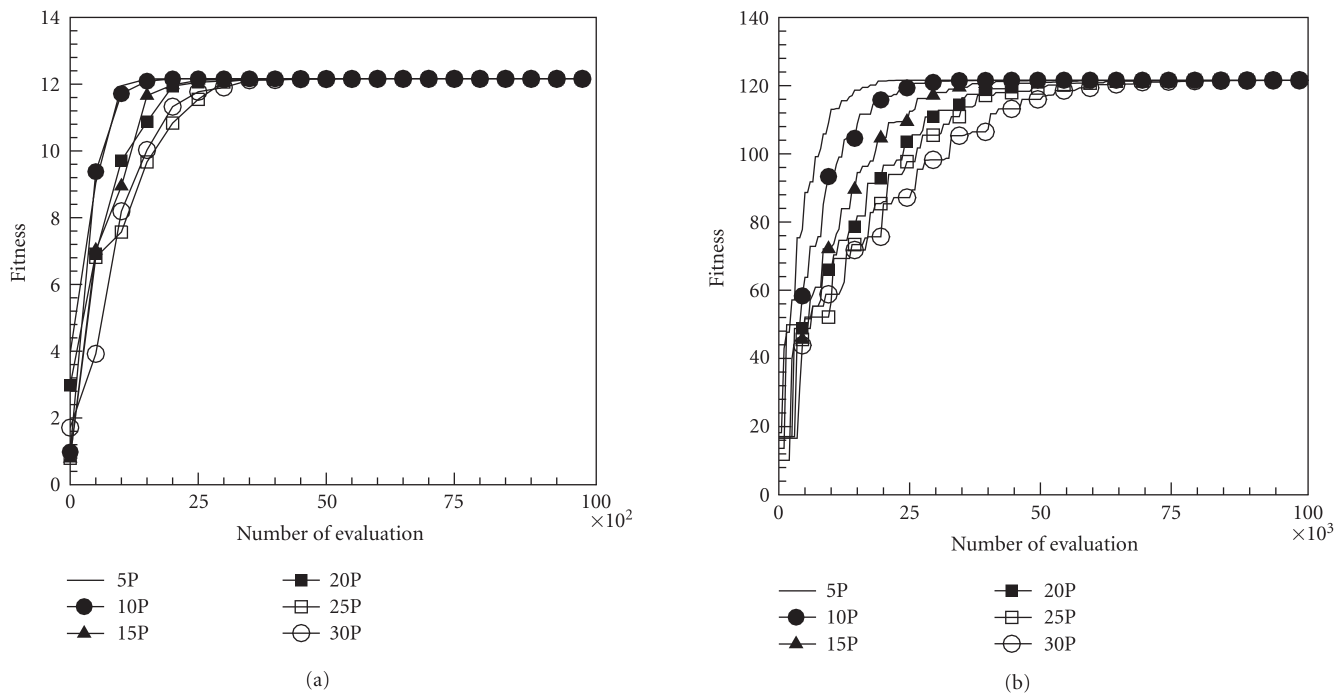


Figure 1: Effect of number of particles using the momentum-type PSO with FFD for solving the LPOP at (a) $N = 5$ and (b) $N = 100$ dimensions.

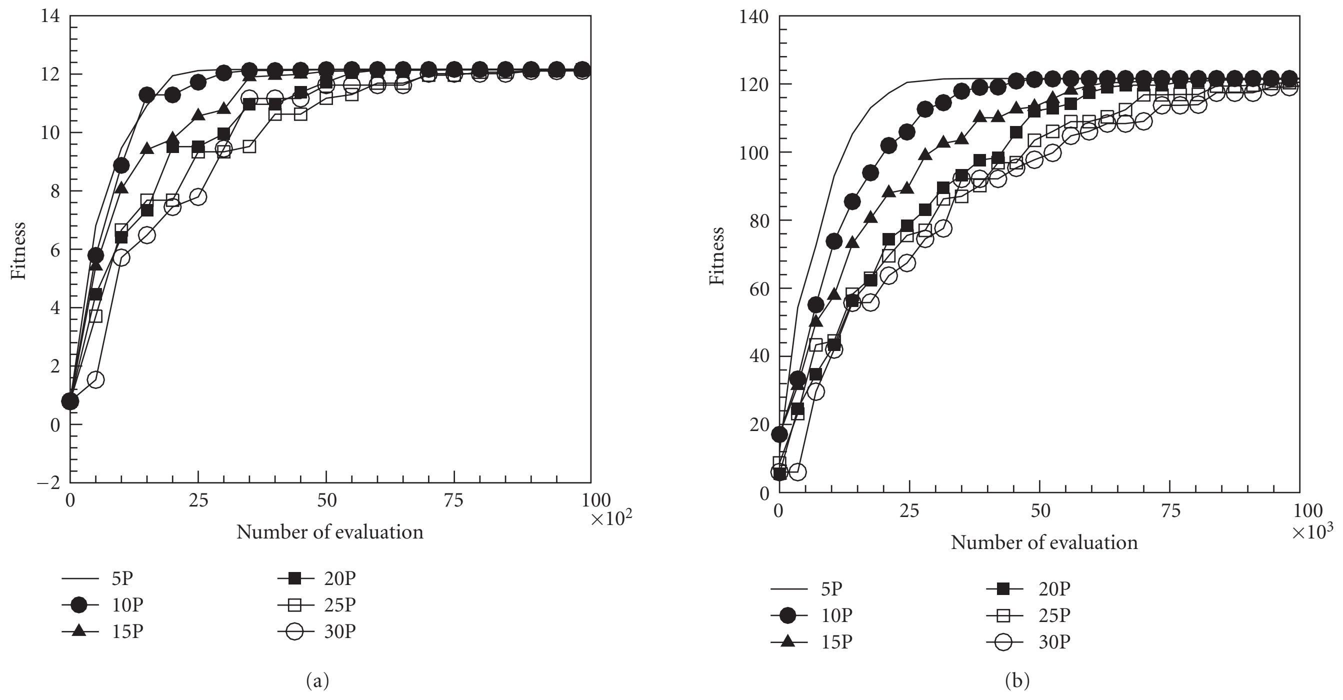


Figure 2: Effect of number of particles using the original PSO with FFD for solving the LPOP at (a) $N = 5$ and (b) $N = 100$ dimensions.

Table 2 lists the 12 objective functions with range constraints and their global optima [18]. Notably, the objectives of problems 1–3 are set to maximize the objective functions $f1$–3, and other problems are set to minimize the objective functions $f4$–12. The optima of the first three maximal functions $f1$, $f2$, and $f3$ are 121.598, 200, and 185, respectively, and the minimal functions $f4$–12 are all zero. Table 3 lists the computational results for the 12 functions at a dimension of $N = 100$ using the four PSOs. Computational results reveal that the evaluation results for maximal functions

Table 2: Twelve objective functions with range constraints and their optima.

Problems	Objective functions and ranges	Optima
1	Maximize $f1 = \sum_{i=1}^{N}\left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right];\ x_i \in [3, 13]$	$1.21598N$
2	Maximize $f2 = \sum_{i=1}^{N-1}\left[\sin(x_i + x_{i+1}) + \sin\left(\frac{2x_i x_{i+1}}{3}\right)\right];\ x_i \in [3, 13]$	$\approx 2N$
3	Maximize $f3 = \sum_{i=1}^{N}[x_i \sin(10\pi x_i)];\ x_i \in [-1, 2]$	$1.85N$
4	Minimize $f4 = \sum_{i=1}^{N}[x_i + 0.5]^2;\ x_i \in [-100, 100]$	0
5	Minimize $f5 = \sum_{i=1}^{N}[x_i^2 - 10\cos(2\pi x_i) + 10];\ x_i \in [-5.12, 5.12]$	0
6	Minimize $f6 = \sum_{i=1}^{N}[x_i^2];\ x_i \in [-5.12, 5.12]$	0
7	Minimize $f7 = \sum_{i=1}^{N}\left[\frac{\sin(10x_i\pi)}{10x_i\pi}\right];\ x_i \in [-0.5, 0.5]$	0
8	Minimize $f8 = -20\exp\left[-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right] - \exp\left[\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right] + 20 + e;\ x_i \in [-30, 30]$	0
9	Minimize $f9 = 418.9828N - \sum_{i=1}^{N}(x_i \sin(\sqrt{\mid x_i \mid}));\ x_i \in [-500, 500]$	0
10	Minimize $f10 = \sum_{i=1}^{N-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2];\ x_i \in [-5.12, 5.12]$	0
11	Minimize $f11 = 6N + \sum_{i=1}^{N}\lfloor x_i \rfloor;\ x_i \in [-5.12, 5.12]$	0
12	Minimize $f12 = \frac{1}{4000}\sum_{i=1}^{N}x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1;\ x_i \in [-600, 600]$	0

$f1$, $f2$, and $f3$ using the proposed momentum-type PSO with FFD were 121.598, 174.452, and 182.798, respectively. These computational results are better than those computed by the other three PSOs. Additionally, the algorithmic performance of the PSOs with FFD is superior to that of PSOs without FFD. Consequently, the proposed PSO with FFD promoted solution searches more effectively than the PSO does without FFD. For the remaining nine minimal function evaluations, $f4$–12, the optima obtained using the proposed momentum-type PSO with FFD were 0.0, 11.946, 0.0, 0.00005, 0.000039, 3781.879, 92.874, 88.0, and 0.000059, respectively, and the average solutions were 0.0, 32.4641, 0.0, 0.0001, 0.0139, 5676.0005, 181.8753, 88.2285, and 0.1009, respectively. Both the best and average results are the best when compared with solutions obtained by the other three PSOs. Additionally, all MAEs and standard deviations are lowest for the nine optimization functions evaluated by the proposed momentum-type PSO with FFD over 20 independent runs. From the computational results (Table 3), the proposed momentum-type PSO with FFD performed well in solving LPOPs. Figures 3(a)–3(l) represent the convergence histories of the 12 objection functions obtained by the four PSOs. The proposed momentum-type PSO is superior to the other PSOs in terms of convergence rate and optimal solution (Figures 3(a)–3(c)). Although the convergence rate of the original PSO is relatively poor for solving LPOPs, it can be significantly improved by introducing FFD into the PSO algorithm. Figures 3(d)–3(l) present results similar to the maximization cases.

4. CONCLUSION

This study examined the performance of the proposed momentum-type PSO with the FFD algorithm, which uses the delta momentum rule to update particle velocity and FFD to locate a particle's best location for handling LPOPs. Using the momentum-type PSO, each particle can adjust its velocity vector according to its previous velocity change rate. The new location of a particle is then determined by analyzing the best combination of cognitive and social learning terms via several factorial experiments based on the FFD. Moreover, this work modified the original PSO by incorporating with FFD, which also employs the classical orthogonal tables, and improved significantly the efficiency of the original Kennedy and Eberhart PSO. Through 12 benchmark function assessments, the proposed momentum-type PSO algorithm

Table 3: Results obtained using the original PSO, momentum-type PSO, original PSO with FFD, and momentum-type PSO with FFD algorithms to solve the 12 benchmark LPOPs.

$f(x)$	Algorithms	f_{opt}	f_{best}	f_{avg}	$\lvert f_{\text{opt}} - f_{\text{avg}}\rvert$	MAE	SD (20 runs)
$f1$	Original PSO	121.598	26.5864	15.3709	106.2271	1.0623	5.8322
	Momentum-type PSO		121.5966	119.6452	1.9528	0.0195	3.2968
	Original PSO with FFD		121.5730	120.6245	0.9735	0.0097	2.7499
	Momentum-type PSO with FFD		121.5980	120.3738	1.2242	0.0122	2.2866
$f2$	Original PSO	200	41.3325	35.9429	164.0571	1.6406	2.9219
	Momentum-type PSO		147.5408	135.8634	64.1366	0.6414	8.0647
	Original PSO with FFD		163.7965	150.1892	49.8108	0.4981	6.3295
	Momentum-type PSO with FFD		174.4516	166.6228	33.3772	0.3338	6.0346
$f3$	Original PSO	185	41.7656	36.0755	148.9245	1.4892	5.6901
	Momentum-type PSO		128.7731	116.8754	68.1246	0.6812	11.8977
	Original PSO with FFD		165.4799	140.0511	44.9489	0.4495	25.4287
	Momentum-type PSO with FFD		182.7978	174.9624	10.0376	0.1004	7.8354
$f4$	Original PSO	0	1.3102	462.9955	462.9955	4.6300	461.6853
	Momentum-type PSO		0.0001	1.9415	1.9415	0.0194	1.9413
	Original PSO with FFD		3.6828	9.2072	9.2072	0.0921	5.5244
	Momentum-type PSO with FFD		0.0000	0.0000	0.0000	0.0000	0.0000
$f5$	Original PSO	0	1685.8392	1774.6315	1774.6315	17.7463	88.7923
	Momentum-type PSO		408.6802	565.3514	565.3514	56.5351	156.6713
	Original PSO with FFD		35.6695	76.3567	76.3567	0.7636	40.6872
	Momentum-type PSO with FFD		11.9463	32.4641	32.4641	0.3246	20.5178
$f6$	Original PSO	0	52.4517	186.2910	186.2910	1.8629	133.8393
	Momentum-type PSO		0.0000	1.3711	1.3711	0.0137	1.3711
	Original PSO with FFD		0.0000	0.0000	0.0000	0.0000	0.0000
	Momentum-type PSO with FFD		0.0000	0.0000	0.0000	0.0000	0.0000
$f7$	Original PSO	0	0.0001	0.0001	0.0001	0.0000	0.0000
	Momentum-type PSO		0.0001	0.0211	0.0211	0.0002	0.0210
	Original PSO with FFD		0.0001	0.0001	0.0001	0.0000	0.0000
	Momentum-type PSO with FFD		0.0001	0.0001	0.0001	0.0000	0.0000
$f8$	Original PSO	0	0.4069	3.9570	3.9570	0.0396	3.5501
	Momentum-type PSO		0.0520	1.3377	1.3377	0.0134	1.2857
	Original PSO with FFD		0.0000	0.6476	0.6476	0.0065	0.6476
	Momentum-type PSO with FFD		0.0000	0.0139	0.0139	0.0001	0.0139
$f9$	Original PSO	0	28122.4766	29843.7729	29843.7729	298.4377	1721.2964
	Momentum-type PSO		7731.2314	9897.2021	9897.2021	98.9720	2165.9706
	Original PSO with FFD		5363.7783	6583.7198	6583.7198	65.8372	1219.9415
	Momentum-type PSO with FFD		3781.8799	5676.0005	5676.0005	56.7600	1894.1206
$f10$	Original PSO	0	577.8867	64444.4250	64444.4250	644.4443	63866.5383
	Momentum-type PSO		373.7911	635.0241	635.0241	6.3502	261.2330
	Original PSO with FFD		341.0312	1292.7659	1292.7659	12.9277	951.7347
	Momentum-type PSO with FFD		92.8743	181.8753	181.8753	1.8188	89.0009
$f11$	Original PSO	0	231.3600	319.4241	319.4241	3.1942	88.0640
	Momentum-type PSO		88.0000	89.7067	89.7067	0.8971	1.7067
	Original PSO with FFD		88.0000	89.3653	89.3653	0.8937	1.3653
	Momentum-type PSO with FFD		88.0000	88.2285	88.2285	0.8823	0.2285
$f12$	Original PSO	0	0.2118	171.7896	171.7896	1.7179	171.5778
	Momentum-type PSO		0.0183	3.2176	3.2176	0.0322	3.1992
	Original PSO with FFD		0.8876	7.6815	7.6815	0.0768	6.7939
	Momentum-type PSO with FFD		0.0001	0.1009	0.1009	0.0010	0.1008

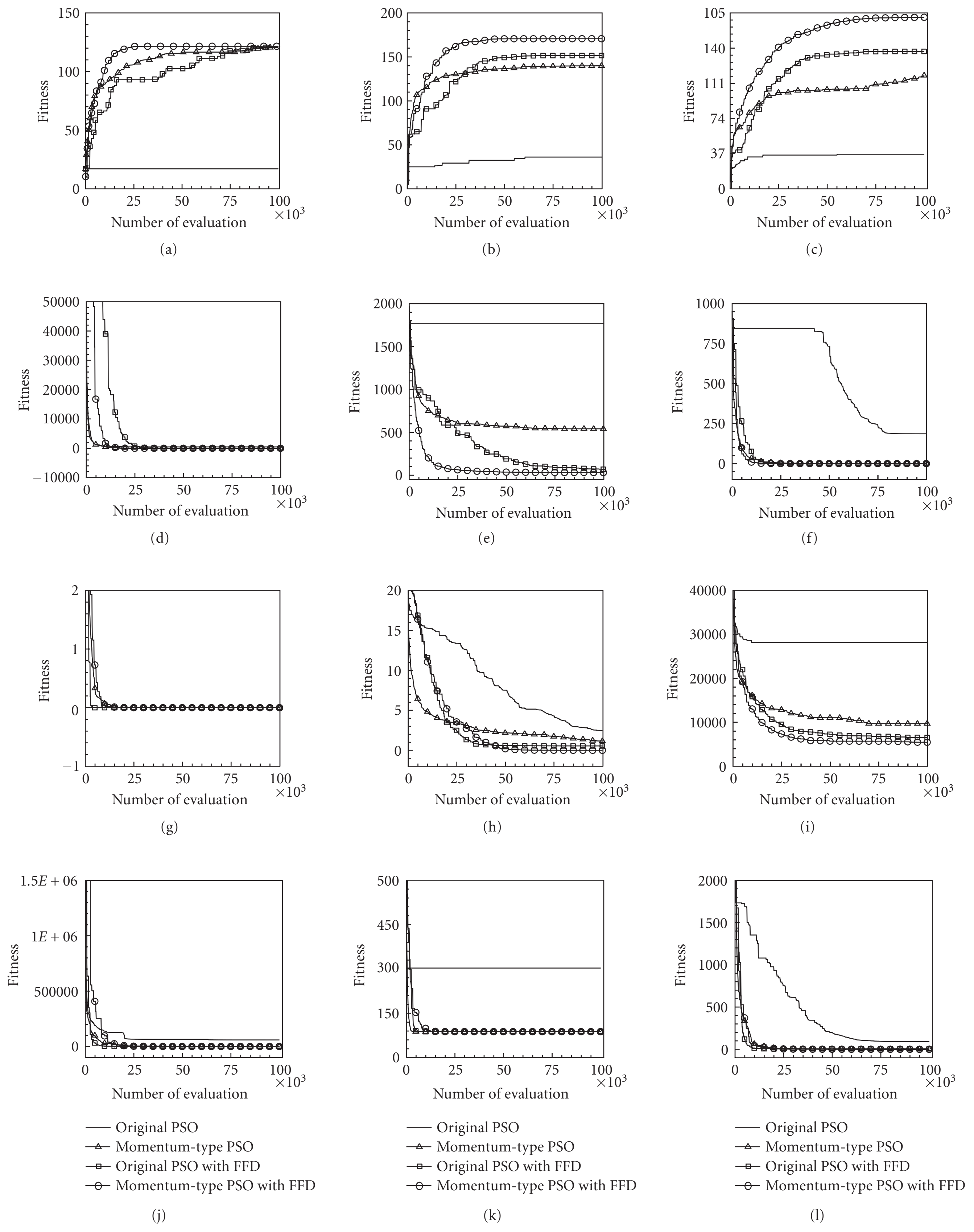


Figure 3: Evaluated results obtained using the four PSOs for solving functions (a) $f1$, (b) $f2$, (c) $f3$, (d) $f4$, (e) $f5$, (f) $f6$, (g) $f7$, (h) $f8$, (i) $f9$, (j) $f10$, (k) $f11$, and (l) $f12$.

outperformed the original PSO, momentum-type PSO and original PSO with FFD in terms of solution accuracy, numerical error and standard deviation. This work also determined that algorithmic performance of the proposed PSO with FFD is superior to that without FFD. Numerical results show that the proposed momentum-type PSO with FFD algorithm is efficient and is a promising method for solving LPOPs.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program*, Springer, Berlin, Germany, 1999.

[4] J. R. Koza, *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Mass, USA, 1992.

[5] L. J. Fogel, "Evolutionary programming in perspective: the top-down view," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds., pp. 135–146, IEEE Press, Piscataway, NJ, USA, 1994.

[6] D. B. Fogel, "An overview of evolutionary programming," in *Evolutionary Algorithms*, L. Davis, K. De Jong, M. Vose, and L. D. Whitley, Eds., IMA Volume in Mathematics and Its Applications, pp. 89–109, Springer, Berlin, Germany, 1999.

[7] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91)*, R. K. Below and L. B. Booker, Eds., pp. 2–9, Morgan Kaufmann, San Diego, Calif, USA, July 1991.

[8] I. Rechenberg, "Evolution strategy," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds., IEEE Press, Piscataway, NJ, USA, 1994.

[9] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, WA, Australia, November-December 1995.

[10] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[11] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 611–616, San Diego, Calif, USA, March 1998.

[12] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1945–1950, Washington, DC, USA, July 1999.

[13] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[14] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *In Proceedings of the Congress of Evolutionary Computation (CEC '99)*, vol. 3, pp. 1951–1957, Washington, DC, USA, July 1999.

[15] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC '00)*, vol. 1, pp. 84–88, La Jolla, Calif, USA, July 2000.

[16] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, no. 2-3, pp. 235–306, 2002.

[17] J.-L. Liu and J.-H. Lin, "Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization," *Engineering Optimization*, vol. 39, no. 3, pp. 287–305, 2007.

[18] S.-Y. Ho, L.-S. Shu, and J.-H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 6, pp. 522–541, 2004.

[19] H. S. Lin, "Design of a novel particle swarm optimization," M.S. thesis, Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, June 2004.

[20] K. R. Bhote, *World Class Quality: Using Design of Experiments to Make it Happen*, American Management Association Press, New York, NY, USA, 1991.

[21] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "Practical distribution state estimation using hybrid particle swarm optimization," in *Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference*, vol. 2, pp. 815–820, Columbus, Ohio, USA, January-February 2001.

[22] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[23] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[25] G. J. Borse, *Numerical Methods with MATLAB*, PWS Publishing, Boston, Mass, USA, 1997.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 654184, 10 pages
doi:10.1155/2008/654184

Research Article

A Simplified Recombinant PSO

Dan Bratton and Tim Blackwell

Department of Computing, Goldsmiths, University of London, New Cross, London SE14 6NW, UK

Correspondence should be addressed to Dan Bratton, d.bratton@gold.ac.uk

Received 20 July 2007; Accepted 3 December 2007

Recommended by Riccardo Poli

Simplified forms of the particle swarm algorithm are very beneficial in contributing to understanding how a particle swarm optimization (PSO) swarm functions. One of these forms, PSO with discrete recombination, is extended and analyzed, demonstrating not just improvements in performance relative to a standard PSO algorithm, but also significantly different behavior, namely, a reduction in bursting patterns due to the removal of stochastic components from the update equations.

1. INTRODUCTION

Originally conceived as a modification to the standard PSO algorithm for use on self-reconfigurable adaptive systems used in on-chip hardware processes, PSO with discrete recombination (PSO-DR) introduces several appealing and effective modifications, resulting in a simpler variant of the original [1]. It is one of the more interesting advances in PSO research over the last few years because these simplifications apparently do not degrade performance yet they remove various issues associated with the stochasticity of the PSO acceleration parameters that hinder theoretical analysis of PSO.

Physical creation of hardware-based optimizers is a substantially more intricate undertaking than software implementations, so fast, simple algorithms are desirable in order to minimize complexity. The comparative straightforwardness of PSO to many other evolutionary optimization algorithms makes it a good choice for this purpose, and further modifications were applied by the authors of [1] in order to simplify it even further and to introduce concepts from recombinant evolutionary techniques. The resulting algorithm, which can be implemented using only addition and subtraction operators and a simple 1-bit random number generator, is well suited for dedicated hardware settings.

Despite this rather specific original design specification, PSO-DR has shown to be a robust optimizer in its own right, equalling or surpassing a more common PSO implementation on a few tested benchmarks [1]. In this paper we extend the original work of Peña et al. by considering alternative topologies and parameter settings, running comparisons over a more comprehensive test suite, deriving simplified variants of the algorithm, and subjecting the model to a burst analysis.

The following section introduces PSO-DR (known here as model 1) as originally defined by Peña et al. and summarizes the burst analysis of [2]. Section 3 describes a series of simplifications to PSO-DR (models 2 and 3) which are introduced in this paper. The motivations for these simplifications are explained. Section 4 presents the results of performance experiments of models 1–3, and for comparative purposes, standard PSO. Following this, the paper proceeds with an empirical investigation of bursting patterns in recombinant PSO. The final section together draws together the experimental results of this paper and advances some ideas for the immediate future of PSO research.

2. PSO WITH DISCRETE RECOMBINATION

The velocity update for particle i in standard PSO (SPSO) in the inertia weight formalism is

$$\mathrm{IW}: v_{id}^{t+1} = wv_{id}^{t} + \frac{\phi}{2}u_1\left(p_{id} - x_{id}^{t}\right) + \frac{\phi}{2}u_2\left(p_{nd} - x_{id}^{t}\right), \quad (1)$$

where d labels components of the position and velocity vectors, $d = 1, 2, \ldots, D$, $\vec{p}_i$ is the personal best position achieved by i, $\vec{p}_n$ is the best position of informers in i's social neighborhood and $u_{1,2} \sim U(0, 1)$ [3]. After velocity update, the particle position is adjusted:

$$x_{id}^{t+1} = v_{id}^{t+1} + x_{id}^{t}. \tag{2}$$

Peña et al. introduced a recombinant version of PSO by replacing either the personal best or the neighborhood best position by the recombinant position [1]. We focus here on the former for reasons of improved performance and the more interesting social aspect. A recombinant position vector $\vec{r}$ is defined by

$$r_{id} = \eta_d p_{ld} + (1 - \eta_d) p_{rd}, \tag{3}$$

where $\eta_d = U\{0, 1\}$ and $\vec{p}_{l,r}$ are immediate left and right neighbors of i in a ring topology. While separate random numbers η_d are used for separate dimensions d, a single value is generated for each single dimension and used for both occurrences of η_d in that dimension. This places $\vec{r_i}$ at a corner of the smallest D-dimensional box which has p_l and p_r at its corners.

The authors of [1], in a search for a very efficient implementation, argued for the removal of the random numbers $u_{1,2}$ from (1) and parameter settings $\phi = 2$ and $w = 0.5$. The velocity update for the original form of PSO-DR is

$$\text{DR}: v_{id}^{t+1} = w v_{id}^{t} + \frac{\phi}{2}(r_{id} - x_{id}^{t}) + \frac{\phi}{2}(p_{nd} - x_{id}^{t}). \tag{4}$$

The choice of ϕ was based on the observation that $\phi \approx 4.0$ in standard PSO, but, since $u_{1,2}$ are uniform in $[0, 1]$, the expectation value of $\phi u_{1,2}$ is 2.0. Furthermore, the multiplication by $w = 0.5$ can be implemented in hardware by a right shift operation. While optimal efficiency is desirable for hardware implementations, this issue does not concern us to the same degree in this study of (4) and it is one aim of this paper to study PSO-DR for arbitrary parameter values.

Although (4) contains a random element in the recombinant position, the acceleration parameters are constant. In other words, the update rule has additive rather than multiplicative stochasticity [2]. This has two ramifications; first, a stability condition can be computed based on the theory of second order, fixed parameter, difference equations and second, recombinant PSO is predicted not to exhibit particle velocity bursts. The details of these results are to be found in [2]. The stability condition is

$$|w| < 1, \quad 0 < \phi < 2(1 + w). \tag{5}$$

It is known that PSO at stagnation, that is, when no improvements to personal bests are occurring, and the particles effectively decouple, exhibits bursts of outliers [4]. These are temporary excursions of the particle to large distances from the attractors. A burst will typically grow to a maximum and then return through a number of damped oscillations to the region of the attractors. The origin of bursts, and of the concomitant fattening of the tails of the position distribution at stagnation, can be traced to the second-order stochastic difference equation

$$x(t + 1) + a(t)x(t) + bx(t - 1) = c(t) \tag{6}$$

which is equivalent to SPSO with the identification $a(t) = (\phi/2)(u_1 + u_2) - w - 1$, $b = w$, and $c(t) = (\phi/2)(u_1 p_1 + u_2 p_2)$ for fixed attractors $p_{1,2}$. Since $\max(|a|) > 0$, amplification of $x(t)$ can occur through repeated multiplication of $x(t)$ by a despite the second order reduction by multiplication by the constant b. Interestingly, the distribution tail of $|x|$, by virtue of the bursts that become increasingly less probable for increasing size, is fattened compared to an exponential falloff as provided by, for example, a Gaussian. A theoretical justification of these power laws and some empirical tests can be found in [2].

PSO bursts differ from the random outliers generated by PSO models which replace velocity by sampling from a distribution with fat tails such as a Richer and Blackwell [5]. In contradistinction to the outliers of these "bare bones" formulations [6], the outliers from bursts occur in sequence, and they are one dimensional. Bursting will therefore produce periods of rectilinear motion where the particle will have a large velocity parallel to a coordinate axis. Furthermore, large bursts may take the particle outside the search space. Although this will not incur any penalty in lost function evaluations if particles that exit the feasible bounds of the problem are not evaluated, as is the common approach to this situation, they are not contributing to the search while in outer space. PSO-DR, which is predicted not to have bursts [2], therefore provides a salient comparison.

3. SIMPLIFYING RECOMBINANT PSO

This section details the two new recombinant models that are being proposed in this paper. To begin, an investigation into PSO-DR reveals more interesting properties of the formulation. Performance plots for a sweep through parameter space to find an optimal balance between the inertia weight coefficient w and the ϕ coefficients show that while the optimal region is spread across the parameter space, it also intersects the axis for the w term (see Figure 1 for results on selected functions from Table 1). This demonstrates that the system is able to obtain good optimal results even at $w = 0.0$ and there is no inertia term in the velocity update equations.

Model 2 PSO-DR sets $w = 0$, with a velocity update,

$$\text{DR2}: v_{id}^{t+1} = \frac{\phi}{2}(r_{id} - x_{id}^{t}) + \frac{\phi}{2}(p_{nd} - x_{id}^{t}). \tag{7}$$

Velocity now serves as a dummy variable in the update equations (1) and (2) and model 2 can be represented as a single, velocity-free rule

$$\text{DR2}: x_{id}^{t+1} = x_{id}^{t} + \frac{\phi}{2}(r_{id} - x_{id}^{t}) + \frac{\phi}{2}(p_{nd} - x_{id}^{t}). \tag{8}$$

At this point, the two ϕ terms were detached and another sweep through parameter space to find an optimal combination of the recombinant component via its coefficient ϕ_1 and the neighborhood best component via its coefficient ϕ_2 was performed. Surprisingly, results again showed that the optimal region intersects an axis, this time for the neighborhood term $(p_{gd} - x_{id}^{t})$ (see Figure 2 for selected results).

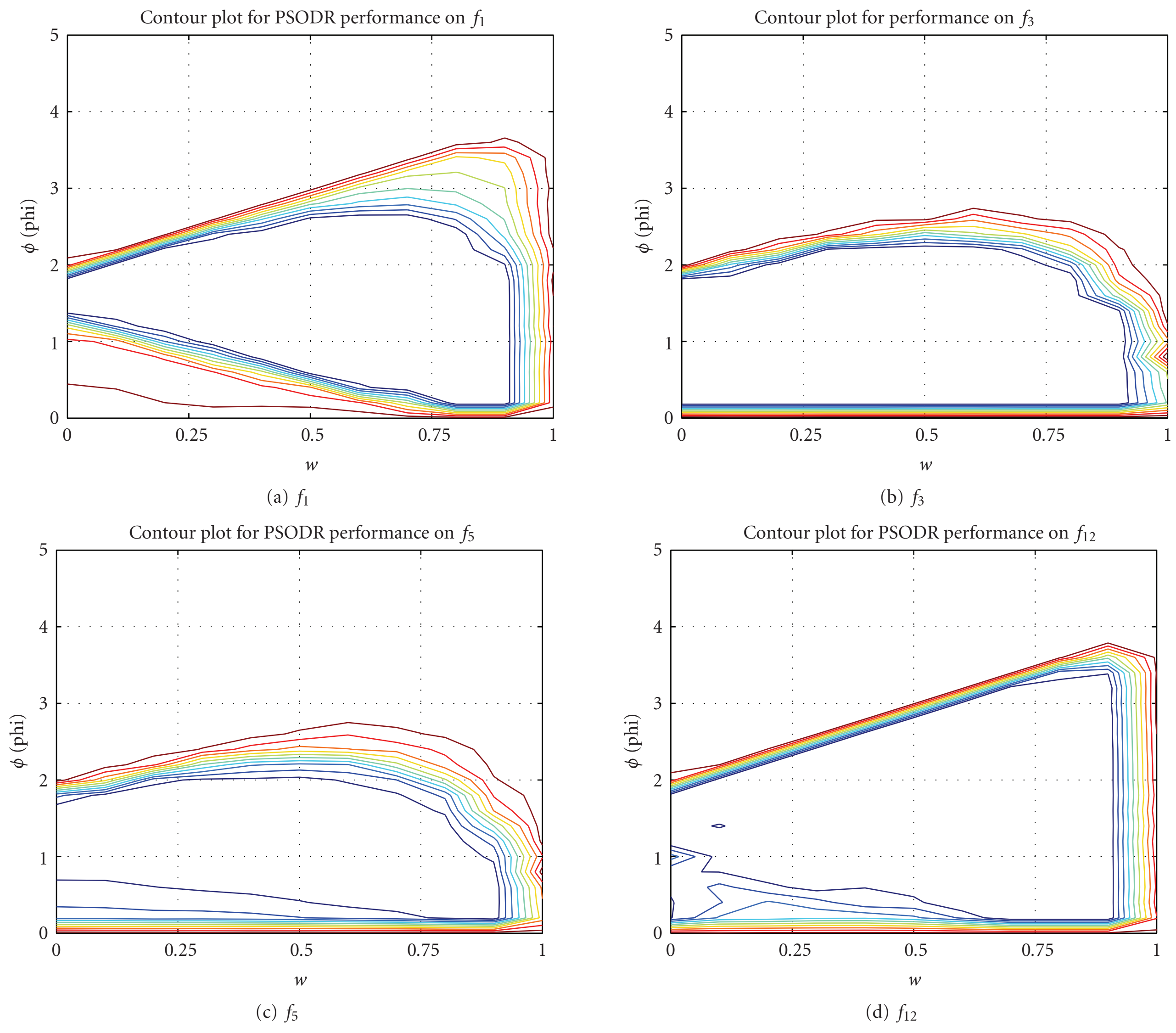


FIGURE 1: Optimal regions for w versus ϕ in PSO-DR model 1, found empirically through 30 runs for each combination of parameters $w = 0.0, \ldots, 1.0, \phi = 0.0, \ldots, 5.0$ at a granularity of 0.1. Each contour line represents a 10% improvement in performance with the region within the innermost line representing the best performing 10% of possible combinations of w and ϕ.

This allows a further simplification to the update equation (4), down to PSO-DR model 3:

$$\text{DR3} : x_{id}^{t+1} = x_{id}^{t} + \phi(r_{id} - x_{id}^{t}) \tag{9}$$

which is clearly a substantial reduction of the original PSO-DR equation.This PSO variant, if it proves to be viable, would raise a couple of interesting questions.To what extent is velocity a necessary component, or is it a relic of the biological origins of PSO [6]? Secondly, how important is the neighborhood component drawn from the single best neighbor? The optimization process of Model 3 is entirely driven by the recombinant component; this idea is reminiscent of fully informed particle swarms (FIPS) [7], where the entire neighborhood influences particle behavior. However, whereas FIPS allows every neighbor to influence a particle's behavior in every dimension, Model 3 allows only a single randomly chosen neighbor to fully influence the particle in each dimension. This gives the particle an updated position that is a combination of the best positions of all of its neighbors throughout all dimensions.

The following section presents evidence that PSO-DR3 is a viable alternative to standard PSO by reporting on performance results for all three models of PSO-DR over a number of commonly used test functions.

4. PERFORMANCE EXPERIMENTS

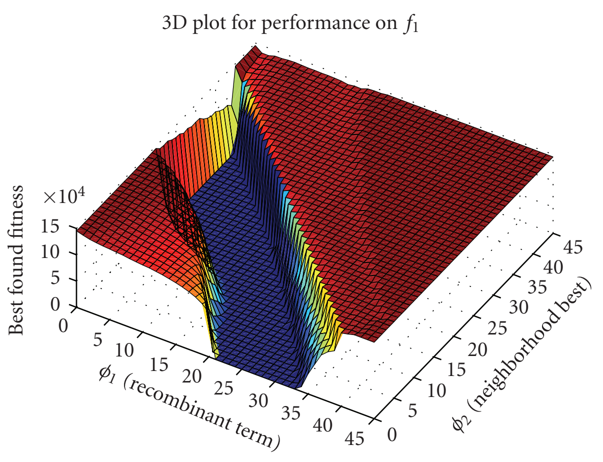
Algorithms were tested over a series of 14 benchmark functions chosen for their variety, shown in Tables 1 and 2. Functions $f_1 - f_3$ are unimodal functions with a single minimum, $f_4 - f_9$ are complex high-dimensional multimodal problems, each containing many local minima and a single

Table 1: Benchmark function equations.

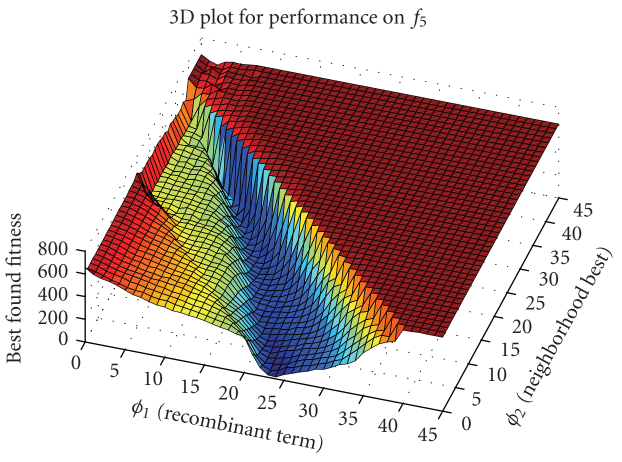
Equation
$f_1 = \sum_{i=1}^{D} x_i^2$
$f_2 = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$
$f_3 = \sum_{i=1}^{D-1}\{100(x_{i+1}-x_i^2)^2 + (x_i-1)^2\}$
$f_4 = -\sum_{i=1}^{D} x_i \sin(\sqrt{x_i})$
$f_5 = \sum_{i=1}^{D}\{x_i^2 - 10\cos(2\pi x_i) + 10\}$
$f_6 = -20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right\} - \exp\left\{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right\} + 20 + e$
$f_7 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
$f_8 = \frac{\pi}{D}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{D-1}(y_i-1)^2\{1+10\sin^2(\pi y_{i+1})\} + (y_D-1)^2\right\} + \sum_{i=1}^{D}\mu(x_i,10,100,4)$
$y_i = 1 + \frac{1}{4}(x_i+1)$
$\mu(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i-a)^m & x_i < -a \end{cases}$
$f_9 = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{D-1}(x_i-1)^2\{1+\sin^2(3\pi x_{i+1})\} + (x_D-1)^2 \times \{1+\sin^2(2\pi x_D)\}\right\} + \sum_{i=1}^{D}\mu(x_i,5,100,4)$
$f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
$f_{11} = \{1 + (x_1+x_2+1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\} \times \{30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}$
$f_{12} = -\sum_{i=1}^{5}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$
$f_{13} = -\sum_{i=1}^{7}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$
$f_{14} = -\sum_{i=1}^{10}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$

global optimum, and $f_{10} - f_{14}$ are lower-dimensional multimodal problems with few local minima and a single global optimum apart from f_{10}, which is symmetric about the origin with two global optima.

Particles were initialized using the *region scaling* technique where initialization takes place in an area of the search space known not to contain the global optimum [8]. To avoid initializing the entire swarm directly within a local minimum, as could be possible with $f_{12} - f_{14}$ if initialization takes place in the bottom quarter of the search space in each dimension (as is common), an area of initialization composed of the randomly chosen top or bottom quarter of each dimension was defined, into which all particles were placed with uniform distribution. This method ensures that the swarm will not be initialized within the same area for every optimization run, but will still be confined to an area at most 0.25^D of the search space, making the chance of initialization directly on or near the global optimum extremely unlikely. In instances where the global optimum was located at the center of the search space (i.e., $f_1, f_2, f_5 - f_7$), the function was *shifted* by a random vector with maximum magnitude of a tenth of the size of the search space in each dimension for each run to remove any chance of a centrist bias [9].



(a) f_1



(b) f_5

Figure 2: Performance plots for ϕ_1 and ϕ_2 in PSO-DR model 2, found empirically through 30 runs for each combination of parameters $\phi_1 = 0.0, \ldots, 4.5, \phi_2 = 0.0, \ldots, 4.5$ at granularity 0.1.

Table 2: Benchmark function details.

Function	Name	D	Feasible bounds
f_1	Sphere/parabola	30	$(-100, 100)^D$
f_2	Schwefel 1.2	30	$(-100, 100)^D$
f_3	Generalized Rosenbrock	30	$(-30, 30)^D$
f_4	Generalized Schwefel 2.6	30	$(-500, 500)^D$
f_5	Generalized Rastrigin	30	$(-5.12, 5.12)^D$
f_6	Ackley	30	$(-32, 32)^D$
f_7	Generalized Griewank	30	$(-600, 600)^D$
f_8	Penalized function P8	30	$(-50, 50)^D$
f_9	Penalized function P16	30	$(-50, 50)^D$
f_{10}	Six-hump camel-back	2	$(-5, 5)^D$
f_{11}	Goldstein-price	2	$(-2, 2)^D$
f_{12}	Shekel 5	4	$(0, 10)^D$
f_{13}	Shekel 7	4	$(0, 10)^D$
f_{14}	Shekel 10	4	$(0, 10)^D$

This investigation tested PSO-DR model 1 using both global (as used in the originally proposed algorithm) and local ring topologies for selecting the neighborhood operator p_n. The parameter settings were Pena's, giving a velocity update with the form

$$v_{id}^{t+1} = 0.5v_{id}^t + (r_{id} - x_{id}^t) + (p_{nd} - x_{id}^t). \qquad (10)$$

Results shown for PSO-DR model 2 use the value $\phi \approx 1.6$, while those for PSO-DR model 3 use $\phi \approx 1.2$. These values were empirically determined to be optimal for these algorithms; an analytical determination is the subject of current research. Results for both models 2 and 3 are shown for runs using a ring topology, which showed superior performance in testing.

For comparison, results are presented for a standard PSO algorithm (SPSO), which operates using the constricted velocity update equation

$$v_{t+1} = \chi\left(v_t + \frac{\phi}{2}u_1(p_i - x_t) + \frac{\phi}{2}u_2(p_g - x_t)\right) \qquad (11)$$

with $\phi = 4.1, \chi = 0.72984$ and with 50 particles [3]. All PSO-DR model tests were carried out using 50 particles as well. Algorithm performance was measured as the minimum error $|f(x) - f(x^\star)|$ found over the trial where $f(x^\star)$ is the fitness at the global optimum for the problem. Results were averaged over 30 independent trials, and are displayed, with standard error, in Table 3. Values less than 10^{-15} have been rounded to 0.0.

Performance results in Table 3 for all models of PSO-DR versus SPSO clearly indicate that it is a competitive variant, especially on highly complex problems such as f_5 (Rastrigin). Statistical tests were performed on these results to determine the significance of the performance differences between the

Table 3: Mean error after 30 trials of 300 000 evaluations. Necessary function evaluations are shown where 0.0 error was attained.

	SPSO Ring	SPSO Global	PSO-DR M1 Ring	PSO-DR M1 Global	PSO-DR M2	PSO-DR M3
f_1	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
	97063 ± 377	46897 ± 421	59322 ± 125	33290 ± 170	60063 ± 41	75810 ± 322
f_2	0.12 ± 0.01	**0.0 ± 0.0**	0.01 ± 0.002	**0.0 ± 0.0**	3.7E-7 ± 7.5E-8	5.14 ± 1.27
	—	297800 ± 928	—	168852 ± 1205	—	—
f_3	6.18 ± 1.07	8.37 ± 2.26	16.79 ± 0.49	0.80 ± 0.29	34.57 ± 5.46	18.64 ± 4.45
	—	—	—	—	—	—
f_4	3385 ± 40	3522 ± 32	2697 ± 36	3754 ± 48	2418 ± 27	1830 ± 46
	—	—	—	—	—	—
f_5	163.50 ± 5.64	140.16 ± 5.87	44.64 ± 2.71	115.51 ± 7.03	35.21 ± 2.13	9.88 ± 0.86
	—	—	—	—	—	—
f_6	18.28 ± 0.85	12.93 ± 1.59	0.68 ± 0.67	18.51 ± 0.90	**0.0 ± 0.0**	**0.0 ± 0.0**
	—	—	—	—	287220 ± 2105	248160 ± 1945
f_7	**0.0 ± 0.0**	0.019 ± 0.004	**0.0 ± 0.0**	0.008 ± 0.002	**0.0 ± 0.0**	**0.0 ± 0.0**
	110616 ± 3320	—	101526 ± 9227	—	81226 ± 6560	70348 ± 2954
f_8	0.004 ± 0.003	0.15 ± 0.05	**0.0 ± 0.0**	0.05 ± 0.02	**0.0 ± 0.0**	**0.0 ± 0.0**
	—	—	61370 ± 249	—	85101 ± 581	95810 ± 655
f_9	**0.0 ± 0.0**	0.003 ± 0.001	**0.0 ± 0.0**	0.002 ± 0.0007	**0.0 ± 0.0**	**0.0 ± 0.0**
	106163 ± 537	—	61793 ± 221	—	86031 ± 377	92416 ± 437
f_{10}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
	9348 ± 190	11808 ± 445	44577 ± 7608	40015 ± 4483	6103 ± 104	5918 ± 75
f_{11}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
	8258 ± 104	7080 ± 108	4772 ± 47	3968 ± 27	6720 ± 66	6846 ± 87
f_{12}	0.59 ± 0.33	4.61 ± 0.54	0.17 ± 0.17	4.34 ± 0.59	**0.0 ± 0.0**	**0.0 ± 0.0**
	—	—	—	—	59958 ± 43	16229 ± 855
f_{13}	1.09 ± 0.45	4.40 ± 0.60	**0.0 ± 0.0**	2.55 ± 0.62	8.1E-11 ± 1.8E-11	**0.0 ± 0.0**
	—	—	13433 ± 249	—	—	14012 ± 764
f_{14}	0.96 ± 0.45	3.24 ± 0.66	**0.0 ± 0.0**	3.13 ± 0.66	6.6E-11 ± 1.7E-11	**0.0 ± 0.0**
	—	—	12760 ± 1386	—	—	121031004 ±

two algorithms. To avoid the problem of the probabilistic nature of t-tests potentially affecting results when conducting multiple significance tests, a modified Bonferroni procedure was applied to values of α for successive tests [10]. This procedure involves inversely ranking observations by ascending values of p, then setting

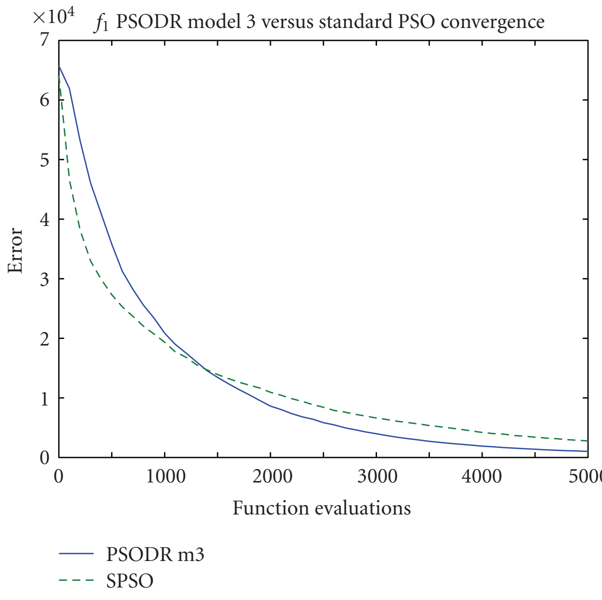
$$\alpha' = \frac{\alpha}{\text{inverse rank}}. \tag{12}$$

Results for these statistical tests on PSO-DR model 3 and SPSO are shown in Table 4 and confirm that the performance is significantly improved on 3 of the 14 tested functions, equivalent for 10 functions, and worsened for 1 function for PSO-DR model 3 versus SPSO with ring topology. Perhaps the most impressive improvement comes for f_5 (Rastrigin), a notoriously difficult multimodal problem that PSO algorithms perform poorly on some problems in high dimensionality.

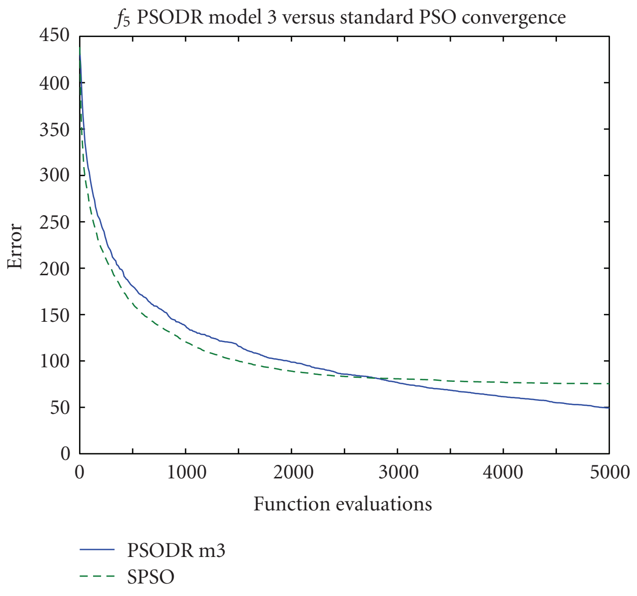
Due to the high number of function evaluations that were performed to obtain these results relative to previous work (where only 30 k–60 k function evaluations might be performed), selected convergence plots are shown in Figure 3. These show that the standard PSO obtains superior results at the very start of the optimization process, up to 5000 function evaluations for the highest observed value (Figure 3(b)). After the point at which this occurs, PSO-DR model 3 surpasses the standard algorithm in performance, and maintains this advantage to the end of the 300 k function evaluations on 7 of the 14 tested problems ($f_4 - f_6, f_8, f_{12} - f_{14}$). On problems for which both algorithms attained equal error levels of 0.0 ($f_1, f_7, f_9 - f_{11}$), the point at which this occurs, that is, when SPSO "catches up" to PSO-DR model 3, can be observed in Table 3. On average, SPSO took 25% more function evaluations to attain the optimum than PSO-DR model 3 on these problems. Finally, for the two problems on which SPSO outperformed PSO-DR model 3 (f_2, f_3), the same early performance is seen with PSO-DR model 3 surpassing SPSO in performance early in the optimization process; in these cases, SPSO eventually repasses the other algorithm by 50 k function evaluations.

Table 4: Significance for SPSO versus PSO-DR model 3 with ring topologies.

Function	p-value	Inverse rank	α'	Significant
f_4	0	14	0.003 571	Yes
f_5	0	13	0.003 846	Yes
f_6	0	12	0.004 167	Yes
f_2	2.11e-11	10	0.005	Yes
f_3	0.0086	11	0.004 545	No
f_{13}	0.02	9	0.005 556	No
f_{14}	0.04	8	0.00 625	No
f_{12}	0.2663	6	0.08	No
f_8	0.3215	7	0.007 143	No
f_1	1	5	0.01	No
f_7	1	4	0.0125	No
f_9	1	3	0.016 667	No
f_{10}	1	2	0.025	No
f_{11}	1	1	0.05	No



(a) f_1

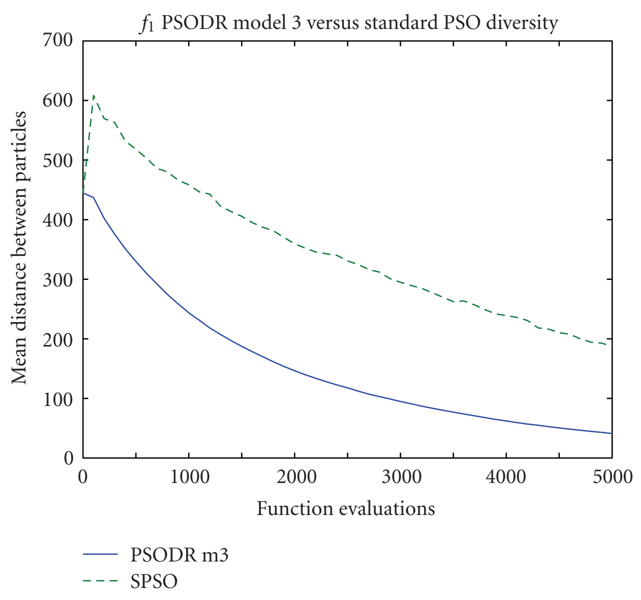


(b) f_5

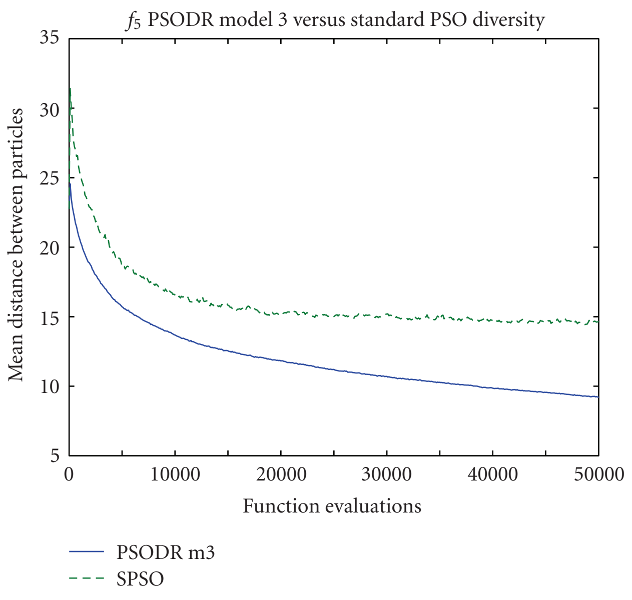
Figure 3: Convergence plots for SPSO and PSO-DR model 3 early in the optimization process.

A potential explanation for this behavior lies in the diversity of the swarms at this point in the optimization process. Figure 4 shows the mean Euclidean distance between particles for the corresponding convergence plots of Figure 3. It should be noted that uniform initialization was used in the trials used to generate these plots; relative performance between the algorithms was unaffected, and initializing particle positions uniformly throughout the search space removes an unrelated phenomenon in subspace initialization wherein the swarm expands greatly beyond the relatively small initialization region at the start of the optimization process to explore the search space. Expansion is common in the first few iterations using uniform initialization as well, but this is inherent to the swarm behavior and influenced only by the size of the entire search space.

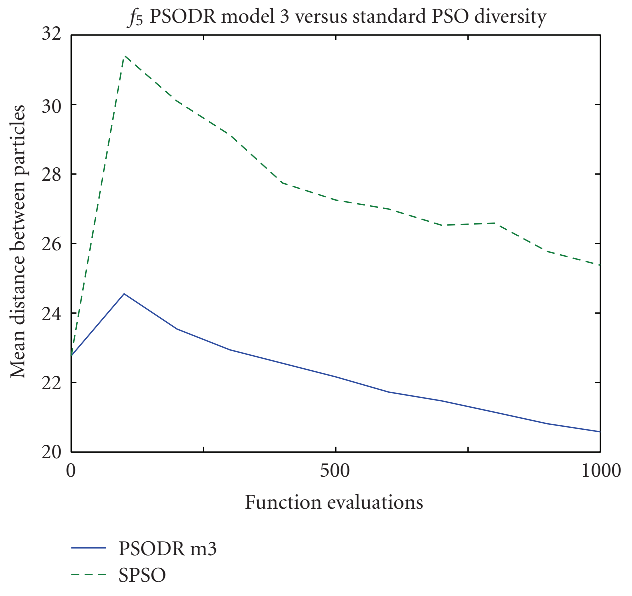
As can be seen in the plots of Figure 4, neither swarm type begins converging immediately following initialization but rather they maintain their diversity or expand slightly. On a comparative basis, the standard PSO swarm expands substantially more than the PSO-DR model 3 swarm; for example Figure 4(c) shows that after the first 100 function evaluations, the mean distance between particles in the standard PSO swarm increases from 23 to 31.5, while the PSO-DR swarm diversity increases only from 23 to 24.5. Similar disparities were observed for all other tested problems.



(a) f_1



(b) f_5



(c) f_5*(zoomed)*

Figure 4: Diversity plots for SPSO and PSO-DR model 3 early in the optimization process.

It is reasonable to gather from these results that the higher swarm diversity for the standard PSO algorithm early in the optimization process demonstrates a wider spread of particle dispersion, and hence an improved probability of finding and starting to explore the basin of attraction for global or good local optima. PSO-DR model 3 expands very little, if at all, early in the optimization process, resulting in delayed acquisition of optimal regions of the search space.

5. EXAMINATION OF BURSTING

Bursts in the velocities of particles are commonly observed using the standard PSO algorithm. These are generated by means of the multiplicative stochasticity of the algorithm [2]. In order to investigate bursting behavior in PSO-DR and SPSO an empirical measure was devised.

This bursting measure was implemented to highlight when a particle had a velocity in a single dimension that was considerably higher than the next highest dimensional velocity. Bursting patterns of behavior were detected by reporting that every time particle velocity in a single dimension was a set amount λ times higher than velocity in the next highest dimension. Bursting behavior is demonstrated in Figure 6, where the velocity of a single particle in a 10-dimensional problem is shown. On the plot of the multidimensional velocity of the SPSO particle, it can be seen that velocity in a

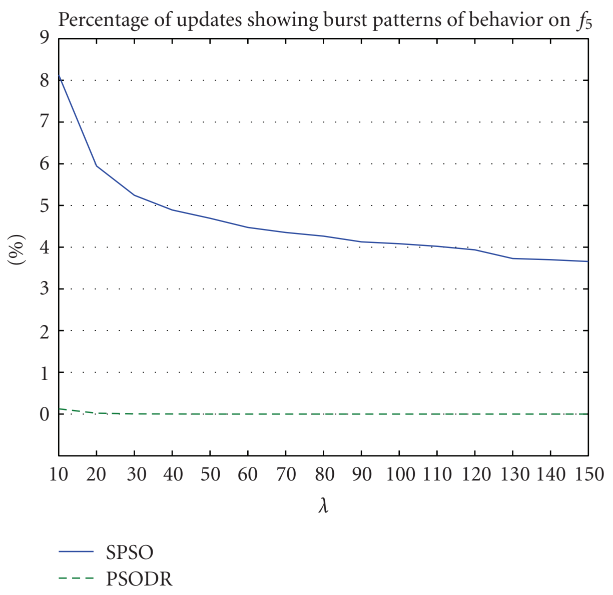
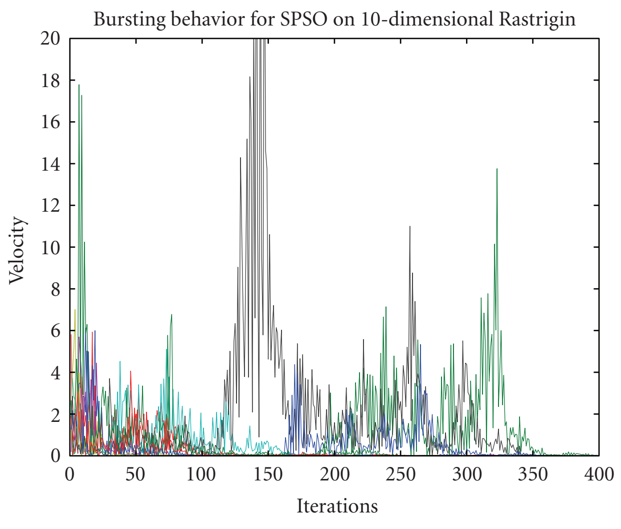
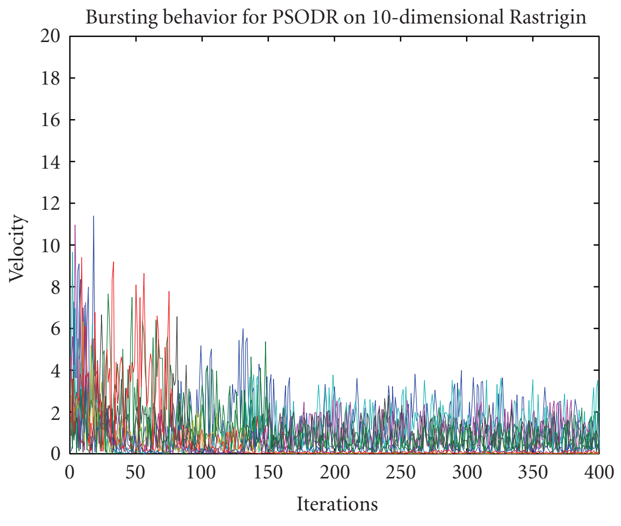


Figure 5: Frequency of updates showing burst behavior for values of λ.



(a) SPSO



(b) PSO-DR

Figure 6: Representative particle velocities for SPSO and PSO-DR on 10D Rastrigin.

single dimension increases suddenly and dramatically while remaining relatively level and low in all other dimensions. This is an example of a velocity burst. While the figure shows velocity for a single particle on a single run, examination of velocity plots for hundreds of particles over dozens of runs confirmed this to be representative of general particle behavior.

Velocity for a PSO-DR particle is also shown in Figure 6, and demonstrates the absence of bursts. Similarly to the SPSO plot, examination of a large number of plots confirmed this to be representative of general behavior for PSO-DR.

Examination of these empirical analyses show that PSO-DR clearly does not contain bursting behavior on the scale of SPSO while demonstrating equal or superior performance on 13 of the 14 benchmark functions, leading to the hypothesis that bursts are not, in fact, integral to the successful operation of particle swarm algorithms. The fact that a very few bursts do occur with PSO-DR indicates that it is a highly improbable feature of DR dynamics.

Analysis performed on statistics of several functions shows that particle updates involving bursts are far less effective than more common nonbursting updates. For example, results showed that for SPSO on f_5 with $\lambda = 100$, on average 20.1% of *all* particle, updates involve an improvement to the particle's best found position p_i, whereas only 1.8% of updates involving bursts result in an improvement to p_i. Likewise, on average 0.9% of all particle, updates improve the best found swarm position g, as opposed to only 0.01% for bursting particles. Burst frequencies for values of λ from 10 to 150 are shown in Figure 5.

It is also interesting to note that far fewer *total* updates result in an improved p_i or g for PSO-DR when compared to SPSO, for example, results showed that 20.1% of all updates improve p_i for SPSO compared with 0.64% for PSO-DR, and 0.91% improve g for SPSO compared with 0.02% for PSO-DR on f_5 for $\lambda = 100$.

6. CONCLUSIONS

Simplification of the standard PSO algorithm is an important step toward understanding how and why it is an effective optimizer. By removing components of the algorithm and seeing how this affects performance, we are granted insight into what those components contribute to overall particle and swarm behaviors.

In particular, this paper has proposed a very simple PSO

$$\text{DR3}: x_{id}^{t+1} = x_{id}^{t} + \phi\left(r_{id} - x_{id}^{t}\right) \tag{13}$$

which offers competitive performance to standard PSO, but removes multiplicative randomness, inertia, and the personal memory term p_i from the position update.

There is still much to be done before questions concerning PSO behavior can be completely answered, and it is expected that the next decade of PSO research will be focused on understanding the basic algorithm that powers both the standard implementation and its variants.

In that light, the PSO-DR variant is important not only because of its improved performance on several benchmark functions, but also because its simplified state allows us to examine what happens to the standard algorithm when pieces are modified or removed. Based on the results presented here, it can be argued that large bursts are not generally beneficial or integral to PSO performance, and may possibly be detrimental. Although the presence of particle outliers is demonstrably important for swarm optimization (as demonstrated in bare bones analysis, [6]), bursts, which are sequences of extreme particle positions, occurring along an axis and reaching outside the search space, remain a special feature of velocity-based swarms. This work, which compares standard PSO to a burst-free but comparable optimizer suggests that bursts are disadvantageous in general. (However, in the coincidence that the objective function has a rectangular symmetry aligned with the axes, then bursting may actually be fortuitous.)

Further, the replacement of the direct personal influence operator p_i from SPSO with the recombinant term r_i derived from its neighborhood in PSO-DR strengthens the case for PSO being mostly reliant on social interaction as opposed to personal experience. This is further supported by the effectiveness of PSO-DR model 3, which lacks a cognitive term altogether. The social behavior occurring inside of a swarm is still a wide-open area in the field, and will hopefully constitute a great deal of the future research devoted to the development of a better understanding of this deceptively simple optimizer.

Another property of PSO-DR resides in attractor jiggling that takes place even at stagnation (no updates to any p_i) since r_i is never fixed. This jiggling will work against convergence and could propel the swarm onwards. This, and other matters concerning the nature of recombination within PSO, will be the subject of further study.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of EPSRC XPS Project (GR/T11234/01). The authors also wish to thank Jim Kennedy for advice and references.

REFERENCES

[1] J. Peña, A. Upegui, and E. Sanchez, "Particle swarm optimization with discrete recombination: an online optimizer for evolvable hardware," in *Proceedings of the 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS '06)*, pp. 163–170, Istanbul, Turkey, June 2006.

[2] T. Blackwell and D. Bratton, "Origin of bursts," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pp. 2613–2620, London, UK, July 2007.

[3] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, Honolulu, Hawaii, USA, April 2007.

[4] J. Kennedy, "Probability and dynamics in the particle swarm," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 1, pp. 340–347, IEEE Press, Portland, Ore, USA, June 2004.

[5] T. J. Richer and T. Blackwell, "The Lévy particle swarm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 808–815, IEEE Press, Vancouver, BC, Canada, July 2006.

[6] J. Kennedy, "Bare bones particle swarms," in *Proceedings of IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 80–87, Indianapolis, Ind, USA, April 2003.

[7] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[8] D. K. Gehlhaar and D. B. Fogel, "Tuning evolutionary programming for conformationally flexible molecular docking," in *Proceedings of the 5th Annual Conference on Evolutionary Programming (EP '96)*, L. Fogel, P. Angeline, and T. Back, Eds., pp. 419–429, MIT Press, San Diego, Calif, USA, February 1996.

[9] C. K. Monson and K. D. Seppi, "Exposing origin-seeking bias in PSO," in *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO '05)*, pp. 241–248, Washington, DC, USA, June 2005.

[10] J. Jaccard and C. K. Wan, *LISREL Approaches to Interaction Effects in Multiple Regression*, vol. 114 of *Quantitative Applications in the Social Sciences*, Sage Publications, Thousand Oaks, Calif, USA, 1996.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 289564, 12 pages
doi:10.1155/2008/289564

Research Article

What Else Is the Evolution of PSO Telling Us?

Laura Dioşan and Mihai Oltean

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania

Correspondence should be addressed to Laura Dioşan, lauras@cs.ubbcluj.ro

Received 20 July 2007; Revised 29 November 2007; Accepted 29 November 2007

Recommended by Alex Freitas

Evolutionary algorithms (EAs) can be used in order to design particle swarm optimization (PSO) algorithms that work, in some cases, considerably better than the human-designed ones. By analyzing the evolutionary process of designing PSO algorithms, we can identify different swarm phenomena (such as patterns or rules) that can give us deep insights about the swarm behavior. The rules that have been observed can help us design better PSO algorithms for optimization. We investigate and analyze swarm phenomena by looking into the process of evolving PSO algorithms. Several test problems have been analyzed in the experiments and interesting facts can be inferred from the strategy evolution process (the particle quality could influence the update order, some particles are updated more frequently than others, the initial swarm size is not always optimal).

1. INTRODUCTION

Various evolutionary and nonevolutionary methods have been proposed in order to solve complex search and optimization problems. Among these methods, a special place is occupied by evolutionary algorithms (EAs) [1, 2] and by the techniques based on swarm intelligence (such as particle swarm optimization (PSO) [3] and ant colony optimization [4]). The main advantage of these methods is given by the possibility to use them for searching in various spaces without performing big changes in the structure of the algorithm. They can be easily adapted (by the human being or by themselves) to the peculiarities of the problem which is being solved.

PSO is a population-based stochastic optimization technique proposed by Kennedy and Eberhart [5–7]. The standard PSO algorithm randomly initializes a group of particles (solutions) and then searches for optima by updating all the particles along a number of iterations. In any iteration, each particle is updated by following a few simple rules [8, 9].

The standard model implies that particles are updated synchronously [6]. This means that the current position and speed of a particle are computed by taking into account only the information from the previous iteration of particles. The model investigated in this paper is a more general one and it was proposed in [10]. In this paper, we are taking further steps into our research by exploring the phenomena that arise inside a swarm during the evolutionary design.

Our analysis focuses on an asynchronous version of the PSO algorithm. This variant has the following characteristics.

(i) When a particle is updated, the current state of the swarm (the position and the velocity of all the particles) is taken into account. The best global and local values are computed for each particle which is about to be updated, because the previous modifications could affect these two values. This is different from the standard PSO algorithm (or synchronous PSO algorithm [6]) where the particles were updated taking into account only the information from the previous iteration (the modifications performed so far by a standard PSO in the current iteration had no influence over the modifications performed further in the current iteration) and it is closer to the asynchronous PSO algorithm [11, 12] that updates particle positions and velocities continuously, based on currently available information.

(ii) In our model, the particles are updated based on their quality. This fitness-based update is important because it could be better to firstly modify the best particles of the swarm and secondly the worst particles (or vice versa). This is again different from the standard asynchronous PSO algorithm [11, 12], which updates the particle positions and velocities by always using the same predefined order: first particle, second particle, and so on (there are no relationships between the update order and the quality of the particles).

(iii) Some particles may be updated more often than other particles. For instance, in some cases, it is more important to update the best particles several times per iteration than to update the worst particles.

(iv) During the evolution, the swarm size can be modified for at least two reasons: some particles perform more moves (in order to improve their quality), while other particles are never updated. This is why the weakest particles are eliminated from the swarm. Unlike the standard PSO algorithm, which works with a pre-established swarm size, the current model finds by itself the optimal size of the swarm along the evolution.

We intend to study how we can obtain better PSO algorithms. In order to achieve this goal we employ an evolutionary approach: we start with a population of randomly generated PSO algorithms and we try to improve them along a fixed number of generations. During the evolution, we try to discover new swarm phenomena such as the special relationships between particles, their quality, their update order, and the optimal swarm size or some rules in the update strategy of the swarm during the evolution process. These rules can be repeatedly applied in order to obtain better approximations of the solution. The process of particle update is influenced by these rules.

Because a PSO is a complex algorithm, we cannot evolve all its aspects. We are taking into account only an important one, namely the order in which the particles are updated. Other aspects, such as the equation used for updating a particle, have been analyzed in [13]. The new swarm phenomena investigated in this paper are

(i) the update frequencies, how many times a particle is updated;
(ii) the order of the updates, the order of particles that is taken into account in order to modify their position and velocity;
(iii) the optimal swarm size.

Such information can help us design better PSO algorithms for optimization. These phenomena have been examined for different test problems.

The paper is structured as follows: Section 2 provides a brief review of the work on PSO parameter optimization. Section 3 describes the model for evolving the PSO update strategy. In Section 4.1, an analysis of the optimal swarm size detected during the evolution is presented. In addition, the frequency of the updates performed in the swarm is investigated. Several rules identified in the PSO update strategy are presented in Section 4.3. Section 4.4 summarizes the most important ideas of this analysis and the main features of the developed model. Conclusions and further work directions are suggested in Section 5.

2. RELATED WORK

Many improvements of the basic form of PSO have been proposed and tested in the literature [14–19]. Also, several analyses of the PSO behavior have been performed [13, 20–22]. Much of this work is focused on the convergence of the PSO algorithm.

Ozcan and Mohan [22] have analyzed the trajectory of a particle in the "original" PSO algorithm (without an inertia weight or a constrict coefficient) and van der Bergh has carried out the first PSO convergence study [23]. Later, Clerc and Kennedy [20] have proposed the model based on the constrict coefficient.

Langdon et al. [21] have evolved kernel functions, which describe the average behavior of a swarm of particles as if it were responding as a single point moving on a landscape transformed by the kernel. The evolved functions (obtained with the genetic programming technique) give another landscape, which is "perceived" by a simple hill climber. The goal for the genetic programming is to evolve a kernel, which causes the movement of the hill climber to resemble the movement of the whole PSO swarm.

Several approaches [13, 19, 24–28] have proposed various hybrid evolutionary algorithms that combine the concepts of EA and PSO.

The aim of the model from [19] has been to extend the PSO algorithm so that it could effectively search into multiconstrained solution spaces (whose constraints are imposed by some real-world problems, such as scheduling), due to the constraints rigidly imposed by the PSO equations. In order to overcome these constraints, this algorithm completely replaces the PSO equations with a self-updating mechanism, which emulates the workings of the equations and which allows flexible incorporation of the real-world heuristics into the algorithm.

Kwong and Jacob [25] have proved the way in which EAs can be used to explore complex pattern formations of swarm systems in 3D space. It is noteworthy that these patterns exhibit a high level of self-organization.

The particle evolutionary swarm optimization algorithm [27] has introduced two new perturbation operators: "c-perturbation" and "m-perturbation." The goal of these operators is to fight premature convergence and poor diversity issues observed in PSO implementations.

Hendtlass [24], Parsopoulos and Vrahatis [26], and Zhang and Xie [28] have used the differential evolution algorithm (suggested by Storn and Price [29]) for the "on the fly" adaptation of the PSO parameters. Using genetic programming, Poli et al. [13] have studied the possibility of evolving the optimal force generating equations, which control the particles in a PSO (forces that stimulate each particle to fly towards the best point sampled by it and towards the swarm best and back).

Several attempts at evolving evolutionary algorithms (EAs) using similar techniques have been performed in the past. A nongenerational EA has been evolved [30] by using the multiexpression programming technique [31]. A generational EA has been evolved [32] by using the linear genetic programming technique [33–35]. Numerical experiments have shown [30, 32] that the evolved EAs perform similarly and sometimes even better than the standard evolutionary approaches with which they have been compared.

3. THE MODEL FOR EVOLVING THE UPDATE STRATEGY OF PARTICLES

The main idea of the model proposed in [10] is to evolve arrays of integers, which provide a meaning for updating the individuals within a PSO algorithm during iteration. The model is a hybrid technique that works at two levels: the first (macro) level consists in a steady-state genetic algorithm (GA) whose chromosomes encode the update strategy of PSO algorithms. In order to compute the quality of a GA chromosome, a PSO algorithm is run (its update order being encoded into that chromosome). Thus, the second (micro) level consists in a modified PSO algorithm that computes the quality of a GA chromosome.

3.1. Representation

The standard PSO algorithm works with a group of particles (solutions) and then searches for the optima by updating them during iteration. Each particle is updated following two "best" values. The first one is the location of the best solution that a particle has achieved so far. This value is called p Best. Another "best" value is the location of the best solution that any neighbor of a particle has achieved so far. This best value is a neighbourhood best and called n Best.

In a standard PSO algorithm, all the particles will be updated once during the course of the iteration. In a real world swarm (such as a flock of birds), not all the birds update their position and velocity at the same time. Some of them update these values more often and others update theirs later or never. By tacking into account these frequencies, it is interesting to discover (evolve) a model that can tell us which particles/birds must be updated and which is the optimal order for updating them.

A GA [1] is used (in [10]) for evolving the update strategy of a PSO algorithm. Each GA individual is a fixed-length string of genes and each gene is an integer number, from the $\{0, 1, \dots, \text{Swarm Size} - 1\}$ set. These values represent the indexes of the particles that will be updated during PSO iterations. It is possible that some particles should be updated more often, while others should not be updated at all. Therefore, a GA chromosome must be transformed so that it should contain only the values from 0 to Max, where Max represents the number of different genes within the current array.

Suppose that we want to evolve the update strategy of a PSO algorithm with eight particles. This means that the Swarm Size = 8 and all the chromosomes of the macrolevel algorithm will have eight genes whose values are in the $\{1, 2, \dots, 8\}$ set. A GA individual with eight genes can be

$$C_1 = (3, 1, 5, 2, 8, 6, 7, 4). \tag{1}$$

In order to compute the fitness of this chromosome, a swarm with eight individuals is used and the following updates are performed during iteration:

$$\begin{aligned} &\text{update (Swarm [3])},\\ &\text{update (Swarm [1])},\\ &\text{update (Swarm [5])},\\ &\text{update (Swarm [2])},\\ &\text{update (Swarm [8])},\\ &\text{update (Swarm [6])},\\ &\text{update (Swarm [7])},\\ &\text{update (Swarm [4])}. \end{aligned} \tag{2}$$

In this example, all the eight particles have been updated once per iteration.

Let us consider another example, which consists of a chromosome C_2 with 8 genes that contain only 5 different values

$$C_2 = (6, 2, 1, 4, 7, 1, 6, 2). \tag{3}$$

In this case, particles 1, 2, and 6 are updated two times each, while the particles 3, 5, and 8 are not updated at all. Because of this, it is necessary to remove the useless particles and to scale the genes of the GA chromosome to the set $\{1, 2, \dots, 5\}$. The obtained chromosome is

$$C_2' = (4, 2, 1, 3, 5, 1, 4, 2). \tag{4}$$

The quality of this chromosome will be computed by using a swarm of size five (five swarm particles), performing the following eight updates:

$$\begin{aligned} &\text{update (Swarm [4])},\\ &\text{update (Swarm [2])},\\ &\text{update (Swarm [1])},\\ &\text{update (Swarm [3])},\\ &\text{update (Swarm [5])},\\ &\text{update (Swarm [1])},\\ &\text{update (Swarm [4])},\\ &\text{update (Swarm [2])}. \end{aligned} \tag{5}$$

Performing this transformation, we can obtain another swarm which has a new size. The model described evolves only the update strategy for a PSO algorithm, but it can also find the optimal swarm size in the same time, even if this parameter is not directly evolved.

We evolve an array of indexes based on the information taken from the function to be optimized. In other words, we evolve an array that contains the update order for the PSO algorithm. This algorithm is used in order to find the optimal value(s) of a function. The quality of the update strategy is given by the performance of the PSO algorithm.

Note that the mentioned mechanism should not be only based on the index of the particles in the Swarm array. This means that it would not be interested in updating a particular position, since the same position can contain a very good individual in one run and a very poor individual in another.

For instance, it is easy to see that all the GA chromosomes, encoding permutations, perform similarly when averaged.

In order to avoid this problem, the Swarm array is sorted in an ascending way (after each iteration), based on the fitness value. The first position will always hold the best particle at the beginning of the iteration. The last particle in this array will always hold the worst particle found at the beginning of the iteration. In this way, it is known that update (Swarm [1]) will mean that the respective particle is not updated, but the best particle at the beginning of the current iteration will be.

3.2. Fitness assignment

The model for evolving the PSO update strategy is structured on two levels: a macrolevel and a microlevel. The macro-level is a GA that evolves the update strategy of a PSO algorithm. For this purpose, a particular function is used as a training problem. The microlevel is a PSO algorithm used for computing the quality of a GA chromosome from the macrolevel.

The array of integers encoded into a GA chromosome represents the update order for the particles used by a PSO algorithm in order to solve a particular problem. The evolved order is embedded in a modified particle swarm optimization algorithm, as described in Section 3.3.

Roughly speaking, the fitness of a GA individual is equal to the fitness of the best solution generated by the PSO algorithm encoded into that GA chromosome. However, since the PSO algorithm uses pseudorandom numbers, it is very likely that successive runs of the same algorithm should generate completely different solutions. This problem can be handled in a standard manner: the PSO algorithm encoded by the GA individual is run multiple times (50 runs, in fact) and the fitness of the GA chromosome is averaged over all the runs.

3.3. The algorithms

The algorithms used in order to evolve the PSO update strategy are described in this section. Because the hybrid technique from [10] combines a GA and a PSO algorithm within a two-level model, two algorithms are described: one for the macrolevel (GA) and another one for the microlevel (PSO algorithm).

3.3.1. The macrolevel algorithm

The macrolevel algorithm is a standard GA [1] used in order to evolve the update order of the particles. We use the steady-state evolutionary model [36] as the underlying mechanism for our GA implementation. The GA starts by creating a random population of individuals. Each individual is a fixed-length array of integer numbers. The following steps are repeated until a given number of generations is reached: two parents are selected using a standard selection procedure [37, 38]. In order to perform selection, which is one step of the algorithm, we run twice a “tournament” [38] between two individuals randomly chosen from the population and select the winner (the one with better fitness). The parents are recombined (using one-cutting point crossover [2, 39]) in order to obtain two offspring O_1 and $O_{2;}$ a crossover point is selected in each parent. The genes after the cutting point are swapped between the parents. The offspring are then considered for weak mutation [40] (the values of one or more genes of each chromosome are replaced with other randomly generated numbers from the $\{1, 2, \ldots, \text{Swarm\_Size}\}$ set), obtaining O_1' and O_2'. The best offspring O^* (out of O_1' and O_2') replaces the worst individual W in the current population only if O^* is better than W [36]. This fact is similar to what happens in nature for longer-lived species where the offspring and parents are alive concurrently and have to compete.

3.3.2. The microlevel algorithm

The microlevel algorithm is a modified PSO algorithm [8, 41] used for computing the fitness of a GA individual from the macrolevel. The algorithm is quite different from the standard synchronous PSO algorithm [8, 41] and from the asynchronous PSO algorithm [11, 12].

The standard PSO algorithm works on two stages: one stage establishes the fitness, the *p* Best and the *n* Best values for each particle, and the other stage determines the velocity and makes the updates for each particle. The standard PSO usually works with two populations/swarms. The individuals are updated by computing the *p* Best and *n* Best values using the information from the previous population. The newly obtained individuals are added to the current population.

The asynchronous PSO algorithm works only in one stage: the fitness, *p* Best, *n* Best, velocity, and position of each particle are continuously updated. The particles are considered one by one for these modifications, following a pre-established order: the first particle, the second particle, and so on.

The developed algorithm performs all the operations in one stage only: it determines the fitness, *p* Best, *n* Best, and velocity values only when a particle is about to be updated. In this manner, the update of the current particle takes into account the previous updates in the current iteration. This PSO algorithm uses only one population/swarm. Each updated particle will automatically replace its parent. Moreover, the genes of the GA chromosome indicate the update order of the particles. The PSO individuals are not modified one by one following the initial order $(1, 2, 3, \ldots)$, but following the sequence encoded into the GA chromosome.

Reference [10] presents some numerical experiments for evolving the PSO update strategies. The results obtained have proved the effectiveness of this approach. In this paper, we just remember that the evolved PSO performs better than the standard PSO (see Figure 1).

4. LESSONS LEARNT DURING THE EVOLUTION

We will try to identify some rules in the update strategy of the swarm during the evolving process. We want to identify these rules because they can help us design better PSO algorithms. Before we detail our analysis, we will give a brief definition of the swarm rule.

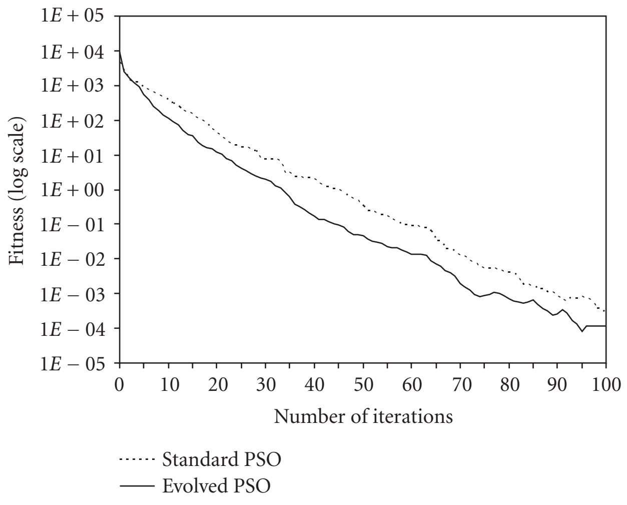


FIGURE 1: The evolution of the fitness of the best particle along the number of iterations for standard PSO (with a fixed order and equal frequency) and evolved PSO (with the optimized update order/frequency). These results have been obtained for the f_1 test function with 5 dimensions. The results are averaged over 30 runs.

A swarm rule is a sequence of operations, which can be repeatedly used in order to generate a new swarm or some subswarms. The set of rules regards the order of the updates (e.g., this order could be correlated with the particle quality: better particles of the swarm are updated and only after that, the weaker particles are), the frequency of the updates (some particles are updated more frequently than others during the search process), and the optimal swarm size. These rules could emphasize some lessons learnt by the particles during the evolutionary process.

All the numerical experiments performed in this paper use a PSO algorithm in order to optimize several well-known functions. Some of them are unimodal test functions, while others are highly multimodal problems (the number of the local minimum increases exponentially with the dimension of the problem [42]). These functions are described in Table 1.

4.1. Determining the optimal swarm size

First we analyze the evolution of the swarm size along the number of GA generations for an unimodal problem. For GA we use a population of 20 individuals which are evolved during 100 generations. Because we fix the problem size to 5 ($n = 5$), the maximal number of particles for each swarm is 14 (according to Clerc suggestions [41], a good value for the swarm size is $10 + 2 \times \sqrt{n}$). Therefore, each GA individual has 14 genes. We perform binary tournament selection, one cutting point recombination (applied with a probability of 0.8) and weak mutation (applied with a probability of 0.1). The parameters of the PSO algorithm (microlevel) are given in Table 2. The real Swarm Size is not included in this table because different PSOs may have different number of particles. However, the number of function evaluations/iteration is 14 for all the evolved PSO.

In Figure 2 we have depicted the evolution of the swarm size (a swarm whose update strategy is encoded into the GA chromosome) for several particular GA generations in the case of De Jong function 1. The function is smooth, unimodal, strongly convex, symmetric, and continuous.

We can observe that the smallest swarm from the first GA generation contains only seven particles and the largest swarm contains eleven particles. These statistics are repeated during the next generation. Starting with the third generation, the smallest swarm and the largest one tend to decrease their sizes. In the last GA generation, the majority of swarms contain only five particles, this size seems to represent the optimal swarm size for the current problem.

In Figure 3 we have depicted the evolution of the minimum, maximum, and average of the swarm size along the number of GA generations. The test problem was, again, function f_1. The average of the swarm size in a particular generation is computed as the sum of swarm sizes (the swarms encoded into all the GA chromosomes) over the number of individuals from the GA population.

We can observe (in Figure 3) that the average of the swarm size decrease from 9 in the first generation to 6 in the 25th generation and to 5.05 in the 85th GA generation. Starting with the 85th generation, the mean of the swarm size is stabilized at level 5; this value seems to be the optimal size of the swarm for the considered problem.

The same experiment was repeated for each test function presented in Table 1 (for $n = 5$). The results are depicted in Figure 4. We can observe that our model is capable of identifying the optimal swarm size. Even if during the first GA generations there are some fluctuations in the swarm sizes, these sizes tend to be more and more stable, eventually freezing at a particular level to the end of the evolutionary process.

Moreover, we can observe in Figure 4 that the optimal evolved swarm sizes for unimodal test functions have a descendent trend, while the optimal evolved swarm sizes for the multimodal test functions tend to increase along the number of GA generations.

In addition to these remarks, we want to verify our model for more difficult problems. For this purpose, we have chosen to evolve a PSO update order for three functions: Griewangk's function 8, Rosenbrock's valley, and Rastrigin's function 6, for each of them being considered 30 dimensions. Rosenbrock's function is unimodal, but it is considered to be difficult as it has a very narrow ridge (the tip of ridge is very sharp and it runs around a parabola), while the other two functions are highly multimodal and nonlinear. Rastringin's function contains millions of local optima in the interval of consideration, making it a fairly difficult problem. In the Griewangk case, the terms of the summation produce a parabola, while the local optima are above parabola level. The location of the minima is regularly distributed.

We analyze the evolution of the swarm size along the number of GA generations for these difficult problems as well. Actually, the same methodology as that from the previous experiment is applied, but 30 dimensions are considered for each function. Each swarm can contain no more than 20 particles (according to the Clerc's suggestions [41]), and a population of 100 individuals (each individual having 20

Table 1: Test functions used in our experimental study: the parameter n represents the problem size and $f_{\min}$ is the minimum value of the function. All the functions should be minimized.

	Test function	Domain	$f_{\min}$
De Jong's function 1	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	0
Axis parallel hyperellipsoid function	$f_2(x) = \sum_{i=1}^{n} (i \cdot x_i^2)$	$[-10, 10]^n$	0
Rosenbrock's valley function	$f_3(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	$[-10, 10]^n$	0
Rastrigin's function 6	$f_4(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i))$	$[-10, 10]^n$	0
Griewangk's function 8	$f_5(x) = \frac{1}{4000} \cdot \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-500, 500]^n$	0
Shifted parabola/sphere[a]	$f_6(x) = \sum_{i=1}^{n} (x_i - g_i)$	$[-100, 100]^n$	0

[a]CEC 2005 benchmark.

Table 2: The parameters of the PSO algorithm (the microlevel algorithm) used in order to compute the fitness of a GA chromosome.

Parameters		$n = 5$	$n = 30$
Number of function evaluations/iteration		14	20
Maximal number of evaluations for each PSO run	f_1	1000	7000
	f_2	1000	7000
	f_3	4000	40000
	f_4	4000	40000
	f_5	1000	9000
	f_6	1000	7000
Learning factor c_1		1.193	
Learning factor c_2		1.193	
Inertia weight		0.721	

genes) is evolved during 100 generations. The parameters of the PSO algorithm are those presented in Table 2.

In Figure 5 we have depicted the evolution of the average swarm size along the number of GA generations. We can observe that the average of the swarm size increases during the first 30 GA generations and it tends to stabilize during the rest of generations for the majority of problems.

4.2. *Which particles are updated more often?*

The evolution of update frequencies for each particle along the number of generations is presented in Figure 6. In order to compute this statistic, we calculate the average number of updates for each particle in all the GA chromosomes. For instance, it is possible to update the best particle (which is the first particle from the swarm because they are sorted based on their quality) two times in a GA chromosome and three times in other GA chromosomes. In other words, the first particle is updated for an average of 2.5 times (supposing that we have only two GA chromosomes for this example). In order to obtain a synthesis over all the evolution process, this average is computed for each particle that can be in a swarm (in our case it is possible to have a maximum of ten particles) over all the individuals from the GA population (20 chromosomes in this experiment).

In Figure 6 we have depicted the evolution of the update frequencies only for the f_1 test function (with five dimensions). Therefore, in what follows, we will detail an analysis for this case.

Taking into account the frequency of the updates performed for a particle, we can observe (Figure 6) that the most frequently updated particle is the first one. The best particle is updated for an average of 1.38 times in the first GA generation. This average rises up to 4.5 during the last generation. Note that in the first GA generation, other particles (p_2 to p_8) are updated more frequently than the first one, but taking into account the frequencies from all generations, the first particle is the most updated.

Although the swarm particles are sorted based on their fitness, the next frequently updated particle is the third particle. The average of the update frequency for this particle starts from 1.5 times in the first GA generation and increases to 3, but this value is obtained only in the 75th generation.

A similar trend can be observed for the second and the fourth particles as well. Unlike these particles, the rest

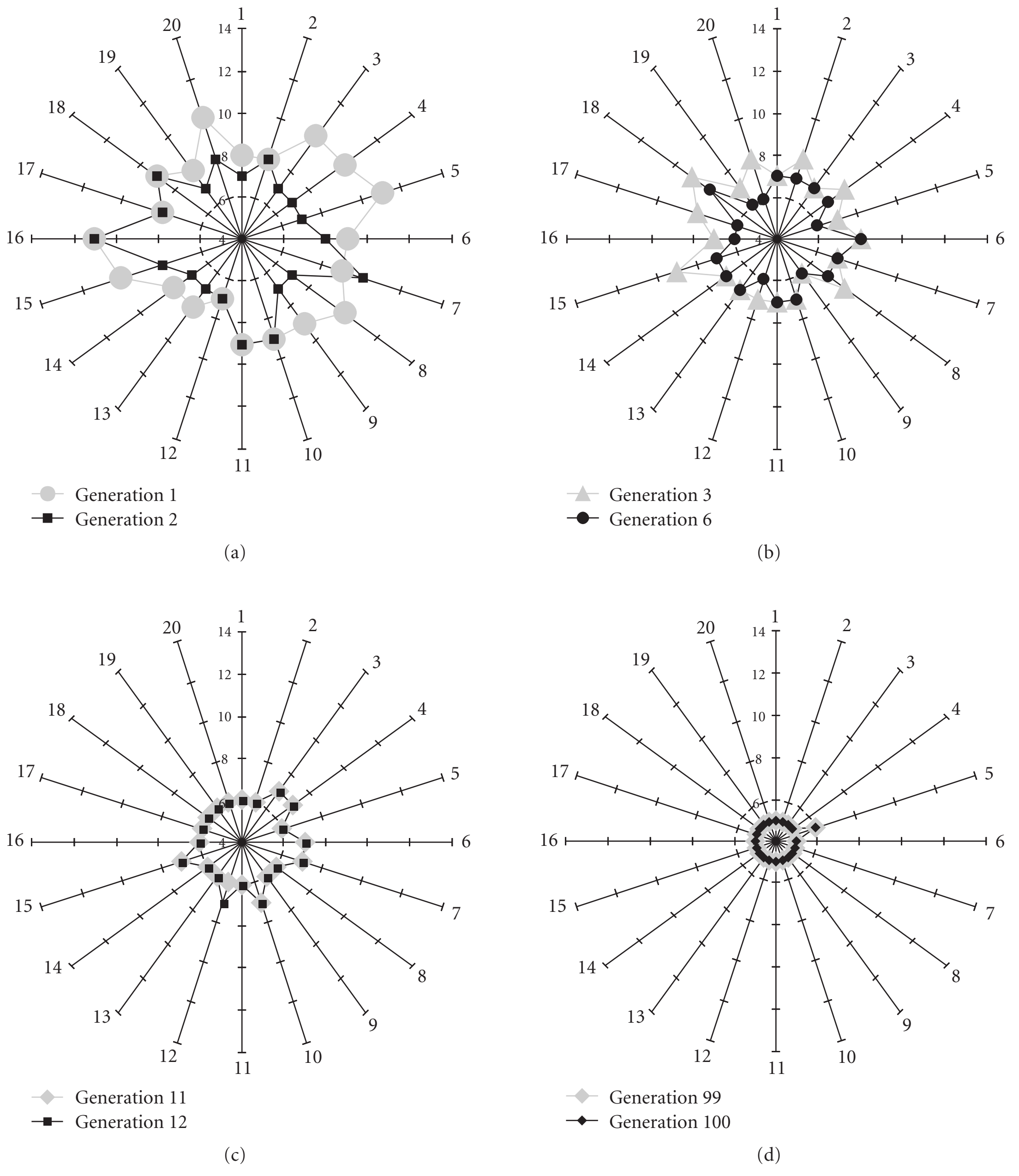


FIGURE 2: The evolution of the swarm size during the GA generations. Each ray depicts the size of a swarm whose update order is encoded into a GA chromosome (we have 20 rays because we work with a GA population made up of 20 chromosomes). These results were obtained for the f_1 test function with 5 dimensions.

of them are updated fewer times as the generation number increases. Moreover, starting with the fourth generation, the eleventh and twelfth particles are no longer updated.

Another remark regards the last two particles from the swarm: p_{13} and p_{14} (the "worst birds" of the swarm). These particles are never updated. In other words, a smaller swarm (with only twelve particles) can solve our problem. This phenomenon can also be observed if we analyze the results obtained for the other test functions: better particles are more frequently updated than the worst particles.

4.3. Analyzing the swarm update order

Several rules identified during the evolution of the PSO update strategy are investigated in this section.

For instance, the qualities of the particles from the PSO algorithm whose update order is encoded in the best GA chromosome for the f_1 test problem with 5 dimensions (obtained in a particular run) are presented in Figure 7. This chromosome is $c = (5\ 6\ 3\ 13\ 6\ 3\ 4\ 3\ 4\ 4\ 5\ 3\ 3\ 5)$ and it encodes the update order for a swarm with only 5 particles (the scaled order of updates being $(3\ 4\ 1\ 5\ 4\ 1\ 2\ 1\ 2\ 2\ 3\ 1\ 1\ 3)$).

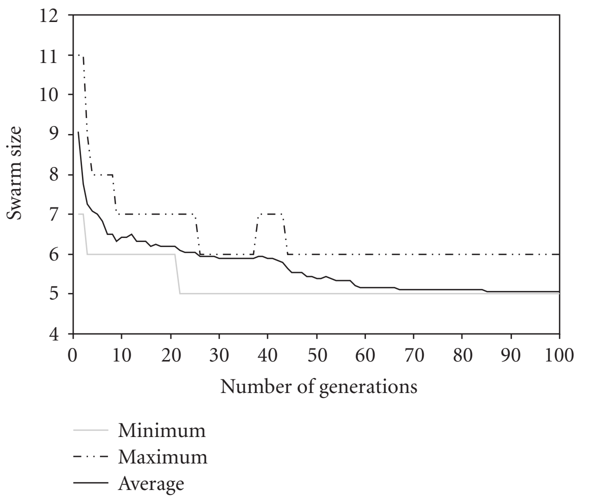


Figure 3: The evolution of the minimum, maximum, and average of the swarm size along the number of generations; the function f_1 with 5 dimensions was used as a test problem.

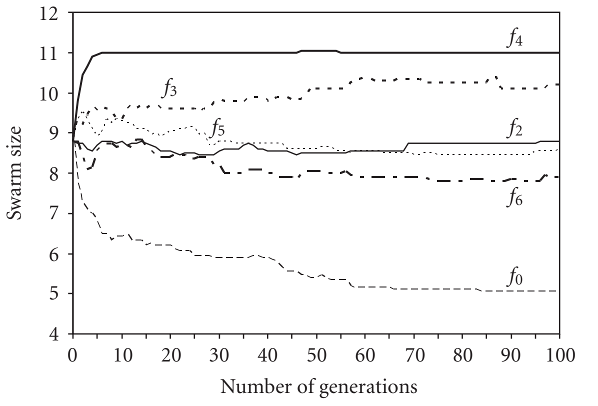


Figure 4: The evolution of the average swarm size along the number of generations for all the test functions ($n = 5$).

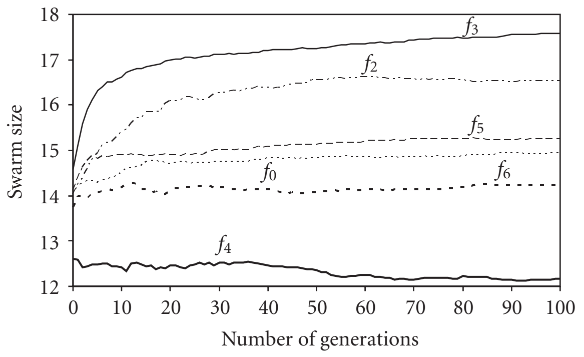


Figure 5: The evolution of the average swarm size along the number of generations for different problems. For all the problems, 30 dimensions have been considered.

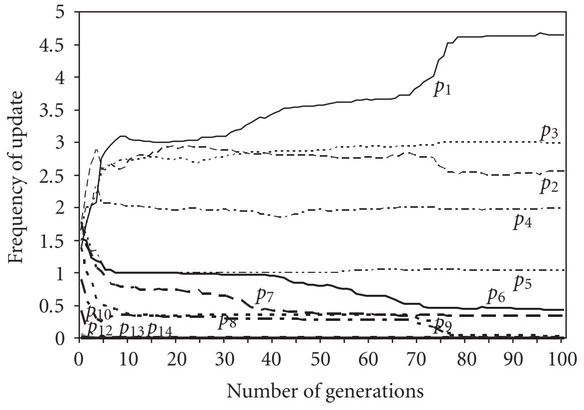


Figure 6: The evolution of the update frequency for each particle during the GA generations. By p_i we have denoted a particle, $i = \overline{1, 14}$.

The values on the Ox scale indicate the update order of the particles. The Oy values indicate the quality of the particles: the lower-sized bars indicate better particles.

The first graphic (top-left corner image) presents the initial swarm with five different particles. The order of the initialization of the particles is as follows: third, fourth, first, fifth, fourth again, first, and so on.

In the second graphic (the top-right image), the first particle is updated at the 3rd step and again, at the 6th step. After this modification, the quality of the first particle (given by the p Best value) is improved. After the 8th step and the 12th step, respectively, when the first particle is updated again, a better fitness is obtained also. Even if at the 13th step the first particle is updated another once, its quality is not improved. An quality improvement is obtained also for the second particle after the 10th step (when this particle is updated for the third time).

A more clear example can be observed in the forth graphic (the middle-right image). In this case, the particles that are modified more times during a PSO iteration (actually all the particles, except the weakest one p_5) improve their quality after each update:

(i) p_3 improves its quality after the 11th and the 14th steps;
(ii) p_4 improves its quality after the 5th step;
(iii) p_1 (the best particle) improves its quality after the 6th, the 8th, the 12th, and the 13th steps;
(iv) p_2 improves its quality after the 9th and the 10th steps.

Table 3 presents the average number of updates performed for each swarm particle of the proposed algorithm (function f_1 with 5 dimensions was considered).

As already stated, we have performed 14 updates inside a swarm even though the swarm size is smaller than 14 (in this case, some particles are updated more than once). For each particle the number of updates performed at each moment is quantified (because we have performed 14 evaluations for each swarm, we obviously have 14 time moments). The particles are indexed based on the p Best value.

The most frequently updated particle is the best one (p_1); it was updated in average for 148 times, while the less frequent updated particle is the worst one. Moreover, we can

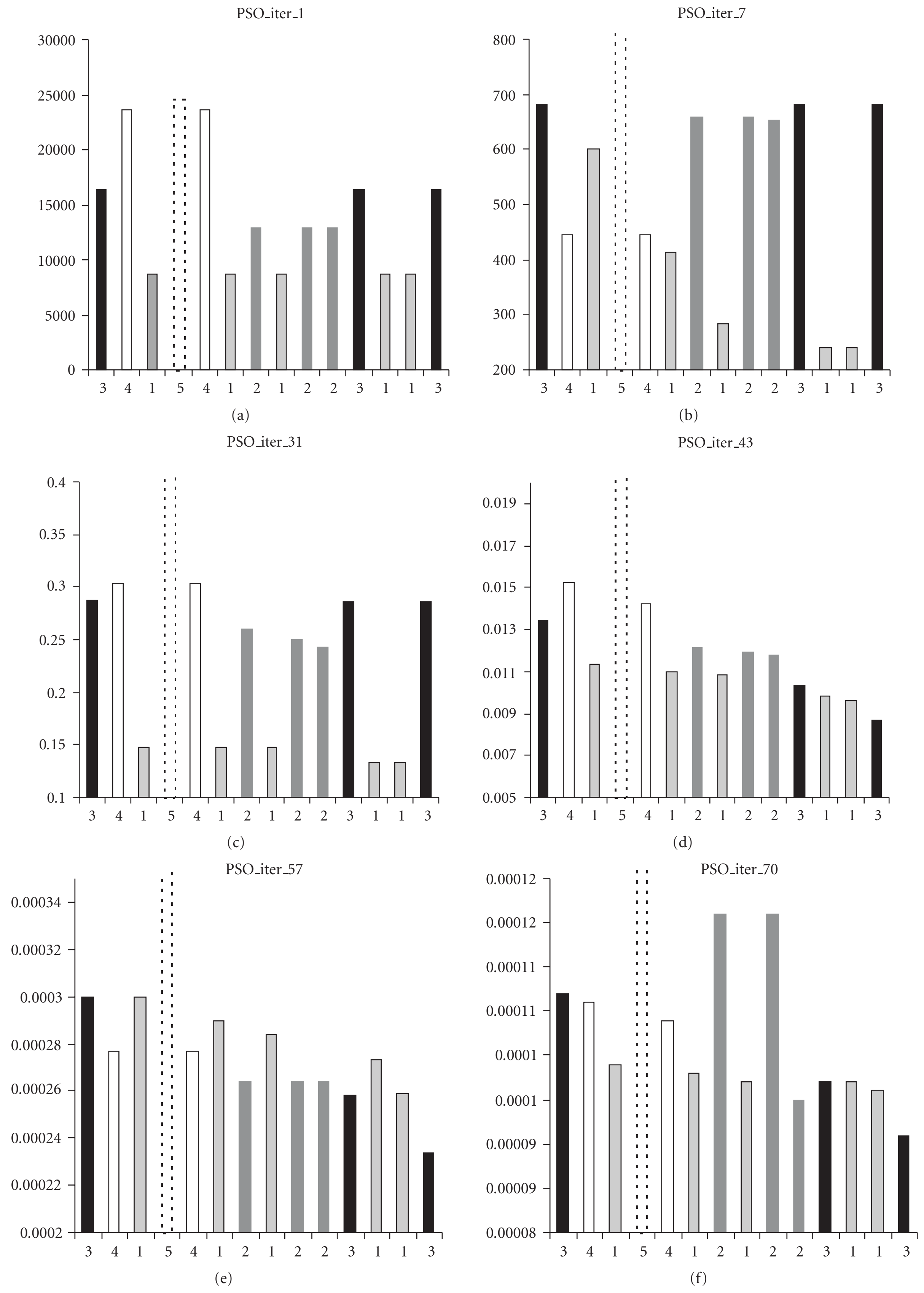


Figure 7: The update order encoded into the best GA chromosome. We have depicted different iterations of the PSO algorithm that updates the particles based on the order encoded into the GA chromosome. Different particles are represented with different colors and the height of the columns represents the quality of a particle; the lower-sized bars indicate better particles.

Table 3: The average number of updates performed for every swarm particle (P) from an iteration to another one or from an update step to another one (during the same PSO iteration).

Particle	p_3	p_4	p_1	p_5	p_4	p_1	p_2	p_1	p_2	p_2	p_3	p_1	p_1	p_3
Number of updates	32	28.6	43.6	18.8	24.8	25.4	41.8	27.6	26.4	25	27	26.4	25.4	24.8

observe that there are more improvements for a particle from a PSO iteration to another PSO iteration than from a step to another one (during the same PSO iteration).

The same conclusion can be drawn if we analyze the results obtained for the other test problems: better particles are more frequently updated than the worst particles are.

4.4. Summarizing the analysis

Analyzing the evolved update order for particles and the rules that appear during the evolution, we can conclude that

(i) the proposed model is able to determine the optimal swarm size at the end of the evolutionary process;
(ii) the evolved update order is based on the particle quality; the best particles are updated, in general, before the worst particles;
(iii) the frequency of the updates: the best particles are more frequently updated than the worst particles; even if the best particle from a swarm is updated more than once, these updates are not consecutive. There are some updates, for other particles than the best, which are performed between two successive updates for the best particle.

5. CONCLUSION AND FURTHER WORK

In this paper, we have performed an analysis of the evolution of PSO algorithms. The model of the PSO algorithm involved in this analysis has several particular properties.

(i) It is different from the standard synchronous PSO where all the particles are simultaneously updated.
(ii) It is similar to the asynchronous PSO where the particles are continuously updated, but it is more general because it considers a dynamical update order and not a predefined one (as in the asynchronous model). Moreover, the update order takes into account the particle quality.

Note that, according to the no-free-lunch theorems [43], we cannot expect to design a perfect PSO which performs the best for all optimization problems. This is why any claim about the generalization ability of the evolved PSO should be made only based on the results provided by numerical experiments.

In addition to this, we have tried to analyze the data generated during the evolution of the update strategy. This will help us understand the nature of the PSO algorithm and design PSO algorithms that use larger swarms. Based on the analysis of the evolved update strategy, we can draw some conclusions. The first one is that the best particles from the swarm are updated most frequently. Moreover, the evolution process can determine the optimal size of the swarm.

Further work will be focused on the following:

(i) several other kinds of function optimization problems will be considered, in an attempt to get more generic results, or to evolve PSOs able to solve a much more challenging and a more interesting kind of problems,
(ii) evolving more parameters of the PSO algorithm (independently, one by one, or synchronously),
(iii) using larger swarms,
(iv) using larger GA population, this will prevent the premature convergence of the macrolevel algorithm,
(v) studying the generalization ability of the evolved PSO algorithm (how well it will perform on some new and difficult problems),
(vi) designing an evolutionary algorithm able to identify by itself the rules analyzed in this paper and studying if these rules will actually improve the quality of a PSO algorithm in terms of convergence speed or accuracy of the solution.

ACKNOWLEDGMENTS

This research was supported by the National Research Grant "Natural computing of excellence," PNCDI 2007, CNCSIS, Romania. PSO source code was taken from http://www.particleswarm.info.

REFERENCES

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[3] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications, and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '01)*, vol. 1, pp. 81–86, IEEE Press, Seoul, Korea, May 2001.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[5] J. Kennedy, "The behavior of particles," in *Proceedings of the 7th International Conference on Evolutionary Programming VII (EP '98)*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., vol. 1447 of *Lecture Notes in Computer Science*, pp. 581–589, Springer, San Diego, Calif, USA, 1998.

[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, pp. 1942–1948, IEEE Service Center, Perth, Australia, November-December 1995.

[7] J. Kennedy and R. C. Eberhart, "The particle swarm: social adaptation in information-processing systems," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glaover, Eds., pp. 379–387, McGraw-Hill, London, UK, 1999.

[8] X. Hu, Y. Shi, and R. Eberhart, “Recent advances in particle swarm,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’04)*, vol. 1, pp. 90–97, IEEE Press, Portland, Ore, USA, June 2004.

[9] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’99)*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 3, pp. 1945–1950, IEEE Service Center, Washington, DC, USA, July 1999.

[10] L. Dioşan and M. Oltean, “Evolving the structure of the particle swarm optimization algorithms,” in *Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP ’06)*, vol. 3906 of *Lecture Notes in Computer Science*, pp. 25–36, Budapest, Hungary, April 2006.

[11] A. Carlisle and G. Dozier, “An off-the-shelf pso,” in *Proceedings of the Particle Swarm Optimization Workshop*, pp. 1–6, Indianapolis, Ind, USA, April 2001.

[12] B.-I. Koh, A. D. George, R. T. Haftka, and B. J. Fregly, “Parallel asynchronous particle swarm optimization,” *International Journal for Numerical Methods in Engineering*, vol. 67, no. 4, pp. 578–595, 2006.

[13] R. Poli, W. B. Langdon, and O. Holland, “Extending particle swarm optimisation via genetic programming,” in *Proceedings of the 8th European Conference on Genetic Programming (EuroGP ’05)*, vol. 3447 of *Lecture Notes in Computer Science*, pp. 291–300, Lausanne, Switzerland, March-April 2005.

[14] C. A. Coello Coello and M. Salazar Lechuga, “MOPSO: a proposal for multiple objective particle swarm optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’02)*, vol. 2, pp. 1051–1056, IEEE Service Center, Honolulu, Hawaii, USA, May 2002.

[15] X. Hu and R. Eberhart, “Multi-objective optimization using dynamic neighborhood particle swarm optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’02)*, vol. 2, pp. 1677–1681, IEEE Service Center, Honolulu, Hawaii, USA, May 2002.

[16] X. Hu, R. C. Eberhart, and Y. Shi, “Swarm intelligence for permutation optimization: case study of n-queens problem,” in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS ’03)*, Y. Com, Ed., pp. 243–246, Indianapolis, Ind, USA, April 2003.

[17] J. Kennedy and R. Eberhart, “A discrete binary version of the particle swarm algorithm,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, pp. 4104–4108, IEEE Service Center, Orlando, Fla, USA, October 1997.

[18] C. K. Mohan and B. Al-kazemi, “Discrete particle swarm optimization,” in *Proceedings of the Particle Swarm Optimization Workshop*, Indianapolis, Ind, USA, April 2001.

[19] D. Srinivasan and T. H. Seow, “Particle swarm inspired evolutionary algorithm (PS-EA) for multi-objective optimization problem,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’03)*, R. Sarker, R. Reynolds, H. Abbass, et al., Eds., pp. 2292–2297, IEEE Press, Canbella, Australia, December 2003.

[20] M. Clerc and J. Kennedy, “The particle swarm—explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[21] W. B. Langdon, R. Poli, and C. R. Stephens, “Kernel methods for PSOs,” Tech. Rep., Computer Science, University of Essex, UK, 2005.

[22] E. Ozcan and C. Mohan, “Particle swarm optimization: surfing the waves,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’99)*, pp. 1939–1944, IEEE Service Center, Washington, DC, USA, July 1999.

[23] F. van den Bergh, *An Analysis of Particle Swarm Optimizers*, Ph.D. thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[24] T. Hendtlass, “A combined swarm differential evolution algorithm for optimization problems,” in *Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE ’01)*, L. Monostori, J. Váncza, and M. Ali, Eds., vol. 2070 of *Lecture Notes in Computer Science*, pp. 11–18, Budapest, Hungary, June 2001.

[25] H. Kwong and C. Jacob, “Evolutionary exploration of dynamic swarm behaviour,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’03)*, R. Sarker, R. Reynolds, H. Abbass, et al., Eds., pp. 367–374, IEEE Press, Canbella, Australia, December 2003.

[26] K. E. Parsopoulos and M. N. Vrahatis, “Recent approaches to global optimization problems through particle swarm optimization,” *Natural Computing*, vol. 1, no. 2–3, pp. 235–306, 2002.

[27] A. E. M. Zavala, A. H. Aguirre, and E. R. V. Diharce, “Particle evolutionary swarm optimization algorithm (PESO),” in *Proceedings of the 6th Mexican International Conference on Computer Science (ENC ’05)*, vol. 2005, pp. 282–289, IEEE Computer Society, Guanajuato, Mexico, September 2005.

[28] W.-J. Zhang and X.-F. Xie, “DEPSO: hybrid particle swarm with differential evolution operator,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3816–3821, IEEE Press, Washington, DC, USA, October 2003.

[29] R. Storn and K. Price, “Differential evolution—a simple and effcient adaptive scheme for global optimization over continuous spaces,” Tech. Rep., International Computer Science Institute, Berkeley, Calif, USA, 1995.

[30] M. Oltean, “Solving even-parity problems using multi expression programming,” in *Proceedings of the 7th Joint Conference on Information Sciences*, K. Chen, Ed., vol. 1, pp. 315–318, Association for Intelligent Machinery, Cary, NC, USA, September 2003.

[31] M. Oltean and C. Grosan, “Evolving evolutionary algorithms using multi expression programming,” in *Proceedings of the 7th European Conference on Artificial Life (ECAL ’03)*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, Eds., vol. 2801 of *Lecture Notes in Computer Science*, pp. 651–658, Springer, Dortmund, Germany, September 2003.

[32] M. Oltean, “Evolving evolutionary algorithms using linear genetic programming,” *Evolutionary Computation*, vol. 13, no. 3, pp. 387–410, 2005.

[33] M. Brameier and W. Banzhaf, “A comparison of linear genetic programming and neural networks in medical data mining,” *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 17–26, 2001.

[34] M. Brameier, W. Banzhaf, et al., “Evolving teams of predictors with linear genetic programming,” *Genetic Programming and Evolvable Machines*, vol. 2, no. 4, pp. 381–407, 2001.

[35] M. Brameier and W. Banzhaf, “Explicit control of diversity and effective variation distance in linear genetic programming,” in *Proceedings of the 5th European Conference on Genetic Programming (EuroGP ’02)*, vol. 2278 of *Lecture Notes in Computer Science*, pp. 37–49, Kinsale, Ireland, April 2002.

[36] G. Syswerda, “A study of reproduction in generational and steady state genetic algorithms,” in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed., pp. 94–101, Morgan Kaufmann, San Francisco, Calif, USA, 1991.

[37] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionprograms*, Springer, NewYork, NY, USA, 1996.

[38] B. L. Miller and D. E. Goldberg, “Genetic algorithms, tournament selection, and the effects of noise,” *Complex Systems*, vol. 9, pp. 193–212, 1995.

[39] L. J. Eshelman, “The CHC adaptive search algorithm: how to have safe search when engaging in non-traditional genetic recombination,” in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed., pp. 265–283, Morgan Kaufmann, San Francisco, Calif, USA, 1991.

[40] W. E. Howden, “Weak mutation testing and completeness of test sets,” *IEEE Transactions on Software Engineering*, vol. 8, no. 4, pp. 371–379, 1982.

[41] M. Clerc, *Particle Swarm Optimization*, ISTE, London, UK, 2006.

[42] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[43] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 482032, 15 pages
doi:10.1155/2008/482032

Research Article

Particle Swarm Optimization for Multimodal Functions: A Clustering Approach

Alessandro Passaro and Antonina Starita

Dipartimento di Informatica, Università di Pisa, Largo Pontecorvo 3, 56127 Pisa, Italy

Correspondence should be addressed to Alessandro Passaro, passaro@di.unipi.it

Received 13 July 2007; Revised 18 December 2007; Accepted 8 February 2008

Recommended by Riccardo Poli

The particle swarm optimization (PSO) algorithm is designed to find a single optimal solution and needs some modifications to be able to locate multiple optima on a multimodal function. In parallel with evolutionary computation algorithms, these modifications can be grouped in the framework of niching. In this work, we present a new approach to niching in PSO based on clustering particles to identify niches. The neighborhood structure, on which particles rely for communication, is exploited together with the niche information to locate multiple optima in parallel. Our approach was implemented in the k-means-based PSO (kPSO), which employs the standard k-means clustering algorithm, improved with a mechanism to adaptively identify the number of clusters. kPSO proved to be a competitive solution when compared with other existing algorithms, since it showed better performance on a benchmark set of multimodal functions.

1. INTRODUCTION

The design of optimization algorithms is usually targeted to the goal of finding a single optimal solution for a given problem. However, in many situations, this can be less than ideal. Some problems have multiple possible solutions which are optimal according to the chosen criterion. In these cases, an optimization algorithm should ideally find all the optimal solutions; then, these solutions can be used in many different ways, depending on the application field: a specific solution can be selected from the set using additional criteria, different solutions can be preferred in different situations, or a combination of multiple solutions can be built.

In general, optimization problems are formalized identifying the function to optimize. When such a function presents multiple global optima, or local optima whose values are very close to the global one, it is said to be *multimodal*. In dealing with multimodal functions, the behavior of a standard optimization algorithm may be less than ideal. In some cases, in fact, an algorithm designed to look for a single optimum would arbitrarily pick just one of the optimal solutions, or it could even be misled by the presence of more than a single optimum and fail to converge (e.g., some genetic algorithms [1]). Thus, it is necessary either to design specific algorithms, or to modify the generic ones in order to optimize multimodal functions.

The latter approach has been taken, for example, in the field of evolutionary computation. Optimization algorithms based on the principles of Darwinian evolution have been modified introducing the concept of *niching*. In an environment with limited resources, the evolutionary process leads to the emergence of different species, which tend to exploit different niches. When applied to a search algorithm, niching allows it to divide the space in different areas and search them in parallel.

In this study we will focus on particle swarm optimization (PSO), a class of optimization algorithms which is strongly tied to evolutionary computation. We will examine how niching techniques can be introduced in the particle swarm algorithm, observing similarities and differences in relation to evolutionary algorithms. After discussing previous approaches towards a niching particle swarm, we will introduce a new approach based on clustering. Moreover, we will produce an implementation of the clustering approach, kPSO, which uses the k-means clustering algorithm to identify niches in the population.

The paper starts with a brief introduction to the PSO algorithm, which focuses on pointing out the importance of the neighborhood structure of the swarm (Section 2). Then, in Section 3, the topics of multimodal function optimization and niching are discussed and previous applications of niching in the context of PSO are presented. Section 4 is dedicated to a detailed introduction of our clustering approach to niching and its first implementation, *k*PSO. In Section 5, we show the results obtained by the new algorithm from the experiments set up to compare its performance to those of other PSO niching algorithms. Finally, in Section 6, conclusions are drawn from the current results, and future research directions are pointed out.

2. PARTICLE SWARM OPTIMIZATION

PSO is a quite recent algorithm which can be used to find optimal (or near optimal) solutions to numerical and combinatorial problems. It is easily implemented and has proven both very effective and quick when applied to a diverse set of optimization problems.

PSO was originally developed by Kennedy and Eberhart in 1995 [2], taking inspiration both in the related field of evolutionary algorithms and in artificial life methodologies. In fact, the origins of particle swarm algorithms are strongly tied to the artificial life theory of bird flocking, fish schooling, and swarming; however, PSO is also considered a kind of evolutionary algorithm, with many similarities in particular with genetic algorithms and evolutionary programming [3].

The PSO algorithm simulates the flight of a population of particles (the swarm) in a multidimensional space, where each particle represents a candidate solution to the optimization problem. Particles' flight is influenced by the best positions previously found by themselves and the other particles. The effect is that particles generally tend towards optimal positions, while still searching a wide area around them.

Here follows a formal description of the PSO algorithm applied to the minimization of a real function of several real variables. We employed one of the most used variations of the algorithm, the constriction factor PSO, based on [4]. Let d be the dimension of the search space and $f : \mathbb{R}^d \to \mathbb{R}$ the function to minimize, then $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ denotes the position of the particle $i \in (1, 2, \ldots, N)$ of the swarm, and $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$ denotes the best position it has ever visited. The index of the best particle in the population (the one which has visited the global best position) is represented by the symbol g. At each time step, t in the simulation, the velocity of the ith particle, represented as $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{id})$, is adjusted along each axis j following the equation [4]:

$$v_{ij}(t+1) = \chi \cdot \big(v_{ij}(t) + \varphi_p \cdot (p_{ij}(t) - x_{ij}(t)) + \varphi_g \cdot (p_{gj}(t) - x_{ij}(t))\big), \tag{1}$$

where φ_p and φ_g are random numbers uniformly distributed in $[0, p_{\text{incr}}]$ and $[0, g_{\text{incr}}]$. p_{incr} and g_{incr} are respectively called the *cognitive* and *social acceleration coefficients*. An illustration of the three factors influencing the velocity update of a particle is given in Figure 1.

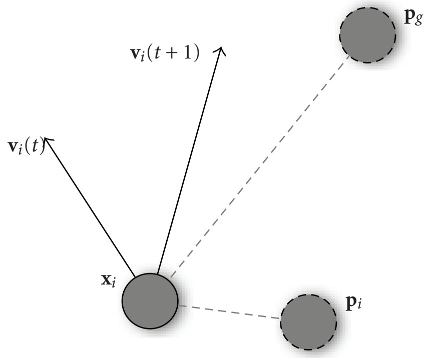


Figure 1: At each step t, a particle $\mathbf{x}_i$ updates its velocity and position. The new velocity $\mathbf{v}_i(t+1)$ is the sum of three terms: the previous velocity $\mathbf{v}_i(t)$, and two terms proportional to the distance from $\mathbf{p}_i$, the best position visited so far by the particle, and from $\mathbf{p}_g$, the best position visited so far by the whole swarm. The new position of the particle is then computed by just adding the new velocity.

The coefficient χ, called the constriction factor, results from the following equation [4]:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \tag{2}$$

where $\varphi = p_{\text{incr}} + g_{\text{incr}} > 4$.

Moreover, particles' velocity can be constricted to stay in a fixed range, by defining a maximum velocity value $V_{\max}$ and applying the following rule after every velocity updating:

$$v_{ij} \in [-V_{\max}, V_{\max}]. \tag{3}$$

In this way, the likelihood of particles leaving the search space is reduced, although indirectly, by limiting the maximum distance a particle will cover in a single step.

The new position of a particle is calculated using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \tag{4}$$

The personal best position of each particle and the global best index are updated using

$$\begin{gathered}\mathbf{p}_i(t+1) = \begin{cases}\mathbf{p}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{p}_i(t)),\\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{p}_i(t)),\end{cases}\\ g = \arg\min_i f(\mathbf{p}_i(t+1)), \quad 1 \leq i \leq N.\end{gathered} \tag{5}$$

An essential feature of the PSO algorithm is the way in which the local and global best positions, $\mathbf{p}_i$ and $\mathbf{p}_g$, and their respective acceleration coefficients, are involved in velocity updates. Conceptually, $\mathbf{p}_i$ (also known as *p*best) represents the particle's autobiographical memory, that is, its own previous experience, and the velocity adjustment associated with it is a kind of *simple nostalgia*, as it leads the particle to return to the position where it obtained its best

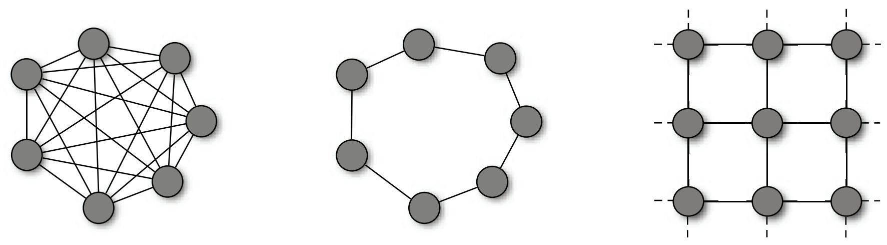


Figure 2: Examples of swarms with different social networks.

evaluation. On the other hand, $\mathbf{p}_g$ (*g*best) is a sort of group knowledge, a common standard which every single particle seeks to attain.

The overall effect is such that when particles find a good position, they begin to look nearby for even better solutions, but, on the other hand, they continue to explore a wider area, likely avoiding premature convergence on local optima and realizing a good balance between exploration of the whole search space and exploitation of known good areas.

2.1. Neighborhood structure

So far, we have described swarms in which particles had access to the accumulated knowledge of the whole population, as they were attracted by the *global* best solution, *g*best. However, this is not by far the only possible choice. More generally, in fact, we can have particles influenced by the best solutions found in their *neighborhood*, that is a specific subset of the population. The mathematical relation which defines the neighbors of each particle constitutes the *neighborhood topology* and can be described by a—typically undirected—graph in which each particle is a vertex and its *neighborhood relations* are represented by adjacent edges (see Figure 2).

The role of the neighborhood topology is to regulate the information flow among particles. In the PSO algorithm, particles communicate by sharing high-fitness solutions they previously located with their neighbors. Thus, the communication takes place along the channels defined by the neighborhood structure. On a dense topology, particles would have many neighbors, so the knowledge of a good position would be rapidly shared. Conversely, on more sparse structures, the information would spread at a slower rate.

The swarm in which a global best position is shared among all particles, also called the *g*best particle swarm, is thus the special case whose topology is a fully connected graph, so that each particle's neighborhood is the whole swarm (Figure 2(a)). At the other extreme is the *cognition-only* model [5], in which we have no edges in the graph: a particle's neighborhood is thus empty. However, in this way we would no longer have a *swarm*, but a group of isolated particles wandering around with no interactions.

A more common variation is the so-called *l*best particle swarm, in which particles are modeled to have only local interactions [6] (Figure 2(b)). The topology structure is a regular ring lattice with k edges per vertex, so that each particle is influenced by k neighbors. A commonly used *l*best case is $k = 2$, resulting in individuals affected by only their immediately adjacent neighbors. In such a swarm, it might be the case that a segment of the population converges on a local optimum, while another converges on a different optimum or keeps searching. However, if an optimum is really the best found by any part of the population, its influence slowly propagates along the ring, and eventually all the particles will be pulled towards it.

First experiments with the particle swarm used mainly the *g*best and *l*best versions, with the first generally thought to perform better in most cases. Since then, many other topologies have been proposed and studied. Suganthan [7], for example, proposed a swarm in which neighborhood relations extend over time, in order to increase cooperation towards the end of a run.

In [8], probably the first systematic study on the performances of different topologies, Kennedy and Mendes recommended the Von Neumann's architecture (Figure 2(c)), in which a particle's neighbors are above, below, and on each side on a two-dimensional lattice, to be the most promising one.

3. MULTIMODAL FUNCTION OPTIMIZATION

A function is multimodal if it has more than one optimum. However, it may have either a single *global* optimum or more than one. When trying to optimize a multimodal function, very different problems arise in each of these cases. Optimization algorithms applied to functions with a single global optimum and many local optima usually have the goal of locating the one global optimum. Thus, their main problem is to avoid deception by the other optima of the multimodal function, which would otherwise result in premature convergence to a suboptimal solution. In the other case, when the function has many optima with the same value, the goal of an optimization algorithm may be to locate *all* of them.

Most optimization techniques are designed to deal with multimodal functions of the first kind. They usually assume that there exists only a single best solution in the search space, and they put efforts in isolating it from other spurious solutions.

The situation is much different when dealing with functions with more than a single global optimum. In this case, standard techniques will usually either favor a single solution, or get confused by the multiple possible solutions and fail to converge to a single one. Therefore, specific algorithms need to be employed or the standard ones need to be modified to be more effective on multimodal functions.

3.1. Niching

In the field of evolutionary algorithms, multimodal functions are dealt with by introducing the concept of niching. Niching techniques are modeled after a phenomenon in nature where animal species specialize in exploiting different kinds of resources, resulting in several species coexisting in the same environment [1]. The introduction of this specialization, or niching, into a search algorithm allows it to divide the space in different areas and search them in parallel. The technique has proven useful when the problem domain includes multiple global and local deceptive optimal solutions.

Given the similarities between the evolutionary and the particle swarm approaches, it is natural to consider a parallel development of niching for PSO. There are, however, some peculiarities of the particle swarm approach which need to be considered.

Niching methods for genetic algorithms modify the way individuals interact with each other, in order to let them specialize on different areas of the search space, instead of converging on a single one. In the evolutionary approach individuals interact rather indirectly, as a combined effect of the selection and recombination operators. In fact, the various niching methods use different approaches, depending on the specific interaction factor they choose to modify.

The situation with the PSO algorithm is rather different. The particles in the swarm interact in a much more straightforward way than evolving individuals. In fact, the interaction is implemented by sharing knowledge of the search space with neighbor particles. Thus, the neighborhood structure of the swarm is probably the single most relevant aspect to consider in our analysis.

Let us consider for example the *g*best swarm, that is a PSO in its simplest form. The *g*best topology let particles interact with the whole swarm. As we have seen in the previous section, this provides for very quick convergence towards a *good* region of the space, which often is identified early in the search process. On a function with multiple global optima, this approach most often will simply converge on one of the optima and ignore the others.

Conversely, neighborhood topologies more sparsely connected lead to very different outcomes. A good example is the *l*best topology, where particles have a very small neighborhood. In this case, the population tends to split into small groups which explore independently the search space. Thus, they can actually locate different optima at the same time, resulting in a sort of implicit niching capability. However, since the neighborhoods in the *l*best swarm overlap, in many cases particles end up converging on the same optimum nonetheless. In the end, the implicit niching performed by the swarm relies too much upon random events during the running of the algorithm to be an efficient niching approach.

Thus, as pointed out by Engelbrecht et al. [9], the standard PSO must be modified to allow the efficient location of multiple solutions. Notwithstanding the differences between the approaches we will discuss for the particle swarm, with respect to the evolutionary ones, we will still refer to them as niching strategies.

3.1.1. Objective function stretching

Objective function stretching, introduced by Parsopoulos et al. [10, 11], was among the first strategies to modify the particle swarm algorithm to deal with functions with multiple optima. Specifically, the goal of the authors was to overcome the limitations of PSO caused by premature convergence to a local solution. The stretching approach operates on the fitness landscape, adapting it to remove local optima. Considering a minimization problem, when the swarm converges on a—possibly local—minimum, the fitness of that position is *stretched* so that it becomes a local maximum. In this way, successive iterations of the PSO algorithm will focus on other areas of the search space, leading to the identification of other solutions.

In [12], Parsopoulos and Vrahatis further developed the same technique to use it as a sequential niching approach. Likewise other sequential niching approaches, the stretching technique has some advantages, but critical issues. It requires no modifications of the underlying PSO algorithm and actually shows good performances on many multimodal functions. However, the effectiveness of the stretching transformation is not uniform on every function. In fact, in some cases it can introduce false minima, which render this method unreliable [13].

3.1.2. NichePSO

In 2002, Brits introduced the *n*best PSO, reportedly the first technique to achieve a *parallel* niching effect in a particle swarm [14]. The *n*best PSO was in particular aimed at locating multiple solutions to a system of equations, and used local neighborhoods determined by spatial proximity.

The same authors subsequently proposed a new approach which used *subswarms* to locate multiple solutions to multimodal function optimization problems: NichePSO [15]. NichePSO maintains a main swarm which can generate a subswarm each time a possible niche is identified. The main swarm is trained using the *cognition-only* model [5], which updates the velocities of particles considering only their personal best position. Since no *social* component is involved, each particle will perform a local search. On the other hand, the subswarms are trained using the GCPSO algorithm [16], which ensures convergence to a local optimum.

Niches in the fitness landscape are identified by monitoring changes in the fitness of particles. Specifically, a new subswarm is created when the fitness of a particle shows very little change over a fixed number of iterations. NichePSO has some rules to decide the absorption of particles into a subswarm, and the merging of two subswarms, which basically depends on the measure of the subswarm radius R_j.

This approach was compared empirically and shown to outperform two niching genetic algorithms—sequential niching [17] and deterministic crowding [1]—on a set of benchmark multimodal function optimization problems [18].

3.1.3. *Parallel vector-based PSO*

Schoeman and Engelbrecht proposed a different niching approach in [19], and implemented it in the vector-based PSO (VPSO). Their approach considered the two vector components of the velocity update formula in (1), which point, respectively, towards the particle's best position and the neighborhood best. When these two vectors point roughly towards the same direction (and thus have a positive dot product), the particle will modify its trajectory towards the neighborhood best, otherwise (negative dot product), it will probably head for another optimal solution. Niches are identified according to this criterion. In fact, once the dot product is determined for each particle in the swarm, the *niche radius* can be defined as the distance from the neighborhood best of the closest particle with a negative dot product.

The original VPSO algorithm identified and exploited niches in a sequential way, starting from the global best and then repeating the procedure for the particles outside its niche. In [20], the same authors developed the parallel vector-based PSO (PVPSO), a more efficient version, based on the same principles, but which followed a parallel approach. In this case, the different niches are identified and maintained in parallel, with the introduction of a special procedure which can merge two niches when they become closer than a specified threshold ϵ.

The vector-based approach has the appealing property of identifying niches by using operations on vectors which are inherent to the particle swarm algorithm. Thus, it provides an elegant way to build a particle swam for multimodal function optimization. However, when it was tested on common benchmark functions, the results showed that its performances were lower than those of other approaches, such as the NichePSO.

3.1.4. *Species-based PSO*

Another niching version of the particle swarm algorithm, the species-based PSO (SPSO) was proposed by Li [21]. The approach was the adaptation to the particle swarm of the species conserving genetic algorithm [22], of whom Li himself had been among the proponents. In particular, SPSO adopted the same procedure for determining the species seeds, which identify the niches in the population. This procedure is in fact essentially analogous to the clearing procedure proposed by Petrowski [23].

Once the species seeds have been identified, all the other particles are assigned to the niche formed by the closest seed, and the neighborhood structure is adapted to reflect the division in niches. In fact, each species seed would serve as the *n*best for all the other particles in its niche.

The SPSO niching approach can dynamically identify the number of niches in the fitness landscape, and also proved to be suitable for the application on dynamics environments [24]. However, it still requires a radius parameter σ to determine the extension of the niches. Moreover, it implicitly assumes that all the niches have roughly the same extension.

3.1.5. *Adaptive niching PSO*

In [25], Bird and Li developed a new algorithm which could adaptively determine the main niching parameters: the adaptive niching PSO (ANPSO).

The first step of the algorithm calculates r, the average distance between each particle and its closest neighbor as

$$r = \frac{\sum_{i=1}^{N} \min_{j\neq i} \|\mathbf{x}_i - \mathbf{x}_j\|}{N}. \tag{6}$$

This value is then used to determine the formation of niches. In fact, ANPSO keeps track of the minimum distance between particles over a number of steps. At each iteration, the graph $\mathcal{G}$ with particles as nodes is considered, and an edge is added between every pair of particles which have been closer than r in the last 2 steps. Niches are formed from the connected subgraphs of $\mathcal{G}$, while all the particles which end up with no edges remain outside any niches. As it can be seen from (6), the computational cost of the niche formation procedure is $O(N^2)$ with respect to distance calculations, which is quite expensive in comparison to other techniques. ANPSO executes a particle swarm simulation with constriction factor, but redefines the neighborhood topology at each step. In fact, once it determines the niches, it uses a *g*best topology for each niche, and a von Neumann topology for unniched particles. In this way, the particles which have formed a niche will tend to perform a local search around an optimum, while the others will continue searching the whole space.

4. CLUSTERING PARTICLES

The particle swarm optimization algorithm has many points in common with evolutionary algorithms. Certainly enough to allow the concept of niching, originated in the evolutionary framework, to be applied to it. However, some peculiarities of PSO must be taken into account. In PSO, there is no selection mechanism: all interactions among particles take place along the neighborhood structure. Thus a niching mechanism should operate directly or indirectly on it, modifying neighborhood relationships among particles. Another aspect by which PSO differs from the evolutionary approach is the memory particles keep of their previous best positions. This information can definitely be exploited in the discovery of niches.

We discussed in the previous section the *implicit* niching capabilities which the PSO enjoys. Though they are not sufficient to provide for an effective method for the location of multiple optima, they are considered a good starting point to build on. As stated by Kennedy and Eberhart:

After a few iterations, particles in the particle swarm are seen to cluster in one or several regions of the search space.

These clusters indicate the presence of optima, where individuals' relatively good performances have caused them to attract their neighbors, who in moving towards the optimal regions improved their own performances, attracting *their* neighbors, and so on [26].

In fact, the tendency of particles to group near the optima of the function is balanced in the end by the social influence of the particle which has found the best position. Depending on the specific neighborhood topology, this influence will extend to the other particles in the swarm at different rates. However, since the graph representing the neighborhood relationship is in general a connected graph, eventually all particles will tend to be attracted towards a single position.

The basic idea behind our approach is to dynamically adapt the neighborhood structure in order to form different niches in the swarm. This is accomplished by applying a standard clustering algorithm to identify niches and then restricting the neighborhood of each particle to the other particles in the same cluster. In this way each cluster of particles tends to perform a local search in the function domain and to locate a different optimum.

4.1. Stereotyping

The use of clustering on the particles of a swarm is not new. In [27], Kennedy introduced the concept of *stereotyping* and discussed its impact on the performance of the PSO algorithm. A stereotype is an individual or the idealization of an individual which represents the *norm* of a group. The algorithm presented by Kennedy used clustering to identify different groups in the swarm and defined the centroid of each cluster as the stereotype which the particles in that cluster would look at.

Kennedy's study was related to the social metaphor underlying the particle swarm. In the standard PSO, particles are attracted towards their personal best position and the best position among their neighbors. This attraction simulated the tendency to follow the best individual of a group. The application of stereotyping shifted this attraction from the best individual to a statistical prototype (the centroid of each cluster).

Stereotypes were identified by a cluster analysis on the particles' previous best positions, performed with a version of the k-means clustering algorithm. The experiments on stereotyping were aimed at studying the swarm behavior when the cluster center was used in place of the individual best or of the neighborhood best in the velocity update formula. Results showed that substituting the individual previous best with its cluster's center gave it some advantage, while substituting the neighborhood best with the respective centroid did not. A possible explanation is that using a stereotype in place of an individual is *on average* a quite good choice, while substituting the *best* individual with the stereotype will in general lower its fitness.

In the end, the use of clustering for a stereotyping approach brought some useful insights in the study of the particle swarm. However, its goals were completely unrelated to niching. In the following, we will introduce a new variation of the particle swarm technique which uses a very similar clustering procedure, but focuses on the identification of niches.

4.2. k-means particle swarm optimization

The algorithm we developed is the k-means particle swarm optimization (kPSO), the first version of which was introduced in [28]. It provides a basic implementation of our clustering approach to niching. In particular, we employed the standard k-means algorithm, which is probably the best-known partitional clustering algorithm [29, 30], to cluster particles according to their $p$best, the previous best position. k-means is a very simple algorithm and, as we have just seen, it had already been applied by Kennedy to cluster particles in a swarm in his research on stereotyping.

kPSO uses the clusters of particles to modify the neighborhood topology so that each particle can communicate only with particles in the same cluster. Therefore, the swarm turns in a collection of *subswarms* which tend to explore different regions of the search space. Since our goal is that the subswarms perform a rather local search, each of them uses a $g$best topology, with all the particles in a cluster connected to each other.

Clustering is performed after the swarm random initialization and then repeated at regular intervals during the swarm simulation. Between two clustering applications the swarm (or more precisely, the subswarms corresponding to the various clusters) follows its (their) normal dynamics. In fact, the multiple applications of the clustering algorithm are meant to keep track of the swarm dynamics: particles in different clusters at early stages of the simulation can end up in the same cluster as they move towards the same local optimum or, in contrast, a single cluster can be split into two as some of its particles fly towards a different optimum.

A relevant difference between our approach and other niching PSO techniques is the fact that the clustering algorithm is not applied at each step of the swarm simulation, but only every c step. The rationale behind this choice lies in the natural tendency of the swarm to cluster around the function optima. The application of a clustering procedure should just enhance this natural tendency and, above all, maintain the clusters over time by blocking communication between particles in different clusters. However, if the algorithm reidentified the subswarms at each step, the particles would not have time to follow their natural dynamics. Therefore, we let them evolve for some steps following the standard PSO dynamics.

In the following, we will discuss in detail the consequences of this approach, and set up some experiments to determine the actual effect it has on the performance of the algorithm. In the mean time, it should be noted that, as a side effect, this choice has the obvious advantage of introducing a smaller computational overhead, since the clustering procedure is applied only $\sim T/c$ times (T being the total number of simulation steps).

After performing the clustering, a cutting procedure is applied: we measure the average number of particles per cluster N_{avg} and proceed by removing the exceeding particles from the clusters which are bigger than the average. To this

```
procedure Identify Niches
    Cluster particles' pbests with the k-means algorithm.
    Calculate the average number of particles per cluster, $N_{avg}$.
    Set $N_u = 0$.
    for each cluster $C_j$ do
        if $N_j > N_{avg}$ then
            remove the $N_j - N_{avg}$ worst particles from $C_j$.
            Add $N_j - N_{avg}$ to $N_u$.
        end if
        Adapt the neighborhood structure for the particles in $C_j$.
    end for
    Reinitialize the $N_u$ un-niched particles.
end procedure
```

Algorithm 1: The procedure to identify niches in the *k*PSO algorithm.

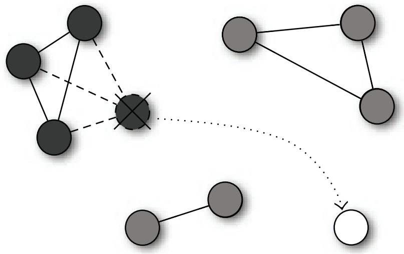


Figure 3: Particles are linked to all the other particles in the same cluster. Clusters with a number of particles greater than the average are reduced and the exceeding particles reinitialized (see the particle in white).

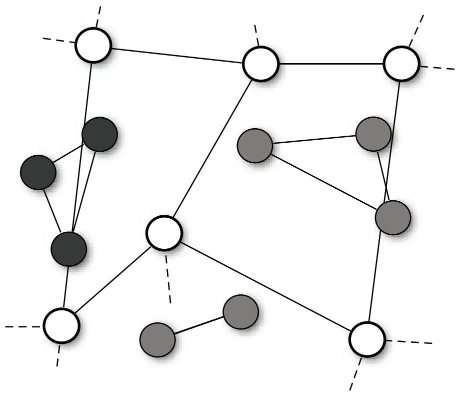


Figure 4: The particles not belonging to any niche (drawn in white) are organized in a von Neumann lattice topology.

end, we keep the particles in each cluster C_j sorted by their previous best position fitness, so that we can remove the worst $N_j - N_{avg}$ particles of the cluster.

The goal of the cutting procedure is to avoid the formation of overcrowded niches, which would end up in a waste of computational power, as too many particles would explore the search space around a single optimum. Instead, the N_u exceeding particles removed from the overcrowded clusters can be randomly reinitialized, in order to explore new areas.

Niches are modeled after the reduced clusters, as described in Algorithm 1. Each particle's neighborhood is set to the ensemble of particles in the same cluster. Thus we will have *k* (the number of clusters) niches whose particles are fully—connected (see Figure 3), realizing a *g*best topology in each niche.

The remaining particles, which were removed from the overcrowded clusters and reinitialized, are used to explore the search space for possible new interesting areas. In the first implementation of the algorithm, these *unniched* particles had no connections, thus they performed the kind of nondeterministic hill-climbing associated with the *cognition-only* model [5]. However, the first experiments showed that the poor capabilities of this model hindered the search for new solutions, leading to a less-than-optimal performance on complex functions.

Thus, we modified the *k*PSO algorithm so that the unniched particles are organized in a von Neumann lattice neighborhood (see Figure 4). In this way, when particles are reinitialized, they start performing a more efficient search for new solutions, which allows the algorithm to identify all the multiple optima.

Once the neighborhood structure has been set both for the niched and the unniched particles, the algorithm performs a fixed number *c* of steps of the constriction-factor PSO. In this phase there are only two aspects in which *k*PSO differs from the standard algorithm.

(i) The particles assigned to a niche C_j will have their velocities clamped to $V_{max} = 2\sigma_j$ (see (8)), which is proportional to the width of the cluster.

(ii) The unniched particles will behave as in a constriction-factor PSO with the von Neumann lattice topology.

After *c* steps, the clustering and cutting procedures are repeated. The pseudocode for *k*PSO is reported in Algorithm 2.

The application of the *k*PSO algorithm requires to set a few additional parameters with respect to the standard PSO.

```
procedure kPSO
    Initialize particles with random positions and velocities.
    Set particles' pbests to their current positions.
    Calculate particles' fitness.
    for step $t = 0 \to T - 1$ do
        if $t \bmod c = 0$ then                    Every $c$ steps.
            Execute the procedure Identify Niches.
        end if
        Update particles' velocities.               Perform the standard PSO step.
        Update particles' positions.
        Recalculate particles' fitness.
        Update particles' and neighborhood best positions.
    end for
end procedure
```

Algorithm 2: *k*PSO algorithm pseudocode.

We already discussed the first one, c, the number of PSO steps between two clustering applications.

The other additional parameters are strongly tied to the specific clustering algorithm chosen, k-means. Since the k-means algorithm depends on the initial assignment of the seeds, it must be repeated a number r_k of times for a single clustering application to reduce its variability. The optimal clustering is chosen by selecting the one with minimal squared error (or scattering) J

$$J = \sum_j \sigma_j, \tag{7}$$

where σ_j is the variance of the cluster C_j

$$\sigma_j^2 = \frac{1}{N_j - 1} \sum_{\mathbf{p} \in C_j} \|\mathbf{p} - \mathbf{m}_j\|^2 \tag{8}$$

with $\mathbf{m}_j$ the center of the jth cluster, and N_j its size. In our experiments we found out that it is generally sufficient to repeat the application of k-means a few times to obtain good results. Thus we set $r_k = 10$.

The second parameter for the k-means algorithm is k, the number of clusters, which is also the most problematic. In fact, the value for k that leads to the best performance of the algorithm is strongly dependent on the number m of optima of the function under consideration. When the latter is known, k can be set to a value which is slightly larger than it. In this way, the algorithm will maintain a sufficient number of clusters to be able to identify all the optima, while possibly forming spurious clusters, that is, clusters not corresponding to any optima.

Regarding those parameters which are in common with the standard constriction-factor PSO, we used the common values of $p_{\text{incr}} = g_{\text{incr}} = 2.05$ and $\chi \simeq 0.730$.

4.3. Estimating the number of clusters

One of the major issues with our approach in *k*PSO is the fact that one needs to specify the number of clusters k, which is strongly related to the number of optima of the function under consideration. Unfortunately, often the number of optima of a function cannot be estimated *a priori*; rather, it would be desirable that it could be discovered by the optimization algorithm itself.

The need to know the number of clusters is inherited by *k*PSO from the specific clustering algorithm we chose, k-means, although it is a problem which is common to a wide class of clustering algorithms. Significant research has focused on the problem of estimating k, using very different approaches [30]. Here we will discuss an approach which leads to the estimation of k by the optimization of a criterion function in a probabilistic mixture-model framework.

In this framework, the objects to be clustered (the particles' position in our case) are assumed to be generated by a mix of several probabilistic distributions. In particular, to each different cluster corresponds a different distribution. Thus, a general model for the whole set of objects will be given by a combination of different probability densities, the mixture densities [31], which can be defined as

$$\rho(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{k} P(C_j)\rho(\mathbf{x} \mid C_j, \boldsymbol{\theta}_j). \tag{9}$$

In (9), $P(C_j)$ is the prior probability of an object $\mathbf{x}$ to belong to the cluster C_j, with the condition that

$$\sum_{j=1}^{k} P(C_j) = 1. \tag{10}$$

Next, $\rho(\mathbf{x} \mid C_j, \boldsymbol{\theta}_j)$ is the conditional probability distribution associated with the same cluster (component density). Finally, $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k)$ is the vector of the parameters of the distributions for all clusters.

Assuming the number of clusters k, the prior probabilities, and the form of each of the probability densities to be known, it is possible to estimate the best parameters of a model by maximizing the probability of generating all the observations (maximum likelihood) as follows:

$$\rho(\{\mathbf{x}_1,\dots,\mathbf{x}_N\} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \rho(\mathbf{x}_\mathbf{i} \mid \boldsymbol{\theta}), \tag{11}$$

or, in a logarithm form

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \rho(\mathbf{x}_\mathbf{i} \mid \boldsymbol{\theta}). \tag{12}$$

It has been proven [29] that the k-means algorithm implicitly performs a maximization of the likelihood under the assumption that the component densities are spherical Gaussian distributions.

Extending on these considerations, finding the number of clusters k is equivalent to fitting the model with the observed data and optimizing some criterion. This can be done by applying the k-means algorithm with k varying in a range of possible values and then choosing the best clustering.

However, the problem arises of choosing a valid criterion to select the best clustering. Here the maximum likelihood is not helping, since it invariably leads to the choice of the highest k. Thus, a large number of alternative criteria have been proposed, which combine concepts from information theory. The most typical examples include the Akaike's information criterion (AIC) [32, 33] and the Bayesian information criterion (BIC) [34, 35].

4.3.1. *The bayesian information criterion*

In our approach, we chose to integrate the Bayesian information criterion (BIC), also known as the Schwarz criterion, in order to estimate the best choice of k. Given a clustering $\mathcal{C}$, formed by k clusters of a swarm with N particles, we can calculate its BIC value with

$$\mathrm{BIC}(\mathcal{C}) = \ell(\mathcal{C}) - \frac{p}{2} \cdot \log N, \tag{13}$$

where $\ell(\mathcal{C})$ is the log-likelihood of the clustering and p is the number of parameters. The formula for the log-likelihood can be calculated considering that we are assuming components densities in the form of spherical Gaussians

$$\rho(\mathbf{x} \mid \mathbf{m}_j, \sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^d}} e^{-(1/2\sigma_j^2)\|\mathbf{x}-\mathbf{m}_j\|^2}, \tag{14}$$

and the class probabilities are estimated with the ratios between the number of particles in a cluster and the total number of particles:

$$P(C_j) = \frac{N_j}{N}. \tag{15}$$

With a few mathematical transformations, the log-likelihood of the clustering, $\ell(\mathcal{C})$, can be written as

$$\ell(\mathcal{C}) = \sum_{i=1}^{N} \log \rho(\mathbf{x}_\mathbf{i} \mid \mathcal{C}) = \sum_{j=1}^{k} \ell(C_j) - N \cdot \log N, \tag{16}$$

where the term $\ell(C_j)$ is the log-likelihood for each cluster C_j:

$$\begin{aligned}\ell(C_j) = &-\frac{N_j}{2} \cdot \log 2\pi - \frac{N_j \cdot d}{2} \cdot \log \sigma_j^2 \\ &- \frac{N_j - 1}{2} + N_j \cdot \log N_j.\end{aligned} \tag{17}$$

The number of parameters p is given by the sum of $k-1$ class probabilities (given the condition in (10)), $d \cdot k$ centroid coordinates, and the k variance estimates σ_j. Thus we have

$$p = (k-1) + d \cdot k + k. \tag{18}$$

In the improved version of the kPSO algorithm, at each clustering application, k-means is thus repeated with varying values for k (usually in the range from 2 to $N/2$), then the clustering with the highest BIC is chosen. In this way, there is no need to set a value for k at the beginning of the run. Rather, it can vary across the execution of the algorithm, as the particles in the swarm slowly discover new promising areas and organize in new niches.

5. EXPERIMENTS

In [28], we presented a first version of the kPSO algorithm without the automatic selection of the number of clusters and evaluated its performance in comparison to those of NichePSO [15] and parallel vector-based PSO [20]. Results showed how kPSO could achieve similar or better results, requiring quite low computational resources.

Thus, in this study we chose to compare the enhanced version of kPSO with automatic selection of the number of clusters (see Section 4.3.1) with other two niching algorithms, the SPSO [21] and ANPSO [25] algorithms, which probably represent the state-of-the-art of niching algorithms for the particle swarm. The study was conducted on the same set of benchmark functions used in [25] and reported in Table 1.

The first function in the study was the Branin RCOS function, a two-dimensional function with 3 global optima in the region we considered (see Figure 5). The second function was the six-hump camel back (Figure 6), with only 2 global optima, but several deceptive local ones. The third function, the Deb's 1st function, was a one-dimensional function with 5 equally spaced global optima (see Figure 7). The fourth one was the Himmelblau function (Figure 8), again two-dimensional, with 4 global optima. Finally, the 5th and last function of the study was the Shubert 2D function, with 18 global optima grouped in 9 clusters and surrounded by a very high number of local optima (Figure 9).

In order to show how the improved version of our algorithm can effectively identify niches surrounding the optima of a function, we report in Figure 10 several significant steps of kPSO running on the Branin RCOS function $M1$. In the snapshots we plotted, it is shown how at the beginning of the run the particles of the swarm are randomly distributed on the search space. Then, during the first steps of the particle swarm simulation, they naturally start to split in different groups, roughly adapting to the fitness landscape.

Table 1: Benchmark functions.

Function	Equation	Domain
Branin RCOS	$M1(x,y) = \left(y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x) + 10$	$-5 \le x \le 10$ $0 \le y \le 15$
Six-Hump camel back	$M2(x,y) = -4\left[\left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2\right]$	$-1.9 \le x \le 1.9$ $-1.1 \le y \le 1.1$
Deb's 1st function	$M3(x) = \sin^6(5\pi x)$	$0 \le x \le 1$
Himmelblau	$M4(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$-6 \le x,\ y \le 6$
Shubert 2D	$M5(x,y) = \sum_{i=1}^{5} i\cos[(i+1)x + i]\sum_{i=1}^{5} i\cos[(i+1)y + i]$	$-10 \le x,\ y \le 10$

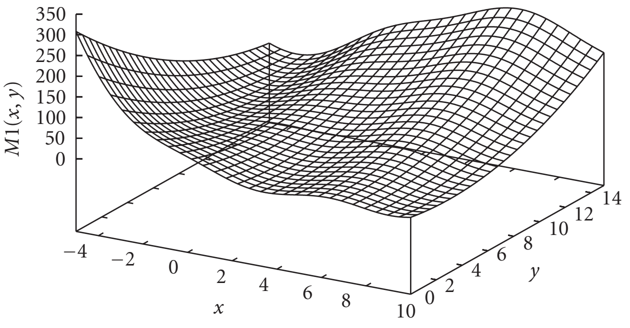


Figure 5: Function $M1$: Branin RCOS.

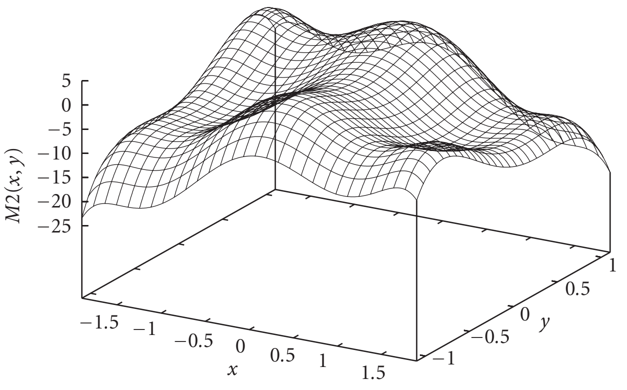


Figure 6: Function $M2$: Six-Hump camel back.

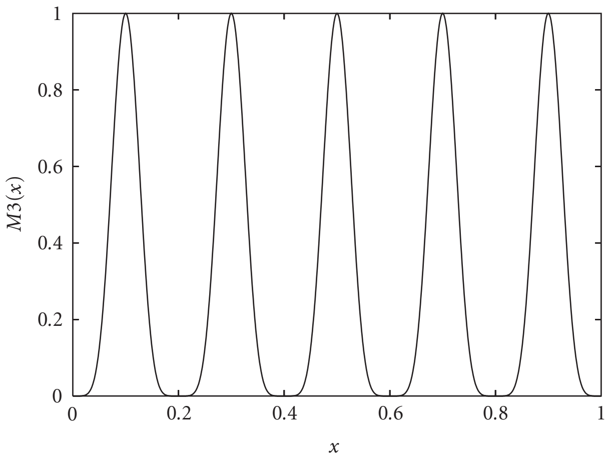


Figure 7: Function $M3$: Deb's 1st function.

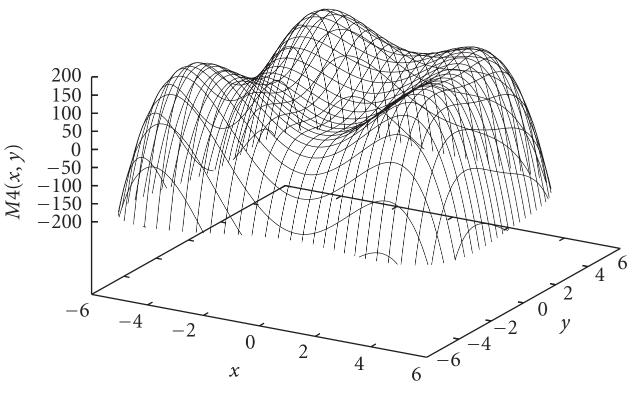


Figure 8: Function $M4$: Himmelblau.

When the first clustering application occurs (see Figure 10(c)), it identifies 4 niches, 3 of which actually correspond to the global optima, while the other is a spurious one. In the particular case that is shown here, the algorithm will eventually converge to exactly 3 clusters, as it is shown in Figure 10(f), at the 62nd simulation step. However, such convergence is not the final goal of *k*PSO: as long as a sufficient number of clusters has formed to cover all the optima, the presence of spurious clusters is really of no harm to the optimization algorithms. Besides, in early phases of the optimization, the presence of additional clusters can be helpful in locating new interesting regions of the search space.

In Table 2 we report the results obtained on the five benchmark functions with *k*PSO, SPSO, and ANPSO. The experiments were carried out so that for each function the optimization algorithm was executed with two different populations sizes. Each execution was repeated 50 times, setting the threshold for the maximum number of iteration to 2000 simulation steps.

In all the experiments, the goal of the optimization algorithm was to locate all the global optima with an accuracy of $\epsilon = 0.00001$ and to maintain all of them for at least 10 simulation steps. Since all the algorithms could actually fulfill the goal within the maximum number of 2000

Table 2: Number of evaluations required to find all global optima (mean and standard deviation).

Function	Particles	SPSO	ANPSO	*k*PSO
*M*1	30	3169 ± 692	5220 ± 3323	**2084** ± **440**
	60	6226 ± 1707	6927 ± 2034	**3688** ± **717**
*M*2	30	2872 ± 827	2798 ± 857	**1124** ± **216**
	60	5820 ± 1469	4569 ± 1316	**2127** ± **341**
*M*3	30	2007 ± 703	6124 ± 2465	**1207** ± **688**
	60	4848 ± 2092	8665 ± 2974	**1654** ± **705**
*M*4	30	4096 ± 731	16308 ± 13157	**2259** ± **539**
	60	7590 ± 2018	17168 ± 12006	**3713** ± **570**
*M*5	300	166050 ± 42214	**82248** ± **10605**	**81194** ± **45646**
	500	219420 ± 80179	**114580** ± **18392**	**117503** ± **77451**

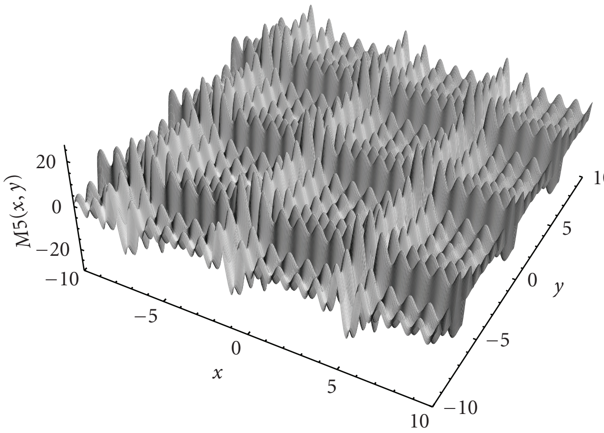


Figure 9: Function $M5$: Shubert 2D.

iterations, we only report in Table 2 their performance in term of the number of function evaluations they required.

The first four functions in the benchmark, $M1, \ldots, M4$, proved to be quite easy to optimize. A population size of just 30 particles was more than sufficient to identify all the global optima with a quite low number of function evaluations. In Table 2, we report also the results with a population of 60 particles in order to compare them to those in [25]. *k*PSO proved to have a clear advantage on all of these functions with respect to SPSO and ANPSO.

In the experiments with functions $M1$ to $M4$ we used a value of $c = 10$ for the number of steps between two clustering applications. With higher values, the performance of the algorithm did not vary significantly, meaning that it was a sufficient number of steps for the particles to adjust to the changed social structure after a clustering application.

A special analysis was conducted regarding the optimization of function $M5$, which was the most complex function of the set, in particular because it presented many local optima surrounding the global ones. As shown in Figure 11, even if *k*PSO was able to find all the optima in all the runs, the number of evaluations required changed greatly depending on the value of c. A small value for c would hamper the tendency of the particles to follow the standard PSO dynamics, since they would be rearranged in different clusters too often. A too high value would instead freeze the swarm on fixed clusters for a long time, resulting in worse performance, since a high number of function evaluations would be wasted. We found that a good value was $c = 50$, so we used it for the other experiments (Table 2 and Figure 12).

The higher degree of complexity of function $M5$ also led to the use of a much larger population. In the experiments reported in Table 2, we employed the two population sizes of 300 and 500 in order to compare the results to those of ANPSO and SPSO as reported in [25]. The performance of *k*PSO appears to be quite better than those of SPSO and comparable to those of ANPSO, although they exhibits a larger variance over the 50 runs. However, *k*PSO successfully located all the optima in $M5$ also with smaller population sizes, as plotted in Figure 12, requiring rather fewer function evaluations than the other algorithms. In particular, with a population of only 200 particles, *k*PSO was able to locate all the global optima performing just 59165 ± 32646 function evaluations.

The set of experiments was intended to test the performance of the improved version of *k*PSO and to compare it with two of the best existing niching PSO algorithms, SPSO and ANPSO. The results we obtained were quite good: the *k*PSO algorithm, with the new mechanism to adaptively determine the number of clusters, outperformed the other algorithms by a good margin on most of the test functions.

The situation was rather more elaborated regarding the most complex function of the group, the Shubert 2D function. In this case, *k*PSO still showed very good results on average, but also a very large variance. Moreover it needed an adjustment of the newly introduced parameter, the number of simulation steps between two clustering applications. In fact, while for the simple functions this number was kept constant at a quite low value, the application to the Shubert function required a much higher value, in a way that is consistent with what we found out in earlier experiments on the influence of c.

5.1. Computational cost

Another important aspect to consider in the evaluation of the different approaches to niching is the computational cost of the niching procedure itself. In general, when considering the

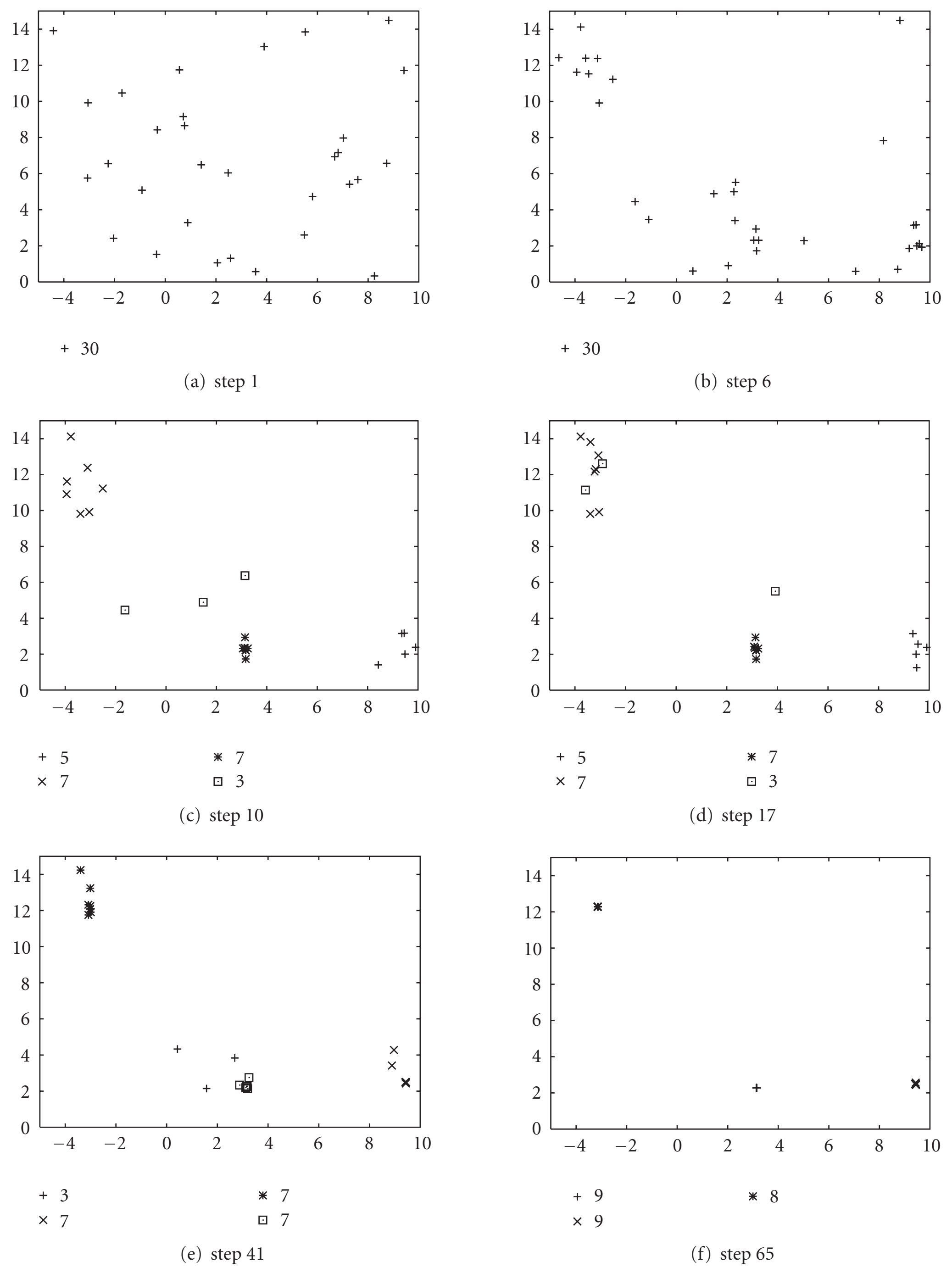


Figure 10: Significant steps of a *k*PSO run on function $M1$. The plots show the previous best positions of the particles. In the first step, particles are randomly distributed on the search space and their previous best position is the current position. At step 6, the swarm is naturally splitting in different groups of particles around the global optima of the function. At step 10, the first clustering is performed, which identifies 4 niches. Step 17 shows how particles belonging to different niches can get mixed up when their niches are actually related to the same global optimum. Finally, at step 65, the algorithm stabilizes on 3 niches corresponding to the 3 global optima.

efficiency of an optimization algorithm, the most significant measure is the one we used in our last experiments, that is the number of function evaluations the algorithm requires to reach the predefined goal. In fact, calculating the function value is usually the single operation with the highest computational cost in a real-world problem.

However, in certain cases, the computational overhead added by other operations required by the optimization algorithm should be taken into account. In the comparison we are discussing, all the algorithms implement a version of the particle swarm approach, thus they share roughly the same (quite low) computational costs related to the simulation of

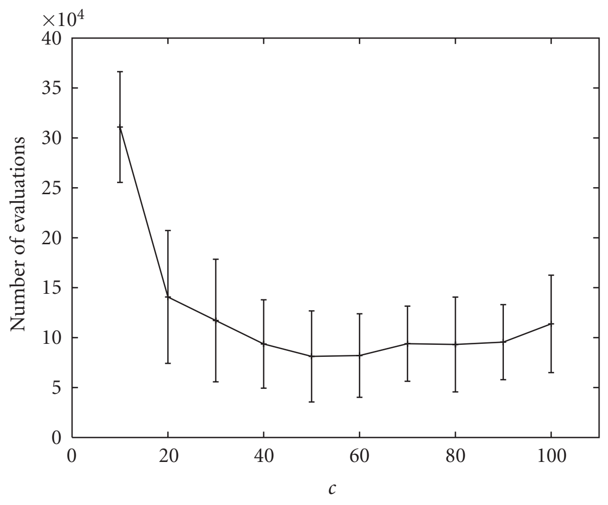


FIGURE 11: Performance of kPSO on function $M5$ using different values for c, the number of steps between clusterings, and a fixed population size of $N = 300$.

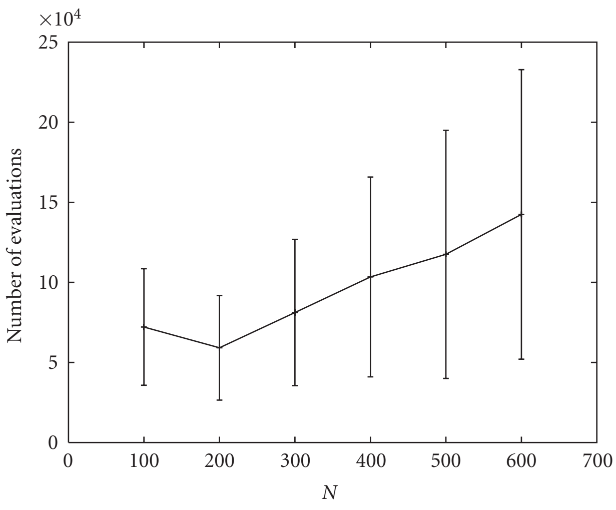


FIGURE 12: Performance of kPSO on function $M5$ using different population sizes N, and a fixed value of $c = 50$ for the number of steps between two clustering applications.

particles' dynamics. In this regard, the only aspect in which they differ is the specific niching procedure they use. The computational cost of these procedures can be evaluated in terms of the number of distance calculations. For the SPSO algorithm, this leads to a computational cost at each iteration that is $O(N \cdot k)$, where N is the swarm size and k the number of seeds or niches. The ANPSO algorithm, instead, implements a more complex procedure, which has a cost of $O(N^2)$.

Our approach basically inherits the computational complexity of the k-means algorithm, that is $O(N \cdot k \cdot r_k)$. But considering that the improved version actually repeats the k-means application in order to determine the best number of clusters, the final computational cost is in the order of $O(N^2 \cdot r_k)$. It should be noted, however, that kPSO, unlike the other algorithms, only executes the niching procedure every c simulation steps, thus reducing its computational overhead by the same factor. Considering the order of magnitude of the parameters r_k and c we used in all the experiments, the average cost per iteration is $O(N^2 \cdot r_k/c) \simeq O(N^2)$, therefore in the same order as ANPSO. A summary of the computational overhead of the various niching procedure, averaged per single iteration, is given in Table 3.

TABLE 3: Computational overhead of the niching procedure employed in different algorithms, averaged per single iteration. N is the swarm size, k the number of niches, r_k the number of runs of k-means, and c the number of steps before clustering.

Algorithm	Cost
SPSO	$O(N \cdot k)$
ANPSO	$O(N^2)$
kPSO	$O(N^2 \cdot r_k/c) \simeq O(N^2)$

While in this analysis it emerges that the SPSO algorithm is the one which adds the lowest computational overhead to identify niches, compared to ANPSO and kPSO, whether this is really important or not depends mainly on two factors.

(i) In many real-world optimization problems, the most relevant component of the computational cost is the number of function evaluations. Thus, an algorithm with a highest overhead in term of distance calculations, but which requires significantly less evaluations, is surely to be preferred.

(ii) Fine-tuning the parameters adds another computational cost. This is just one of the main issues Bird and Li tried to assess with the introduction of ANPSO over SPSO. The latter, in fact, requires the specification of a fixed niche radius, thus it realistically needs to be executed multiple times to find the best value, while ANPSO can adaptively determine it during a single run. In this regard, kPSO is more similar to ANPSO, as it can adaptively determine k, the most problem-dependent parameter. However, it still needs a value for c to be chosen, which can significantly influence its efficiency, but does not really hinder its ability to locate all the optima.

6. CONCLUSIONS

In this paper, we introduced a new approach to niching in particle swarm optimization based on clustering. Then we described kPSO, our implementation which uses the k-means clustering algorithm to identify niches and the Bayesian information criterion to determine the number of clusters. The resulting algorithm maintains essentially the same structure as the standard PSO. In fact the niching effect is obtained just by influencing the neighborhood structure of the swarm, while the dynamics of the single particles remain unchanged. Thus, each subswarm created by the clustering process performs a local search with the same efficiency as the standard PSO.

The *k*PSO algorithm was competitive with other niching algorithms for the particle swarm. In particular, it markedly outperformed one of the best existing algorithms, SPSO, in term of the number of function evaluations needed to discover all the optima of the test functions. Even though the computational cost of the clustering procedure in *k*PSO is higher than that of SPSO, we thought that it is balanced by the improved ability to adapt with far less manual tuning to different function landscapes, and by the sensible gain in performance. In this respect, the advantages of *k*PSO are rather similar to those of ANPSO, another interesting algorithm which adaptively determines the main niching parameters. The two algorithms, in fact, introduce a comparable computational overhead with the procedure to identify niches. *k*PSO, however, showed higher or comparable performance in all the tests we conducted.

Research on the clustering approach will continue in several directions. To begin with, the *k*PSO algorithm will be put to test with higher-dimensionality benchmark functions together with real-world problems in order to better assess its capabilities. Another interesting research line involves the development of a more flexible version of *k*PSO, which will avoid the need to set *c*, the number of steps between clusterings. This could be accomplished for example by developing an algorithm which could estimate when a new clustering application is needed, rather than performing it at fixed intervals. Further research will also be devoted to investigate the employment of different clustering algorithms rather than the *k*-means.

REFERENCES

[1] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. thesis, University of Illinois at Urbana-Champaign, Champaign, Ill, USA, 1995.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, IEEE Service Center, Perth, Western Australia, November-December 1995.

[3] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 601–610, Springer, San Diego, Calif, USA, March 1998.

[4] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[5] J. Kennedy, "The particle swarmml: social adaptation of knowledge," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '97)*, pp. 303–308, Indianapolis, Ind, USA, April 1997.

[6] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th IEEE International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[7] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1959–1962, Washington, DC, USA, July 1999.

[8] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1671–1676, Honolulu, Hawaii, USA, May 2002.

[9] A. P. Engelbrecht, B. S. Masiye, and G. Pampara, "Niching ability of basic particle swarm optimization algorithms," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 397–400, Pasadena, Calif, USA, June 2005.

[10] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis, "Stretching technique for obtaining global minimizers through particle swarm optimization," in *Proceedings of the Particle Swarm Optimization Workshop*, pp. 22–29, Indianapolis, Ind, USA, April 2001.

[11] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis, "Improving particle swarm optimizer by function "stretching"," in *Nonconvex Optimization and Applications*, vol. 54, pp. 445–457, chapter 3, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[12] K. E. Parsopoulos and M. N. Vrahatis, "Modification of the particle swarm optimizer for locating all the global minima," in *Artificial Neural Networks and Genetic Algorithms*, V. Kurkova, N. C. Steele, R. Neruda, and M. Karny, Eds., Computer Science Series, pp. 324–327, Springer, Wien, Austria, 2001.

[13] F. van den Bergh, "An analysis of particle swarm optimizers," Ph.D. thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[14] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Solving systems of unconstrained equations using particle swarm optimization," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '02)*, vol. 3, pp. 100–105, Hammamet, Tunisia, October 2002.

[15] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAl '02)*, L. Wang, K. C. Tan, T. Furuhashi, J. H. Kim, and X. Yao, Eds., vol. 2, pp. 692–696, Singapore, November 2002.

[16] F. van den Bergh and A. P. Engelbrecht, "A new locally convergent particle swarm optimizer," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '02)*, vol. 3, pp. 96–101, Hammamet, Tunisia, October 2002.

[17] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993.

[18] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Scalability of niche PSO," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 228–234, Indianapolis, Ind, USA, April 2003.

[19] I. L. Schoeman and A. P. Engelbrecht, "Using vector operations to identify niches for particle swarm optimization," in *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems (CCIS '04)*, vol. 1, pp. 361–366, Singapore, December 2004.

[20] I. L. Schoeman and A. P. Engelbrecht, "A parallel vector-based particle swarm optimizer," in *Proceedings of the 7th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA '05)*, Coimbra, Portugal, March 2005.

[21] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO '04)*, K. Deb, R. Poli, W. Banzhaf, et al., Eds., vol. 3102 of *Lecture Notes in Computer Science*, pp. 105–116, Springer, Seattle, Wash, USA, June 2004.

[22] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, 2002.

[23] A. Petrowski, "Clearing procedure as a niching method for genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 798–803, Nagoya, Japan, May 1996.

[24] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '04)*, vol. 1, pp. 98–103, IEEE Service Center, Portland, Ore, USA, June 2004.

[25] S. Bird and X. Li, "Adaptively choosing niching parameters in a PSO," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06)*, vol. 1, pp. 3–10, ACM Press, Seattle, Wash, USA, July 2006.

[26] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Fransisco, Calif, US, 2001.

[27] J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '00)*, vol. 2, pp. 1507–1512, La Jolla, Calif, USA, July 2000.

[28] A. Passaro and A. Starita, "Clustering particles for multimodal function optimization," in *Proceedings of ECAI Workshop on Evolutionary Computation*, Riva del Garda, Italy, August 2006.

[29] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, NY, USA, 1973.

[30] R. Xu and D. Wunsch II, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 3, no. 16, pp. 645–678, 2005.

[31] G. J. McLachlan and D. Peel, *Finite Mixture Models*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, NY, USA, 2000.

[32] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[33] M. P. Windham and A. Cutler, "Information ratios for validating mixture analyses," *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 1188–1192, 1992.

[34] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[35] D. Pelleg and A. Moore, "X-means: extending k-means with efficient estimation of the number of clusters," in *Proceedings of the 17th International Conference on Machine Learning (ICML '00)*, pp. 727–734, Morgan Kaufmann, Stanford, Calif, USA, June-July 2000.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 315950, 8 pages
doi:10.1155/2008/315950

Research Article

Optimizing the Operation Sequence of a Multihead Surface Mounting Machine Using a Discrete Particle Swarm Optimization Algorithm

Yee Ming Chen[1] and Chun-Ta Lin[2]

[1] *Department of Industrial Engineering Management, Yuan Ze University, 135 Far-east Road, Chung-li, Taoyuan 320, Taiwan*
[2] *Department of Information Management, Yu-Da College of Business, 168 Hsueh-fu Road, Tan-wen Village, Chaochiao Township, Miao-li 361, Taiwan*

Correspondence should be addressed to Yee Ming Chen, chenyeeming@saturn.yzu.edu.tw

Received 19 July 2007; Revised 12 February 2008; Accepted 25 March 2008

Recommended by Riccardo Poli

The optimization of the nozzle selection, for sequencing component pick and place operations, is very important to the efficiency of multihead surface mounting machine (SMM). The nozzle change operation, that is, choosing the best nozzle head relative pair that is most effective for picking and placing components onto the printed circuit board (PCB), significantly adds to the overall assembly time. In this paper, as a practical application, we focus on a discrete particle swarm optimization (DPSO) algorithm for multihead SMM which is used to minimize the number of nozzle change operations and pick and place operations simultaneously. To evaluate the performance of the proposed algorithm, we test it on assembly tasks of PCBs through simulations. The results of computer experiments show that this DPSO algorithm was superior to the standard PSO algorithm.

1. INTRODUCTION

The optimization of feeder setup and component placement sequences is very important to the efficiency of a surface mount machine (SMM). The increase of the number of components to be placed on a single board has made the reduction of assembly time probably the most important issue to further cut down production costs and increase productivity. The assembly of printed circuit board (PCB) on a production line can be divided into four process-planning subproblems: grouping (i.e., assigning PCB types to product families and to machines), allocation (i.e., assigning nozzles to the heads), arrangement (i.e., assigning reels to slots on the feeder rack), and sequencing (i.e., the components' pick-and-place operations). Many publications have been devoted to these complex optimization subproblems, and various models and techniques have been presented, for example, by Ball and Magazine [1] and Brad et al. [2]. These subproblems are tightly intertwined, and each of them is very difficult to solve to optimality. For example, the quality of a component pick-and-place sequence is dependent on the feeder setup and vice versa [2]. Various approaches have been proposed to improve sequence of placement points and/or feeder assignment for the PCB assembly process [3]. Egbelu et al. [4] investigated assigning components to feeder slots and sequencing component placement onto the PCB in order to minimize the total assembly cycle time. They classified machines depending upon whether the PCB table and the feeder carrier were stationary or not. They developed rules such as the centroid rule, the proportion rule, the partition rule, and the weighted rule to initially assign components to feeder slots. To obtain the component insertion sequence, they modeled the problem as a traveling salesman problem (TSP). Finally, by having the component insertion sequence and the initial feeder setup, Egbelu et al. [4] converted the component slot assignment into a quadratic assignment problem (QAP) where the cutting plane and exchange heuristic were used. Results showed that the assembly cycle times were minimized when both the feeder rack and PCB table were able to move (Egbelu et al. [4]). Some researchers (see, e.g., [5]) have tackled the subproblems independently but also have some researchers (see, e.g., Ho and Ji [6]) preferred to solve these subproblems in an integrated way. The literature is rich with work that has tackled this subject, especially investigating how to improve the efficiency of SMM [7, 8]. The technological characteristics of the SMM can also influence the nature of some of the subproblems to be solved and the formulation of the associated models [9]. As a result, many researchers solved the problem as a unique problem since the problem relies heavily on the machine characteristics (Ho and Ji, 2003). This causes difficulties in applying or comparing the various approaches from the literature.

In the recent years, a heuristic approach was developed to solve the subproblems. Genetic algorithms (GAs) have been applied successfully to a wide variety of optimization arrangement/sequencing problems [10]. The GA and its many versions have been popular in academia and the industry mainly because of their intuitiveness, ease of implementation, and the ability to effectively solve highly nonlinear, mixed integer optimization problems that are typical of complex engineering systems. The drawback of the GA is its expensive computational cost. Particle swarm optimization (PSO), introduced by [11] is a relatively recent heuristic search method whose mechanics are inspired by the swarming or collaborative behavior of biological populations. PSO is similar to the GA in the sense that these two heuristics are population-based search methods. In other words, PSO and the GA move from a set of points (population) to another set of points in a single iteration with likely improvement using a combination of deterministic and probabilistic rules. Critical ingredients for the success of PSO are simplicity, ease of operation, and great flexibility.

When the SMM has more than one nozzle per head (or even a single nozzle per head), choosing an effective nozzle group (or a nozzle) is important since a nozzle change operation is time consuming. Optimizing the pick and place operations without considering the nozzles changing operations may not be efficient since it may cause many unnecessary nozzle changes that will significantly reduce the machine throughput. However, to date, only a few researchers [3] have tackled the nozzle optimization problem. Due to its global and local exploration abilities, simplicity in coding and consistency in performance, PSO has been widely applied in many fields beyond its original applications to the solution of continuous optimization problems. In our research, we mainly use discrete data to process SMM problems. Therefore, developing a mechanism to realize discrete optimization problem is attractive. Hence, in this paper, we develop a discrete particle swarm optimization (DPSO) algorithm for nozzles selection and sequencing components pick and place operations in a SMM equipped with multiple placement heads. The objective of the algorithm is to minimize the nozzle selection for sequencing component pick and place operations so as to increase the machines throughput.

The organization of this paper is as follows. Section 2 is a description of the SMM problem and its assumptions. In Section 3, we propose a DPSO algorithm for multihead SMM which is used to minimizing the nozzle changes

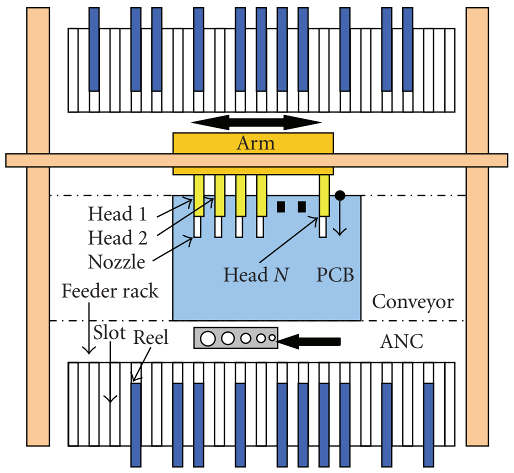


Figure 1: Schematic diagram of a multihead surface mounting machine.

and minimizing the pick and place operations. The fourth section discusses the experiment results we have gained for the heuristics described in this paper applied to 10 test print circuit boards of SMM. The last section covers the conclusions we have drawn from this research and gives a brief overview of planned future research.

2. SMM DESCRIPTION AND ASSUMPTIONS

SMMs are classified into five categories based on their specification and operational methods. These are: dual delivery, multistation, turret style, multihead, and a sequential SMM. In general, each SMM has a feeder rack (sometimes called a feeder carrier), a PCB table (or worktable), head(s), nozzle(s) (or gripper(s)), and a tool magazine. The multihead surface mounting machine is becoming increasingly popular. Its strong point is that the mounting speed is high though the price is low. Components are supplied to the machine by reels. Each reel contains only one type of components. Two feeder racks are located on both sides of the conveyor, and each feeder rack has a number of slots for reels. The actual pick-and-place unit is the arm, which has multiple heads. The arm always moves from one location to another along a straight line at a constant speed. Therefore, the distance between two coordinate positions (x_1, y_1) and $(x_2 y_2)$ can be defined as $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$ which is called the Euclidean metric. The heads pickup components from feeders and place them on the board. Since the distance between adjacent heads is equal to a multiple of the distance between two adjacent slots, the arm can pick multiple components simultaneously by one pickup operation. This operation is called simultaneous pickup. Each head has a nozzle that grips and holds a component until it is placed on the board. Nozzles of different diameters are used depending on the size of the component to be retrieved. The nozzle is automatically changed at the automatic nozzle changer (ANC) (see Figure 1) when it cannot grip the required component. The assumptions of the problem are as follows.

(1) The number of heads is not limited. That is the arm may have any number of heads.

(2) The different types of components cannot be contained in one reel.

(3) The number of reels required to complete the job is, at most, equal to the capacity of the feeder racks.

The above assumptions are practical and true in real settings.

3. DISCRETE PARTICLE SWARM OPTIMIZATION ALGORITHMS

PSO is inherently continuous but recently the discrete particle swarm optimization algorithm has been proposed to solve discrete problems successfully. Ucar and Tasgetiren [12] proposed a discrete particle swarm optimization algorithm to determine a sequence of n jobs to be processed through m machines that minimizes the number of tardy jobs. Other papers proposing discrete particle swarm optimization algorithms can be found in Shen et al. [13], Tasgetiren et al. [14], Onwubolu [15], and Clerc [16].

Since, for each printed circuit board, the location of each component has been decided, the critical decisions left for optimization are where the different reel groups should be placed in feeder slots. Moreover, in the multihead case, the optimal solution does not only require the local optimization of a single head but also the global optimization of the head group. In addition, there is only a finite number of feeder slots, and each feeder slot can only contain one reel of component(s). These properties match the properties of particle swarm optimization that is why we propose the DPSO algorithms to solve the SMM's arrangement/sequencing problems.

Considering the characteristics of the multihead case, we decompose the component arrangement/sequencing problem into three phases: assignment of reels to heads, construction of reel-groups, and the assignment of component placement. In phase I, heads assignment algorithm, the most critical decision is to assign jobs to each head with a balanced loading simultaneously. Therefore, in order to balance the loading of each head, first, we construct a head-nozzle relative matrix; then, an algorithm is developed to minimize the variance of component loadings on each head based on the component types and their associated nozzle types to produce an optimal solution of head assignment.

3.1. *Phase I: heads assignment algorithm*

In order to balance the loading of each head, the heads assignment algorithm essentially solves the following linear programming problems. Let H be the head set where H_i denoted the nozzle types which assigned to head i. Let N be the nozzle set where N_j is the set which contained the components associated with the nozzle type j, and let M_j be the number of components associated to nozzle type j. Then,

Step 1. Construct a head-nozzle relative matrix $A(i,j)$ through randomly assigning each nozzle to each head and satisfied:

$$H = AN, \quad \text{where } A(i,j) = 0 \text{ or } 1, \quad \sum_j A(i,j) = 1, \quad \text{for every } i. \tag{1}$$

$A(i,j) = 1$ means that the nozzle j has been assigned to head i; in contradiction to this, $A(i,j) = 0$ means that there is nothing assigned to head i.

Step 2. Calculate the fitness of head assignment to minimize the variance of component loading on each head:

$$\text{Min Fitness} = \sum_i \left(\sum_j M_j \times A(i,j) - \text{Average\_loading} \right)^2. \tag{2}$$

3.2. *Phase II: reel grouping optimization*

In phase II, since the time for the operation of pick-and-place for each component is constant, the only difference is how many nozzle changes are needed in the pick-and-place process. The assignment of nozzles to each head was optimally determined by phase I; therefore, the next objective is to minimize the total number of times of nozzle changes in the pick-and-place process according to the different type of nozzle needed. In addition, each reel on the feeder rack can only hold one kind of component. The optimal number of reel groups needed to minimize the number of times of nozzle changes can be developed as follows.

Step 1. Based on phase I, construct a matrix of component assigned for the head group, that is $H = AN \approx H_i = \cup A(i,j) \times N_j$.

Step 2. From the component matrix, randomly assign components to each head and generate the reel group, $\{\text{reel\_group}_i(j,k) \mid 1 \le i \le \text{No\_reel\_group}\}$, where reel_group$_i(j,k)$ means that the component assigned to head k on the jth row for the ith reel group.

Step 3. Determine whether it is necessary to change the nozzle from the consecutive component group assignment, if so, calculate the number of nozzle changes and create a new reel group, otherwise, continue the assignment process.

Step 4. Evaluate the performance of assignment to minimize the number of nozzle changes and repeat Steps 1 and 3 until a stop criterion is met.

3.3. *Phase III: DPSO for assignment of component placement*

In Phase III, the assignment of component placement based on particle swarm optimization approach is developed in order to minimize the total time of pick-and-place calculated by the traveling distances, one dimension Euclidean distance times the unit assembly time in the entire process. The discrete particle swarm optimization (DPSO) approach is used to search the optimal locations for the reel group. Since the number of feeder slots in each surface mounting machine is finite and discrete, each particle cannot fly outside the range of feeder rack, that is, the position of each particle is constrained. In order to satisfy these constrains, the $V_{\max}$ velocity update equation is used in this case, and the maximum velocity which each particle can fly is calculated as the half of the number of feeder slots. Furthermore, Zhang et al. [17] proposed a new boundary handling method, which is called as "periodic mode." It provides an infinite search space for the flying of particles, which is composed of periodic copies of original search space with same fitness landscape. Since each feeder slot can only contain one reel, some modification on the position update equation with boundary constraints handling should be made to protect against infeasibility.

In this phase, assignment of component placement, the fitness is determined as follows.

$$\begin{aligned} A = {} & \text{the distance that the arm moves from automatic nozzle} \\ & \text{changer (ANC) to reel to pickup component} \\ = {} & |P(N(\text{the last change for this pick-up}) \\ & - P(\text{reel\_group}_i(j,1)))|, \end{aligned} \tag{3}$$

where $P(N(\text{the last change for this pick-up})$ denotes the last position of associated nozzle type needed to changed in this pickup sequence.

$$\begin{aligned} B = {} & \text{the distance that arm moves from reel to broad and to} \\ & \text{place the components} \\ = {} & |X_{\text{id}}(i) - P(\text{reel\_group}_i(j,1))| \\ & \text{for } 1 \le i \le \text{Opt\_No\_reel\_group}, \end{aligned} \tag{4}$$

where $P(\text{reel\_group}_i(j,1))$ denotes the position of component assigned to head 1 in the reel group i of jth row on the print board.

$$\begin{aligned} C = {} & \text{the distance between the different components positions} \\ & \text{in this placement sequence} \\ = {} & \sum_{k=1}^{\text{No\_of\_head}-1} |P(\text{reel\_group}_i(j,k)) \\ & - p(\text{reel\_group}(j,k+1))|. \end{aligned} \tag{5}$$

$$\begin{aligned} D = {} & \text{the distance between the last position of this placement} \\ & \text{and the position of current reel group} \\ = {} & |X_{\text{id}}(i) - P(\text{reel\_group}_i(j,\text{No\_of\_Heads}))|. \end{aligned} \tag{6}$$

$$\begin{aligned} E = {} & \text{the distance between the last position of this placement} \\ & \text{and the position of current nozzle to be released on ANC} \\ = {} & |P(N(\text{the current nozzle type needed to be changed}) \\ & - P(\text{reel\_group}_i(j,\text{No\_of\_Heads}))|. \end{aligned} \tag{7}$$

$$\begin{aligned} F = {} & \text{the distance between the last position of this placement} \\ & \text{and the position of next reel group on the feeder slot} \\ & \text{without changing nozzle} \\ = {} & |P(\text{reel\_group}_i(j,\text{No\_of\_Heads})) - X_{\text{id}}(i+1)|. \end{aligned} \tag{8}$$

$$\begin{aligned} G = {} & \text{the distance that the arm moves from nozzle to nozzle to} \\ & \text{change different nozzle type on ATC} \\ = {} & |P(N(i)) - P(N(j))|, \end{aligned} \tag{9}$$

where $P(N(i))$ denotes the position of nozzle type i.

$$H = \text{the time needed for nozzle's change.} \tag{10}$$

The algorithm for component placement is given below.

Step 1. An individual particle of reel groups is randomly assigned to a feeder rack, and an initial solution based on the optimal assignment of reel groups from Algorithm II is generated according to the fitness function:

$$\begin{aligned} \text{fitness} = {} & \sum_i \sum_j \sum_k (A+B+C+D+E+F+G) + H \\ & \times \text{No\_of\_Changes}, \end{aligned} \tag{11}$$

where $1 \le i \le \text{Opt\_No\_Reel\_Group}$, $1 \le j \le \text{Row}(i)$, and $\text{Row}(i)$ denotes the number of rows in reel group i, $1 \le k \le \text{No\_of\_Heads}$.

Step 2. Run the iterations of population with the following update equation.

(i) Update the velocity with maximum velocity, $V_{\max}$, constrained within the range of feeder rack. The velocity update equation is influenced simultaneously by the individual's experience and neighbor's experience as follows:

$$\begin{aligned} V_{\text{id}}^{\text{new}} = {} & w \times V_{\text{id}}^{\text{old}} + c_1 \times \text{rand}_1() \times (P_{\text{id}} - X_{\text{id}}) + c_2 \\ & \times \text{rand}_2() \times (P_{\text{gd}} - X_{\text{id}}), \end{aligned} \tag{12}$$

$$\text{if } V_{\text{id}} > V_{\max}, \quad V_{\text{id}} = V_{\max}, \tag{13a}$$

$$\text{else if } V_{\text{id}} < -V_{\max}, \quad V_{\text{id}} = -V_{\max}, \tag{13b}$$

where w is the inertia weight in the range [0, 1], the cognitive parameter $c_1 \in [0.5, 2]$, and the social parameter $c_2 \in [1, 3]$.

(ii) Update the particle's position using the new velocity:

$$X_{\text{id}}^{\text{new}} = X_{\text{id}}^{\text{old}} + V_{\text{id}}^{\text{new}}. \tag{14}$$

(iii) If the reel allocation is out of boundary, then a boundary constraint handling modification on position updating is made as the follows:

$$X_{\text{id}}^{\text{new}} = X_{\text{id}}^{\text{old}} + (-1) \times V_{\text{id}}^{\text{new}}. \tag{15}$$

If the reel allocation is infeasible, then go to (i) again.

Table 1: The comparison of experiment results between DPSO and the standard PSO.

	Average time		Opt_solution_time		Improvement		Paired t test
Components	DPSO	PSO	DPSO	PSO	DPSO	PSO	$\alpha = 0.05$
21	372	375	311	335	16.3%	10.63%	0.2893
42	1345	1351	1166	1172	13.3%	13.25%	0.4187
63	1686	2243	1477	1990	12.38%	11.29%	0.0000*
84	3916	4732	3411	4203	12.90%	11.17%	0.0000*
105	5925	6033	5282	5389	10.86%	10.67%	0.3853
126	5986	7277	5357	6607	10.50%	9.21%	0.0000*
147	7038	8562	6287	7937	10.67%	7.30%	0.0000*
168	8087	9684	7443	8845	7.96%	8.67%	0.0000*
189	9091	11013	8667	10417	4.67%	5.41%	0.0000*
210	11793	12379	10005	11617	15.16%	6.15%	0.0122*

*means that the paired t-test is significant, that is, $p < \alpha/2$.

Table 2: The CPU time comparison between DPSO and the standard PSO.

	Components									
	21	42	63	84	105	126	147	168	189	210
Approaches										
DPSO	16.5	32	51	77	94	117	137	163	190	204
PSO	16.65	33	52	82	106	120	152	173	196	216
Saving (%)	0.91	3.03	1.92	6.10	11.32	2.50	9.87	5.78	3.06	5.56

*Saving = (CPU time of PSO-CPU time of DPSO)/ CPU time of PSO *100%.

Step 3. Based on the fitness function, update P_{id} and P_{gd}.

Step 4. Repeat Steps 2 and 3 until the stopping criterion is met.

Figure 2 shows the flowchart of the DPSO for optimization of the operation sequence placement in multiheaded SMM.

4. EXPERIMENT RESULTS

According to the DPSO approach proposed in Section 3, even though we only considered one machine to one printed circuit board, there are many possibilities of combination of processing procedures. Depending on what nozzle is assigned to what head or what component is assigned to what reel of each head different solutions will emerge. In this section, we consider a practical SMM that has 3 heads for a $300 \times 250\,\text{mm}^2$ board. In order to emphasize the effectiveness of DPSO, we made a comparison of the standard PSO approach with our DPSO approach according to the quality of solution found. This is measured by the optimal solution found, the average traveling time (unit assembly time), the improvement percentage, and paired t-test for significance. The computer simulation is performed in the operating environment of Window XP Professional with Pentium III CPU, 846 MHz, 1 GB RAM. The software was coded in MATLAB 7.0.

Since the exhaustive examination of the quality of solutions requires a large number of problems, the algorithm is tested on a set of randomly generated problems. The component locations on the print board are randomly determined. In these simulations, both DPSO and the standard PSO use the same parameters. Namely, the population size is 10, and the relative parameters of velocity update equation, the cognition parameter c_1, and the social parameter c_2 are set equal to 2 to result in the most effective search of the problem domain [11]. Furthermore, a linear decreasing inertia weight w is used in this simulation.

To illustrate the performance of this meta-heuristic approach, the number of placements and component types ranged between 21 and 210 components to be placed and of up to 34 different component types. Tables 1 and 2 show the results of 10 different PCB boards with 20 runs with a different random seed for each problem case.

In order to compare the performances of opt_solution_time, we define the improvement percentage of average time versus opt_solution_time as

$$\text{Improvement} = \frac{(\text{Average Time} - \text{Opt\_Solution\_Time})}{\text{Average Time}} \times 100\%. \tag{16}$$

The results are shown on the fourth column of Table 1. In addition, under the confidence level of $\alpha = 0.05$, a paired t-test mentioned on the fifth column of Table 1 is calculated as follows: $t = (\overline{X} - \overline{Y})\sqrt{(n(n-1))/\sum_{i=1}^{n}(\hat{x}_i - \hat{y}_i)^2}$ with $n-1$ degree of freedom and $\hat{x}_i = (x_i - \overline{X})$, $\hat{y}_i = (y_i - \overline{Y})$, where $\overline{X}$ is the sample mean of the observed set $\{x_i = i \mid 1 \leq i \leq 20\}$,

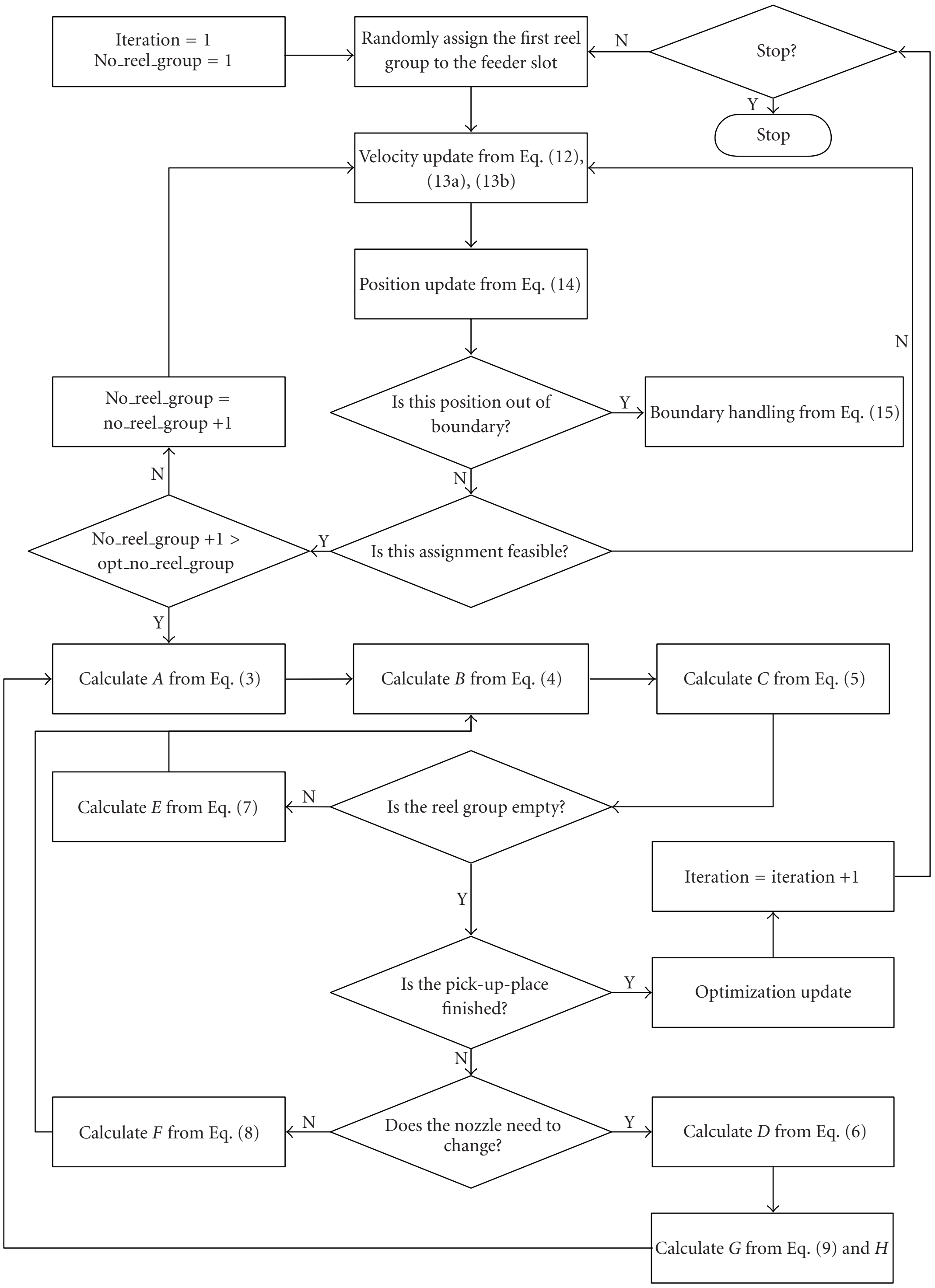

Iteration = 1
No_reel_group = 1
Randomly assign the first reel group to the feeder slot
Stop?
N
Y
Stop
Velocity update from Eq. (12), (13a), (13b)
Position update from Eq. (14)
Is this position out of boundary?
Y
Boundary handling from Eq. (15)
N
No_reel_group = no_reel_group +1
N
No_reel_group +1 > opt_no_reel_group
Y
Is this assignment feasible?
N
Y
Calculate A from Eq. (3)
Calculate B from Eq. (4)
Calculate C from Eq. (5)
Calculate E from Eq. (7)
N
Is the reel group empty?
Y
Iteration = iteration +1
Is the pick-up-place finished?
Y
Optimization update
N
Calculate F from Eq. (8)
N
Does the nozzle need to change?
Y
Calculate D from Eq. (6)
Calculate G from Eq. (9) and H

Figure 2: The flowchart of DPSO.

the number of iteration in observation, and $\overline{Y}$ is the sample mean of the observed set $\{y_i\}$, the traveling time at iteration i.

From Table 1, it is easy to see that the DPSO outperforms the standard PSO in every case; particularly, as the number of components increased, the difference in performance is more significant. Moreover, with the reversed modification of boundary handling in (15), the CPU time needed to search the optimal solution through DPSO is a little shorter than the time taken by the standard PSO, as shown in Table 2.

It shows that DPSO outperforms the standard PSO on the ability to find the optimal solutions, and the performance improvement is shown in Table 2. From the significance

t-paired test, we also can see that the performance of DPSO is better than the performance of standard PSO in PCB component assignment/sequencing problems.

5. CONCLUSION

In this paper, we proposed a DPSO approach to solve the problem of minimizing the PCB assembly time and simultaneously optimizing assignment/sequencing problems for multiheaded SMM. We decomposed the DPSO approach into three phases: heads assignment algorithm, reel grouping optimization, and the assignment of component placement. These results lead to minimize the total assembly time of assignment/sequencing time of the placement of component on PCB board. From the experiment results, the DPSO outperforms the standard PSO in the quality of the solution found and in the time taken to search for the optimal solution.

In this paper, we only discuss the application of DPSO in a SMM. But, it is easy to modify the DPSO approach for different applications including the consideration of component placement for multiple printed circuit boards operation simultaneously and with time limitation on operations. In addition, the different parameter selection for the DPSO and different velocity updating equation selection applied in DPSO can be compared and considered.

APPENDIX

NOMENCLATURE

$V_{\mathrm{id}}^{\mathrm{old}}$:	the individual particle previous velocity.
$V_{\mathrm{id}}^{\mathrm{new}}$:	the updating velocity of individual particle in next movement.
$V_{\max}$:	the limitation velocity updating is calculated by half of the number of slots.
N_{ij}:	the nozzle of type j assigned to head i in the head-nozzle relative matrix.
$X_{\mathrm{id}}^{\mathrm{old}}$:	the previous position of individual particle.
$X_{\mathrm{id}}^{\mathrm{new}}$:	the updating position of individual particle.
$X_{\mathrm{id}}^{\mathrm{nozzle}}$:	the position of nozzle.
$X_{\mathrm{id}}^{\mathrm{reel}}$:	the position of reel group.
X_{id}:	the current position of individual particle.
P_{id}:	the best-so-far position of individual particle.
P_{gd}:	the best-so-far position of neighbors.
no_change:	the number of nozzle changes in need.
no_component:	the number of components used in board.
Average_loading:	the average of component loading at each head is calculated by dividing the number of components by the number of heads.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable suggestions and advices to improve the quality of the paper.

REFERENCES

[1] M. O. Ball and M. J. Magazine, "Sequencing of insertions in printed circuit board assembly," *Operations Research*, vol. 36, no. 2, pp. 192–201, 1988.

[2] J. F. Bard, R. W. Clayton, and T. A. Feo, "Machine setup and component placement in printed circuit board assembly," *International Journal of Flexible Manufacturing Systems*, vol. 6, no. 1, pp. 5–31, 1994.

[3] M. Ayob and G. Kendall, "A new dynamic point specification approach to optimise surface mount placement machine in printed circuit board assembly," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT '02)*, vol. 1, pp. 486–491, Bangkok, Thailand, December 2002.

[4] P. J. Egbelu, C.-T. Wu, and R. Pilgaonkar, "Robotic assembly of printed circuit boards with component feeder location consideration," *Production Planning & Control*, vol. 7, no. 2, pp. 162–175, 1996.

[5] R. H. Ahmadi and J. W. Mamer, "Routing heuristics for automated pick and place machines," *European Journal of Operational Research*, vol. 117, no. 3, pp. 533–552, 1999.

[6] W. Ho and P. Ji, "Hight a hybrid genetic algorithm for component sequencing and feeder arrangement," *Journal of Intelligent Manufacturing*, vol. 15, no. 3, pp. 307–315, 2004.

[7] E. Duman and I. Or, "Precedence constrained TSP arising in printed circuit board assembly," *International Journal of Production Research*, vol. 42, no. 1, pp. 67–78, 2004.

[8] M. Grunow, H.-O. Günther, M. Schleusener, and I. O. Yilmaz, "Operations planning for collect-and-place machines in PCB assembly," *Computers & Industrial Engineering*, vol. 47, no. 4, pp. 409–429, 2004.

[9] Y. Crama, J. van de Klundert, and F. C. R. Spieksma, "Production planning problems in printed circuit board assembly," *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 339–361, 2002.

[10] W. Lee, S. Lee, B. Lee, and Y. Lee, "A genetic optimization approach to operation of a multi-head surface mounting machine," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, vol. E83-A, no. 9, pp. 1748–1756, 2000.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, IEEE Service Center, Perth, WA, Australia, November-December 1995.

[12] H. Ucar and M. F. Tasgetiren, "A particle swarm optimization algorithm for permutation flow shop sequencing problem with the number of tardy jobs criterion," in *Proceedings of the 5th International Symposium on Intelligent Manufacturing Systems (IMS '06)*, Sakarya, Turkey, May 2006.

[13] Q. Shen, J.-H. Jiang, C.-X. Jiao, G.-L. Shen, and R.-Q. Yu, "Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists," *European Journal of Pharmaceutical Sciences*, vol. 22, no. 2-3, pp. 145–152, 2004.

[14] M. F. Tasgetiren, Y. Liang, and M. Sevkli, "Particle swarm optimization and differential evolution algorithms for single

machine total weighted tardiness problem," *Annals of Operations Research*, 2004.

[15] G. C. Onwubolu, "TRIBES application to the flow shop scheduling problem," in *New Optimization Techniques in Engineering*, pp. 517–536, Springer, New York, NY, USA, 2004.

[16] M. Clerc, "Discrete particle swarm optimization, illustrated by the traveling salesman problem," in *New Optimization Techniques in Engineering*, pp. 219–239, Springer, New York, NY, USA, 2004.

[17] W.-J. Zhang, X.-F. Xie, and D.-C. Bi, "Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 2, pp. 2307–2311, Portland, Ore, USA, June 2004.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 248798, 13 pages
doi:10.1155/2008/248798

Research Article

Multidisciplinary Optimization of Aerocapture Maneuvers

Roberto Armellin and Michèle Lavagna

Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy

Correspondence should be addressed to Roberto Armellin, armellin@aero.polimi.it

Received 15 July 2007; Accepted 13 December 2007

Recommended by Alex Freitas

A multidisciplinary-multiobjective optimization of aerocapture maneuvers is presented. The proposed approach allows a detailed analysis of the coupling among vehicle's shape, trajectory control, and thermal protection system design. A set of simplified models are developed to address this analysis and a multiobjective particle swarm optimizer is adopted to obtain the set of Pareto optimal solutions. In order to deal with an unconstrained multiobjective optimization, a two-point boundary value problem is formulated to implicitly satisfy the constraints on the atmospheric exit conditions. The trajectories of the most promising solutions are further optimized in a more refined dynamical system by solving an optimal control problem using a direct multiple shooting transcription method. Furthermore, a more complete vehicle control is considered. All the simulations presented consider an aerocapture at Mars with a polar orbit of 200 km of altitude as target orbit.

1. INTRODUCTION

Aerocapture is a state-of-the-art technology considered to reduce the cost of planetary exploration. This technique, firstly proposed by Cruz in 1979 [1], allows the reduction of fuel cost for planetary insertion by using atmospheric drag to decrease the total orbital energy of the vehicle. The aerocapture is designed to aerodynamically decelerate a spacecraft from hyperbolic approach to a captured orbit within a single pass through the atmosphere with no propulsion exploitation. Once the vehicle enters the atmosphere, bank angle modulation is used to safely remain within the flight corridor, preventing skip-out or planetary impact. Propulsion is used for attitude control and periapsis raise only.

Several missions—such as Magellan and Mars Global Surveyor—have already employed aerobraking strategy: multiple atmospheric passes over an extended period of time allow to gradually get the desired orbit; but in these cases an impulsive maneuver is first required to make the target planet capture the spacecraft. Future missions, like either robotic and human missions to Mars or the Titan Explorer, are considering using a lifting body to perform an aerocapture maneuver at the arrival planet: a lifting body is less sensitive to variation in the entry angle and the drag is easily modulated. As counterpart, sophisticated guidance algorithms are required to successfully drive the vehicle in the atmospheric path.

Many studies in the last thirty years focused on the aerocapture maneuver optimization. This optimization was mainly conducted in terms of trajectory: the shape and the aerodynamic characteristics were fixed. Within this framework many techniques were developed to optimize an aerocapture maneuver focusing on different control variables and on the minimization of the path constraints [2–6].

Bearing in mind space utilization and exploration, the payload mass delivery's capability is an open issue that leads to a wider analysis of aeroassisted maneuvers. The shape definition could be considered as an additional degree of freedom to enhance the overall maneuver efficiency. In this frame, parametric studies analyzing the influence of different shapes on the considered maneuvers were accomplished [7, 8].

The influence of variation of shape on this class of problems shows the high sensitivity the aeroassisted maneuvers have, and confirms the need of a multidisciplinary approach. In this context, the work of Sudmeijer and Mooij underlined the relevance of the shape optimization process to improve the performance of reentry probes [9].

A multidisciplinary-multiobjective approach for the coupled vehicle's shape and trajectory optimization of aerogravity assist maneuvers has been recently proposed by the authors [10]. The coupling among vehicle's shape, trajectory control, and heating rates has been exploited and a tool useful for the preliminary design of aerogravity assist maneuvers has been developed.

A similar architecture is here proposed to address the preliminary design of aerocapture maneuvers from a multidisciplinary-multiobjective standpoint. It is important to note that the aerocapture modeling and the optimization problem setup is completely different form the aerogravity assist one. In fact, a capsule-like vehicle is considered, a different control strategy is employed (exploiting both the angle of attach and the bank angle), and a different set of ordinary differential equations is carried out for the simplified dynamical model. Furthermore, the optimization process is aimed in this case to minimize the thermal protection system mass and to maximize the volumetric efficiency of the capsule, thus optimizing the payload mass delivery capability. Moreover, the aerocapture optimization problem is more challenging, as it involves the solution of a boundary value problem within each objective function evaluation. This expedient is necessary both to avoid equality constraints and to bound the propellant required for the periapsis raise.

A multiobjective particle swarm optimizer (MOPSO) is applied to detect the set of Pareto optimal solutions of the analyzed problem. The particle swarm optimization (PSO) method has been introduced for the first time by Kennedy and Eberhart in 1995 [11, 12]. Since its introduction several works have been carried out on PSO improving the original method and proving its efficiency; an overview of the more important ones is given by Poli et al. in [13]. In recent years there have been several proposals to extend particle swarm optimization to multiobjective optimization problems [14–18]. The algorithm implemented in this work is based on that proposed by Coello et al., the adoption of a variable inertia and the application of the preservation of feasible solution method (FSM) being the two main differences.

The paper is organized as follows. The models developed for the multidisciplinary treatment of the maneuver are presented first. In particular the configuration, the aerodynamics, the thermal protection system (TPS), and the dynamics models are described. Some considerations on the clashing requirements of maximizing the vehicle volumetric efficiency and reducing the TPS mass follow. Subsequently the guidelines of the MOPSO implementations are given. The coupled vehicle's shape and trajectory optimization is then presented and results are discussed. As a conclusion the trajectory refinement is analyzed.

2. MODELS

2.1. Capsule

The capsules considered in this work are axisimmetric vehicles whose geometry is described by means of five parameters $(r_n, r_b, r_r, \theta, \delta)$. These parameters are the vehicle nose radius, the base radius, the rear-base radius, the front cone half-angle, and the rear-cone half-angle, as shown in Figure 1.

By allowing a quite wide search space for the parameters, significantly different configurations can be obtained, from

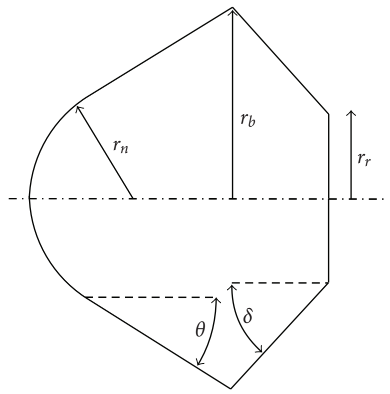


FIGURE 1: Shape parameters $(r_n, r_b, r_r, \theta, \delta)$ visualization.

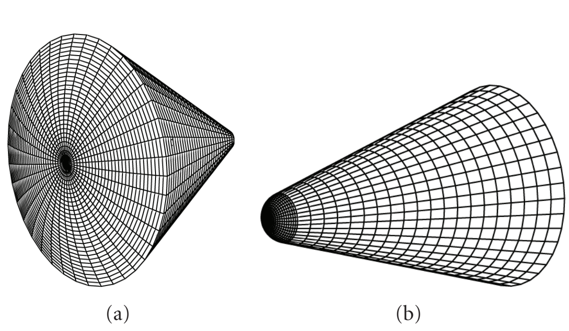


FIGURE 2: (a) Higly blunted vehicle and (b) slender vehicle.

highly blunted to slender vehicles, as reported in Figure 2. The base area $S = \pi r_b^2$ is considered as reference surface for aerodynamic coefficients. A constraint of $5\,\text{m}^3$ on capsule volume is considered. A density value $\rho_v = 230\,\text{kg/m}^3$ is considered for a volume of $3.8\,\text{m}^3$, equivalent to that of Mars Express mission, whereas a reduced density value is considered for the exceeding volume, as it serves only as structural mass.

2.2. Newtonian flow

The aerodynamic properties for the capsule-like vehicles described in the previous section are computed by summing up the contributes given by each panel in which the geometry is discretized. The Newtonian flow theory with Lee modification is applied for the estimate of the pressure coefficient, C_p [19].

The modified Newtonian flow is a local surface inclination method in which C_p depends only on the local surface deflection angle α; it does not depend on any surrounding flowfield. Within this approximation, the classical expression for the pressure coefficient can be derived as

$$C_p = C_{p_{\max}} \sin^2\alpha, \tag{1}$$

where the maximum value of the pressure coefficients, $C_{p_{\max}}$, is evaluated at the stagnation point behind a normal shock wave by

$$C_{p_{\max}} = \frac{p_0 - p}{(1/2)\rho v^2}. \tag{2}$$

In the former expression p, v, and ρ are the asymptotic flow pressure, velocity, and density, respectively; p_0 indicates the stagnation point pressure. As the free stream collides only against the frontal area of a body and it cannot curl around the body and collide against the back surface, the shadow part of the body is characterized by free-stream pressure and therefore $C_p = 0$.

In order to encompass high temperature effects, the value of p_0 is computed with the NASA Chemical Equilibrium with Application code [20]. As a result the computed stagnation pressure takes into account of the high temperature effects that characterize hypersonic flows.

The aerodynamic code has been validated on the Viking lander capsule hypersonic aerodynamics data from on-ground and on-board measurements, which have been recently used by Edquist for comparison to the LAURA Navier-Stokes code [21]. A high accuracy, average errors lower than 6%, is achieved for both drag and lift-to-drag coefficients. The obtained accuracy is remarkably high and comparable to that attained through the sophisticated CFD code LAURA.

2.3. Thermal protection system

As aerodynamic efficiency is not a major issue for aerocapture application, and the heating rate experienced are comparable to previous Mars missions, ablative thermal protection systems are the natural selection for this kind of maneuvers. In fact, ablative materials can accommodate heating rates and heat loads through phase change and mass loss; this represents the classical approach to TPS used for over 40 years in a broad range of applications, and all NASA planetary entry probes (to date) have used it [22].

Therefore, the main issue for the aerocapture maneuver stays in lowering the TPS mass fraction. In fact the mass saving gained by reducing the propellant required for the achievement of the final orbit must not be jeopardized by the need of a heavy heat shield. An analysis on past NASA missions reveals a direct connection between the TPS mass fraction and the total heat q_0 experienced by the vehicles at the stagnation point. Based on this consideration, the TPS mass fraction is simply estimated through the power law fit curve:

$$\text{TPS}\,\% = \frac{m_{\text{TPS}}}{m} = 0.0091 q_0^{0.51575}. \tag{3}$$

Neglecting the radiative heating, the total heat is calculated integrating the well-known relation for convective heating [23]:

$$\dot{q}_0 = 1.35(10^{-8})\left(\frac{\rho}{r_n}\right)^{1/2} v^{3.04}\left(1 - \frac{h_w}{h_0}\right), \tag{4}$$

in which h_w is the wall enthalpy, and h_0 is the total enthalpy.

2.4. Dynamics

Two different sets of ordinary differential equations are chosen to describe the vehicle dynamics: a simpler formulation to facilitate the solution of the coupled trajectory and vehicle's shape optimization problem, and a more complex set for trajectory further refinement.

The complete dynamical model, written in a local noninertial reference frame, reads

$$\begin{aligned}
\dot{r} &= v \sin\gamma, \\
\dot{\vartheta} &= \frac{v\cos\gamma\cos\psi}{r\cos\varphi}, \\
\dot{\varphi} &= \frac{v\cos\gamma\sin\psi}{r}, \\
\dot{v} &= \frac{D}{m} - g\sin\gamma, \\
v\dot{\gamma} &= \frac{L\cos\sigma}{m} - g\cos\gamma + \frac{v^2\cos\gamma}{r}, \\
v\dot{\psi} &= \frac{L\sin\sigma}{m\cos\gamma} - \frac{v^2\tan\varphi\cos\gamma\cos\psi}{r},
\end{aligned} \tag{5}$$

in which the state vector $(r, \vartheta, \varphi, v, \gamma, \psi)$ is made up by the orbital radius, the longitude, the latitude, the velocity, the flight path angle, and the heading angle of the spacecraft. Furthermore, $L = (1/2)\rho S C_L v^2$ and $D = (1/2)\rho S C_D v^2$ are the classical expressions for the lift and the drag force, σ is the bank angle and g the gravitational acceleration. (Consult [10] for a complete description of the dynamics.)

In the first phase of the optimization process, when the coupled shape and trajectory's optimization are considered, we are only concerned with parameters directly related to enter a closed orbit, no matter whether a specific three dimensional orbit the spacecraft will be finally placed on. For this reason, the variables of interest are r and v only. In order to analyze the behavior of these variables, the equation of flight path angle must also be taken into account. The simplified model for the aerocapture maneuver is made up by a set of three differential equations. To facilitate the analysis, the equations of motion are nondimensionalized [4]. The dimensionless arc length s replaces the time t:

$$ds = \sqrt{\frac{\beta}{r_e}}\, v\, dt. \tag{6}$$

The altitude is nondimensionalized with

$$y = \frac{\rho}{\rho_e} = e^{-\beta h}, \tag{7}$$

in which the altitude h is relative to the planet's atmosphere boundary, $h < 0$ means the vehicle is within the atmosphere. Note that a simple exponential model for the planetary density $\rho = e^{-\beta h}$ is considered, in which β is the inverse scale height of the atmosphere, whose boundary is fixed for Mars at 100 km of altitude. With the concerned speed, the following expression is used:

$$x = \ln\left(\frac{v_e}{v}\right)^2. \tag{8}$$

In the previous equations, and throughout the paper, the e subscript refers to properties at the atmosphere's boundary. The remaining required nondimensional parameters are defined by

$$\varepsilon = \sqrt{\frac{\beta}{r_e}}\frac{\rho_e S C_D}{m}, \qquad \delta = \frac{\mu/r_e}{v_e^2}, \tag{9}$$

in which μ is the gravitational parameter of the the planet. The complete nondimensional set of equations is then

$$\begin{aligned}\dot{y} &= -\sqrt{\beta r_e}\, y \sin\gamma,\\ \dot{x} &= \varepsilon y + \frac{2\delta e^x}{\sqrt{\beta r_e}}\sin\gamma,\\ \dot{\gamma} &= \frac{\varepsilon}{2}\left(\frac{C_L}{C_D}\right)\cos\sigma + \frac{\cos\gamma}{\sqrt{\beta r_e}}\left(1-\delta e^x\right).\end{aligned} \tag{10}$$

The initial values of x and y are known from the entry condition in the planetary sphere of influence, whereas the entire final state is univocally defined if the apogee of the exit trajectory is prescribed to minimize the required Δv for the circularization. In order to have the same number of differential equations and boundary conditions, two differential equations must be added to the system (10). For this purpose, the arc length is scaled as $s = T\tau$ where τ belongs to the interval $[0,1]$, and T is the unknown final arc length, which is an unknown constant. One additional differential equation is then $\dot{T} = 0$. Furthermore, if a constant bank angle control law is considered, the differential equation $\dot{\sigma} = 0$ can be used to match the number of equations and the number of boundary conditions. The constant bank angle approximation represents the simplest control law that can be adopted as far as the achievement of the tridimensional final orbit is not of concern. In this work frame, the complete set of equations in the new independent variable τ is

$$\begin{aligned}\dot{y} &= T\left(-\sqrt{\beta r_e}\, y\sin\gamma\right),\\ \dot{x} &= T\left(\varepsilon y + \frac{2\delta e^x}{\sqrt{\beta r_e}}\sin\gamma\right),\\ \dot{\gamma} &= T\left(\frac{\varepsilon}{2}\frac{C_L}{C_D}\cos\sigma + \frac{\cos\gamma}{\sqrt{\beta r_e}}\left(1-\delta e^x\right)\right),\\ \dot{T} &= 0,\\ \dot{\sigma} &= 0.\end{aligned} \tag{11}$$

This set of equations is suitable to define the two-point boundary value problem (TPBVP) described in detail in the following section.

3. VOLUMETRIC EFFICIENCY AND TPS MASS PERCENTAGE ANALYSIS

To better understand the results of the following section, the effects of the vehicle's shape on the trajectory minimum altitude, on the heat shield percentage, and on the volumetric efficiency are analyzed firstly. The simplified model of the dynamic (11) is considered.

The selected control law imposes both bank angle σ and the angle of attack α to be constant; the C_L/C_D is, therefore, determined by the angle of attack α. The goal is to use the atmospheric path to reduce the Δv required to achieve a circular target orbit of 200 km of altitude. Within this frame, the atmospheric path is used to model the incoming hyperbola into an elliptical trajectory having the apogee at an altitude of 200 km, without considering the achievement of the 3D target orbit. The achieved orbital plane as the vehicle leaves the atmosphere, is assumed to be the target orbit plane: that matching condition is assured by a proper choice of the entry plane in the planet's atmosphere (i.e., the proper selection of the pericenter radius of the incoming hyperbola). The satisfaction of this constraint is addressed within the solution refinement process. The atmospheric path is computed by solving the TPBVP after the vehicle aerodynamic coefficients, the volume, and the mass have been computed. Some details of the TPBVP formulation are now illustrated.

The conditions at the edge of the atmosphere are labeled with the subscript e, those at Mars sphere of influence by ∞, and the entry and exit conditions with the superscripts $-$ and $+$, respectively. The atmospheric boundary is settled at 100 km over Mars mean radius. The 2-body problem and (11) completely describes the system dynamics outside and inside the atmosphere respectively [24]. The velocity magnitude at the boundary of the sphere of influence is assigned to $v_\infty^- = 5$ km/s. At the atmospheric entry, the initial radius r_e^- is r_e and the entry velocity v_e^- is computed by the solution of the Vis-Viva equation:

$$\frac{1}{2}v_e^{-2} - \frac{\mu}{r_e^-} = \frac{1}{2}v_\infty^{-2}. \tag{12}$$

In order to apply a tangential Δv correction, which represents the minimum propellant strategy, the apogee r_a of the trajectory obtained after the atmospheric phase is constrained to lie on the target orbit. Thus, the values of γ_e^+ and Δv required for the circularization can be computed as a function of the final velocity v_e^+ through the following procedure.

(1) Evaluate the energy of the exit trajectory:

$$E = \frac{1}{2}v_e^{+2} - \frac{\mu}{r_e^+}. \tag{13}$$

(2) Compute the semimajor axis and the eccentricity:

$$a = -\frac{\mu}{2E}, \qquad e = \frac{r_a}{a} - 1. \tag{14}$$

(3) Obtain the angular momentum:

$$\Gamma = \sqrt{a(1-e^2)\mu}. \tag{15}$$

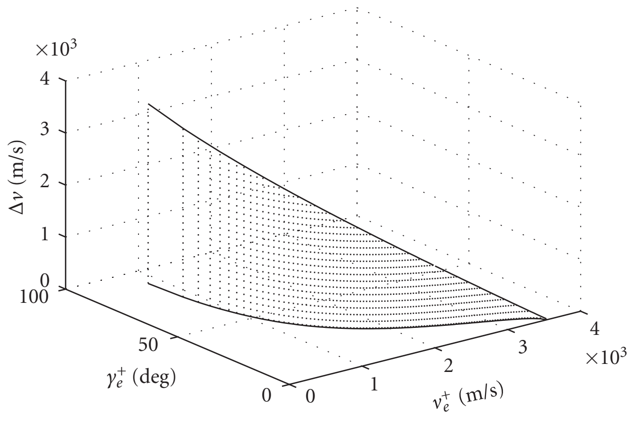


FIGURE 3: The v_e^+-γ_e^+ constraint and the associated Δv.

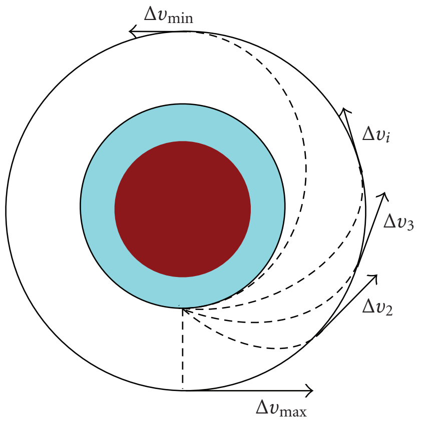


FIGURE 4: Sketch of the periapsis raise Δv.

(4) Finally, compute the exit flight path angle and the Δv burn:

$$\gamma_e^+ = acos\left(\frac{\Gamma}{v_e^+ r_e^+}\right),$$
$$\Delta v = \sqrt{\frac{\mu}{r_a}} - \sqrt{-\frac{\mu}{a} + 2\frac{\mu}{r_a}}. \quad (16)$$

The procedure admits solutions of Figure 3 for values of v_e^+ in the range [825.9, 3527.4] m/s.

A subset of exit trajectories with the apogee on the target orbit are sketched in Figure 4. The $\Delta v_{\max}$ corresponds to the unrealistic vertical exit from the atmosphere. In that case the spacecraft would arrive at the apogee at zero velocity, resulting in a $\Delta v = 3453.7$ m/s: the circular target orbit velocity. The minimum Δv correction is 24.4 m/s and it is associated to the maximum exit velocity $v_e^+ = 3527.4$ m/s and to the minimum exit flight path angle $\gamma_e^+ = 0$ deg. The exit trajectory can be identified with an Hohmann transfer connecting the circular orbit at the atmosphere boundary to the target orbit, with the main difference being only the apogee impulse to be required.

To facilitate the TPBVP numerical solution, a 3500 m/s constraint on v_e^+ is posed; the resultant γ_e^+ and Δv are 1.73 deg and 52.6 m/s, respectively. It's worth noting that the computed trajectories are not optimal from the Δv standpoint, although the selected constraints assure them to be close to the theoretical minimum (considering the required additional propellant). The proposed procedure assures a complete knowledge of the final state x, y, and γ; the remaining unknowns, that is, the total arc length T, the entry flight path angle γ_e^- and the bank angle σ, are computed by the TPBVP solver. A linear multipoint method [25–27] is applied to solve the problem and the second order solution derived by Vihn et al. [4] is used as first guess solution. The algorithm shows quadratic convergence typical of Newton's method and converges on average within 10 iterations. The integration interval is transcribed using 100 nodes to assure absolute and relative accuracy of 10^{-8}.

The Δv correction is univocally determined once the constraints on the final conditions are satisfied, therefore no Δv optimization is required. As the Δv minimization is substituted by a constraint satisfaction problem, the attention can be focused on two other important aspects of the aerocapture design: the vehicle volumetric efficiency maximization and the TPS mass fraction minimization. A high volumetric efficiency has two main benefits: a smaller launcher fairing is required as the vehicle is more bulky, and the vehicle's volume can be better exploited for the spacecraft accommodation. The minimization of the TPS mass ratio leads to a maximization of the mission outcome, as the percentage of payload mass increases. It needs to be remarked that the TPS mass must be lower than the propellant required to achieve the target trajectory without using the aerocapture in order to consider the aerocapture beneficial. These goals are monitored by the TPS mass fraction of (3) and

$$\eta = \frac{l_{\max}}{V}, \quad (17)$$

where $l_{\max}$ is the maximum linear dimension of the capsule. The main difference between these two indexes is that η is simply a geometric value, whereas TPS% depends on the vehicle's geometry and on the atmospheric trajectory.

It is now shown that these indexes have clashing behaviors, thus justifying the multiobjective optimization approach described in the following section. The vehicles of Figures 5 and 6 are considered to clarify this concept: the first one is more compact and it is characterized by $\eta = 0.59\,\text{m}^{-2}$, the second has a bigger base area and being shorter, resulting in $\eta = 0.71\,\text{m}^{-2}$.Both the shapes have a volume of $5\,\text{m}^3$, and the same trim angle of attack $\alpha = -20$ deg. The bank angle that guarantees the satisfaction of the constraint on the exit conditions is $\sigma = 163.05$ deg for the first shape and $\sigma = 115.7$ deg for the second. Figures 7 and 8 describe the atmospheric path and the velocity history. Shape 1 requires a steeper entry, a longer maneuver, and a lower altitude in order to achieve the required kinetic energy loss. As a result, the vehicle experiences a greater value of total heat load, therefore a greater TPS mass is needed. More specifically, the TPS mass percentage are 9.48 and 7.44 for Shape 1 and Shape 2, respectively. In general, for a given mass, a more compact vehicle (i.e., high volumetric efficiency) must fly deeper into the atmosphere to lose its kinetic energy, thus experiencing

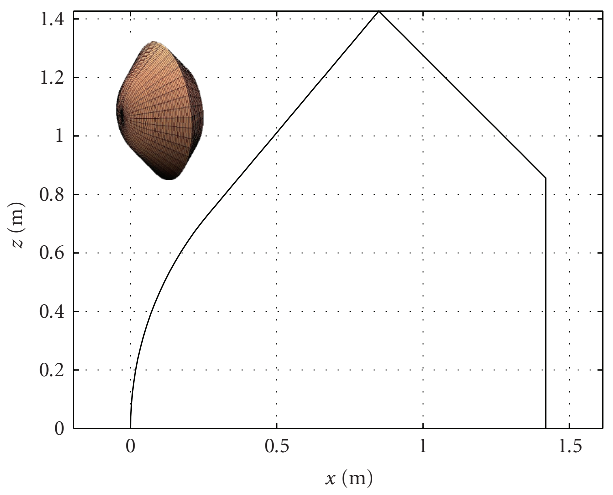


Figure 5: Shape 1, volumetric efficiency oriented vehicle.

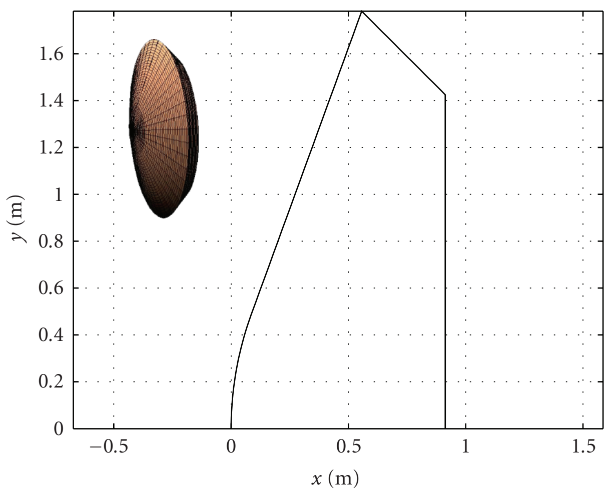


Figure 6: Shape 2, TPS% oriented vehicle.

high thermal loads (i.e., high TPS mass). It should be underlined that within the framework of constant vehicle volume and density, the performance index η plays a role similar to that of the ballistic coefficient, defined as $m/(SC_D)$, and frequently used in literature. A high volumetric efficiency translates into a high ballistic coefficient (see Table 1), thus a great value of atmosphere density is required to produce a significant amount of drag. Furthermore, for a fixed shape, the ballistic coefficient almost linearly increases with the vehicle's dimension. That is the reason why special devices like parachutes or ballutes are considered when the aerocapture maneuver for human missions to Mars is studied.

4. MULTIDISCIPLINARY OPTIMIZATION

4.1. *Multiobjective particle swarm optimizer*

As shown in the previous section, it is difficult to identify a single objective function when designing an aerocapture maneuver. In order to use classic optimization codes (i.e., gradient based methods) to solve a multiobjective optimization problem a common practise is to merge the different objective functions into a single scalar objective function by means of weighting factors. This technique requires an accurate selection of the weights, and it has, as major drawback, the identification of a single optimal solution per run. On the contrary, population-based optimizers can be more easily modified to deal with a vector of objective functions delivering the entire set of Pareto optimal solutions. Furthermore, particle swarm optimization seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization [28]. In a multiobjective optimization problem, the objective function is an M-dimensional vector:

$$\mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x})\big). \tag{18}$$

In this frame, a criterion to compare vectors is necessary to identify the optimal solution set. The Pareto dominance is the appropriate criterion to serve this aim, enabling the solutions ranking [29].

The MOPSO implemented for the solution of the problem at hand is based on the following algorithmic flow.

(1) Randomly initialize a number of individuals or particles N within the design space.

(2) Evaluate the objective function:

$$\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i) \quad \text{for } i = 1, \dots, N. \tag{19}$$

(3) Update the personal best solution $\mathbf{p}_{\text{best}}$. The solutions are compared using the Pareto dominance criterion. Thus, for each particle we have

$$\mathbf{p}_{\text{best}} = \begin{cases} \mathbf{x}_i, & \text{if } \mathbf{x}_i \text{ dominates } \mathbf{p}_{\text{best}}, \\ \mathbf{p}_{\text{best}}, & \text{if } \mathbf{p}_{\text{best}} \text{ dominates } \mathbf{x}_i, \\ \mathbf{x}_i \text{ or } \mathbf{p}_{\text{best}}, & \text{randomly in the other cases.} \end{cases} \qquad \text{for } i = 1, \dots, N, \tag{20}$$

(4) Update global best list $\mathbf{G}_{\text{best}}$. In the multiobjective problem, $\mathbf{G}_{\text{best}}$ is the analogous of the scalar global best $\mathbf{g}_{\text{best}}$ and it represents the entire set of nondominated solutions. This list is updated by processing the subset of nondominated solutions $\mathbf{x}_j$ with $j = 1, \dots, N^* \le N$.

  (i) If $\mathbf{x}_j$ is dominated by one of the solution belonging to the list, do not updated the list.

  (ii) If $\mathbf{x}_j$ dominates one or more solutions belonging to the list, then add $\mathbf{x}_j$ to the $\mathbf{G}_{\text{best}}$ list and delete the dominated solutions.

  (iii) If $\mathbf{x}_j$ neither dominates nor is dominated by any solution belonging to the $\mathbf{G}_{\text{best}}$ list, then simply add $\mathbf{x}_j$ to the list.

(5) Update the global best solution $\mathbf{g}_{\text{best}}$. Note that the $\mathbf{g}_{\text{best}}$ is univocally defined for a scalar objective function, whereas it must be opportunely chosen within

TABLE 1: Numerical results for the two analyzed shapes.

Parameter		Shape 1	Shape 2
TPS%	—	9.48	7.44
η	m^{-2}	0.56	0.71
α	deg	−20	−20
σ	deg	163.18	115.7
C_L/C_D	—	0.15	0.31
$m_0/(SC_D)$	kg/m^2	135.99	69.62
$h_{\min}$	km	43.01	48.6
γ_e^-	deg	−8.57	−8.11

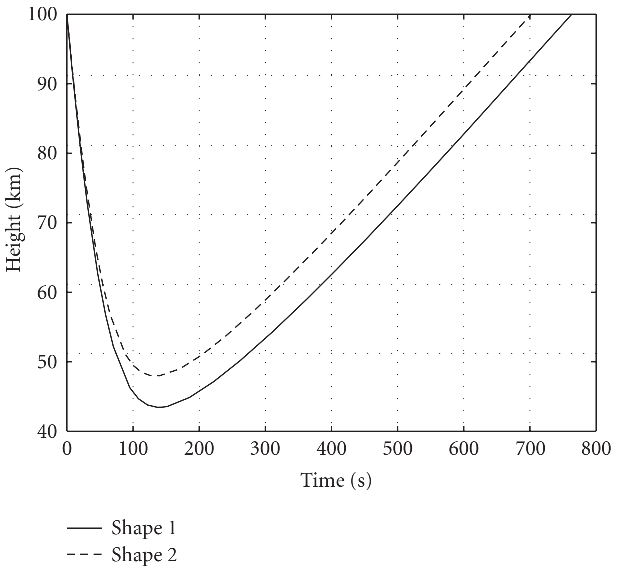


FIGURE 7: Shape 1 and Shape 2 altitude profile comparison.

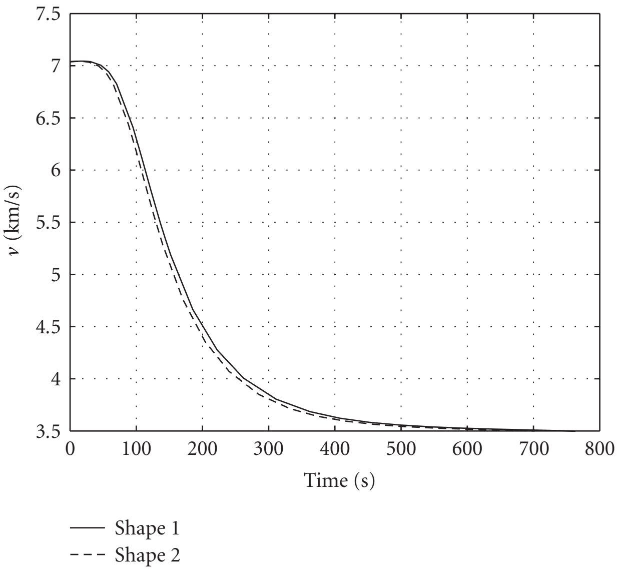


FIGURE 8: Shape 1 and Shape 2 velocity comparison.

the $\mathbf{G}_{\text{best}}$ list in the multiobjective case. The selection of the $\mathbf{g}_{\text{best}}$ plays a key role in obtaining a uniform set of Pareto optimal solutions. For this purpose, a uniform 30-cell grid in the objective space is defined at each iteration and the number of solutions belonging to each grid cell is calculated. Based on this number, a roulette-wheel method is then applied to promote the selection of $\mathbf{g}_{\text{best}}$ in a low-populated grid cell.

(6) Compute the new particles position by

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1}\Delta t \quad \text{for } i = 1,\ldots,N, \tag{21}$$

in which $\mathbf{v}_i^{k+1}$ is the velocity of the ith particle at the $(k+1)$ iteration, given by

$$\mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + c_1 r_1 \frac{\mathbf{x}_i^k - \mathbf{p}_{\text{best}}}{\Delta t} + c_2 r_2 \frac{\mathbf{x}_i^k - \mathbf{g}_{\text{best}}}{\Delta t}. \tag{22}$$

(7) Repeat (2)–(6) until the convergence criterion is satisfied or the maximum number of iterations is reached.

The parameters c_1 and c_2 of (22) are considered constant and equal to 2 during the optimization, assuring a balance between local and global terms. A linear decrease of w with the iteration number in the interval $[0.4, 1.4]$ is adopted. In particular a greater value of the inertia enables a better exploration of the search domain in the first phase of the optimization, whereas a lower value allows a better analysis of the most promising areas of research space in the subsequent phases. Note that if the position of a particle goes outside the search space, the violated component of the decision vector takes the value of the corresponding boundary and its velocity component is multiplied by a random number between $[-1, 0]$.

The maximum numbers of particle belonging to the $\mathbf{G}_{\text{best}}$ is fixed to 100 units. The same procedure adopted for selecting the $\mathbf{g}_{\text{best}}$ is used to delete those solutions belonging to a highly populated grid-cell, if the maximum list size is exceeded.

The problem addressed with implemented MOPSO is characterized by the presence of inequality constraints necessary to guarantee a minimum aerodynamic performance and vehicle's length. As the feasible domain inside the search space is sufficiently large the FSM is adopted for the constraints handling [30]. More specifically the swarm initialization is performed randomly, but only feasible solutions are retained. This implies that the first step of the algorithm generally requires the evaluation of a number of solutions greater than the population size. Furthermore, only feasible solutions are counted for the $\mathbf{g}_{\text{best}}$ and $\mathbf{p}_{\text{best}}$ values during the optimization. The initial velocity of the particle is set to be 0.

The convergence criterion adopted is based on the comparison of the average position of the non dominated solutions in the objective space with the same average position of the previous 20 iterations. If the componentwise difference of this two vectors is lower than 1% the Pareto set of optimal solutions is assumed to have been found. Furthermore, a maximum number of iterations of 100 and a 20-particle swarm are considered. These values are chosen, on the basis of several experiments, to assure an acceptable repeatability of the Pareto optimal solution set with a limited computational time.

4.2. Optimization architecture

Since the minimization of η and TPS% are two clashing requirements, a multiobjective optimization architecture is adopted. The overall optimization architecture is then given, the scheme of Figure 9 can be used as a visual aid. The optimization variables are the four geometric parameters $r_n \in [0.1, 0.9], r_r \in [0.2, 0.9], \theta \in [20, 70]$ deg, $\delta \in [5, 60]$ deg, and the angle of attack $\alpha \in [-30, 30]$ deg. Note that the vehicle nose radius and the rear radius are expressed as fraction of the base radius, whose value is computed to satisfy the constraint of $5\,\text{m}^3$ for the capsule's volume. The sphere of influence entry velocity $v_\infty^- = 5$ km/s and the vehicle's density $\rho_v = 230\,\text{kg/m}^3$ are user provided constants necessary to define the optimization problem.

For each set of optimization variables the vehicle's aerodynamic performance at the trim angle of attack are computed. At this point the TPBVP can be formulated and solved, delivering the total heat load experienced by the vehicle. The γ_e^+ is constrained to 1.73 deg as in Section 3, resulting in the same Δv of 52.6 m/s for all the solutions. The two performance indexes are then simply evaluated and the optimizer iterates until the convergence criterion is satisfied or the maximum iteration number reached. Two simple inequality constraints are also considered on the lift-to-drag ratio, $|L/D| \geq 0.3$, and vehicle length, $l \geq 0.8$ m. The fist one is necessary to provide the vehicle with a means to accurately control the atmospheric path, the second removes solutions with extremely short length from the search space. A function evaluation takes on average 1.26 s on a Intel Pentium 4, 2.53 GHz Desktop.

4.3. Experiments and results

The values of the MOPSO parameters given in Section 4.1 are the result of a tuning process based on several experiments in which the swarm size, the maximum number of iterations, and the value of the inertia have been changed. The average behavior of the algorithm is shown in Figure 10 with a plot of 30 simulations in which the solutions belonging to the $\mathbf{G}_{\text{best}}$ list are interpolated by means of cubic splines to avoid cluttering the plot. The result is satisfactory from the preliminary design point of view. The algorithm is very effective in computing the Pareto optimal solutions for low TPS mass vehicles; on the other hand suboptimal solutions appear in the high volumetric efficiency region. This behavior is mainly due to the difficulty in finding feasible solutions of the TPBVP when compact capsules are considered. In these cases, is not always possible to decelerate the vehicle's as required to satisfy the constraints imposed in the TPBVP formulation. Furthermore, note that the poor behavior in the flat part of the Pareto front is due more to the interpolation process than to the actual values of the solutions. Only 5 simulations out of 30 stop for the satisfaction of the convergence criteria, whereas the reamaining ones reach the maximum number of iterations. Nevertheless, it has been noticed that increasing the iterations does not significantly improve the quality of the solution, whereas augmenting the computational time. These considerations underline the difficulty of defining an appropiate convergence criteria in multiobjective optimization.

The $\mathbf{x}$ in Figure 11 shows the mean values of the two objective functions for each of the Pareto set analyzed before. It is worth noting that the mean values are all within a range of 5%. Their values also show that, as already pointed out, the left branch of the Pareto front is more densely sampled than the right one. The □ and • describe a typical behavior of the algortithm when the constant value $w = 0.4$ is employed for the iniertia. In these cases the algorithm tend to locally converge to one of the two branches of the Pareto set, as clearly enlighten in Figure 12.

In Figure 13 the $\mathbf{G}_{\text{best}}$ list after 100 iterations of one of the 30 simulations performed is plotted. The Pareto front confirms the considerations reported in Section 3: the volumetric efficiency oriented solutions (bottom right) are compact capsules with higher value of TPS mass; as opposite the TPS oriented solutions tend to have high base radius and to be very short.

Table 2 summarizes the numerical data of the five solutions highlighted in the previous figure. All the solutions tend to the minimum value allowed for the lift-to-drag ratio in order to decelerate the capsule at higher altitudes, thus reducing the TPS mass ratio. Soln 2 to Soln 5 are characterized by almost the same value of the front cone half-angle which allows to consider large nose radii. Soln 1 belongs to a different class of vehicles: the front cone half-angle is the lowest allowed, and the forebody is longer than the aftbody. The difference between these two classes of vehicles is highlighted also by the different sign of the lift-to-drag ratio: Soln 1 is the only shape that produces negative C_L for negative angles of attack. Eventually, note that TPS oriented solutions have higher values of minimum altitude, shallower entry angles, and lower ballistic coefficient.

The maximum of TPS% is 13.30, which results in 126 kg of TPS mass in the worse case solution. In order to establish the aerocapture effectiveness, the TPS mass must be compared with the propellant required for the circularization without employing aerocapture. This value is computed in a 2-body approximation and the spacecraft is assumed to

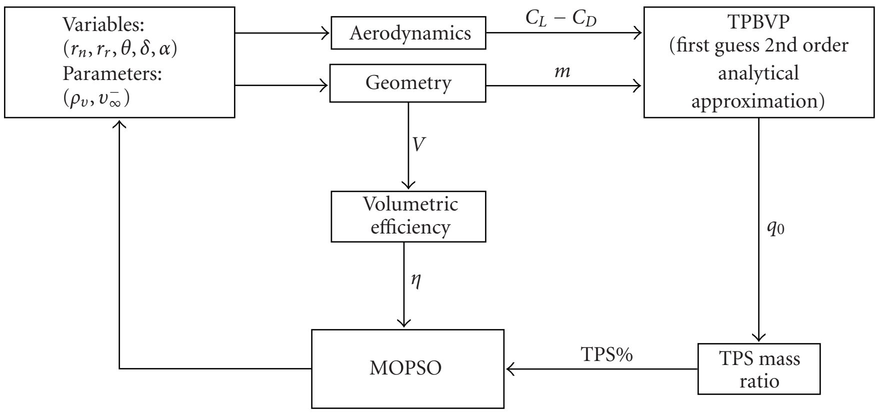


FIGURE 9: Aerocapture optimization scheme.

TABLE 2: Numerical results for five solutions belonging to the Pareto optimal set.

Parameter		Soln 1	Soln 2	Soln 3	Soln 4	Soln 5
TPS%	—	13.28	9	7.9	7.5	6.1
η	m^{-2}	0.46	0.55	0.63	0.68	0.89
α	deg	−29.45	−21.13	−19.3	−20.3	−19.27
σ	deg	57.4	116.9	117.2	115.8	115
C_L/C_D	—	−0.31	0.32	0.30	0.31	0.30
$m/(SC_D)$	kg/m^2	299.72	120.26	86.13	77.11	45.02
$h_{\min}$	km	36.5	43.9	46.7	47.7	52.3
γ_e^-	deg	−9.08	−8.50	−8.25	−8.19	−7.73
r_b	m	1.16	1.37	1.59	1.69	2.19
r_n	m	0.73	1.22	1.42	1.52	1.94
r_r	m	0.58	0.47	0.45	0.48	0.82
θ	deg	21	68.51	70	69.64	70
δ	deg	41	56.59	40	42.07	5.2

switch on the thrusters at the hyperbola pericenter, located at 200 km of altitude. The Δv required to circularize the orbit is given by

$$\Delta v = \sqrt{\frac{\mu}{r_a} + v_\infty^{-2}} - \sqrt{\frac{\mu}{r_a}} \tag{23}$$

and the propellant fraction by rocket equation

$$\frac{m_p}{m} = 1 - e^{-(\Delta v/I_{\text{sp}} g_0)}, \tag{24}$$

in which I_{sp} is the thruster specific impulse, and $g_0 = 9.81\,\text{m/s}^2$ represents the Earth gravitational acceleration at sea level. For a value of $I_{\text{sp}} = 400\,\text{s}$ it results $m_p/m = 0.594$, which means that more than the 59% of the spacecraft should be propellant, largely greater than the 13.30% of TPS. Therefore the aerocapture represents an effective means to reduce the propellant compared to a classical orbital circularization, thus significantly increasing the mass specifically devoted to the payload. In the following section, the refinement of the Soln 2 and Soln 3 is addressed, as they show a good compromise between volumetric efficiency and TPS mass ratio.

5. TRAJECTORY REFINEMENT

The dynamical model applied for the solution of the TPBVP considers the evolution of the velocity, the altitude, and the flight path angle only, and terms of order $\Delta r/r_e$ are neglected.

In order to address the problem of achieving the 3D target orbit the complete dynamical model (3) is considered. The constant bank angle approximation is lifted and an optimal control problem is formulated to optimize the bank profile. In the trajectory refinement the vehicle's shape is fixed and the constant angle of attack strategy is retained, taking advantage of the values computed in the coupled trajectory-shape optimization. Thus, all the variables considered in the former optimization are fixed parameters. The state and control vectors are $\mathbf{x} = (r, \theta, \varphi, v, \gamma, \psi)^T$ and $\mathbf{u} = \sigma$, respectively. The initial conditions are expressed by

$$\begin{aligned} r_e^- &= r_e, \\ \mathbf{v}_\infty^-(\mathbf{x}_e^-) &= \mathbf{v}_\infty^-. \end{aligned} \tag{25}$$

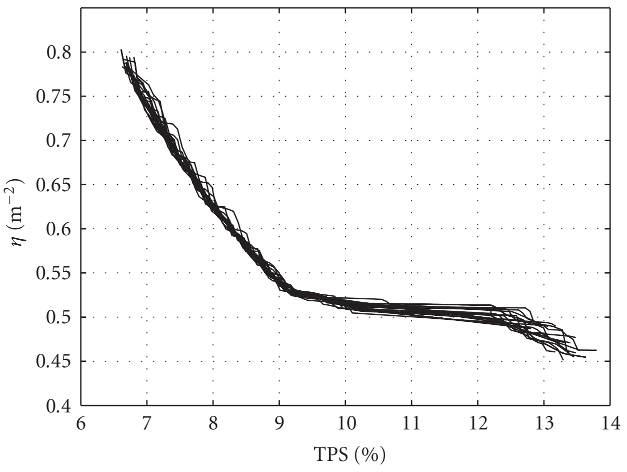


FIGURE 10: Cubic spline interpolated Pareto optimal sets.

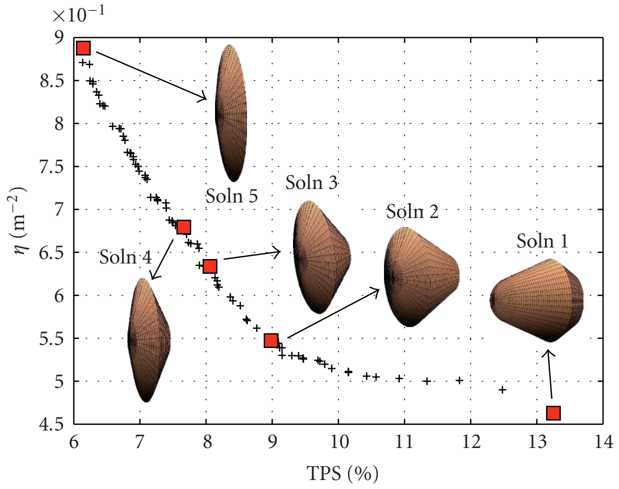


FIGURE 13: Aerocapture Pareto optimal solutions.

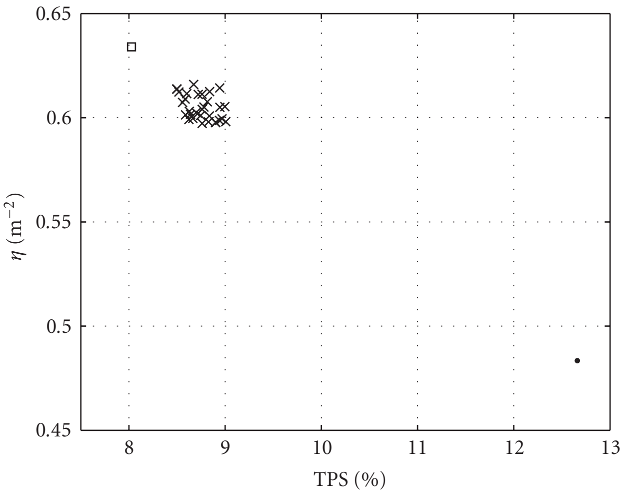


FIGURE 11: Mean values of the Pareto optimal solutions.

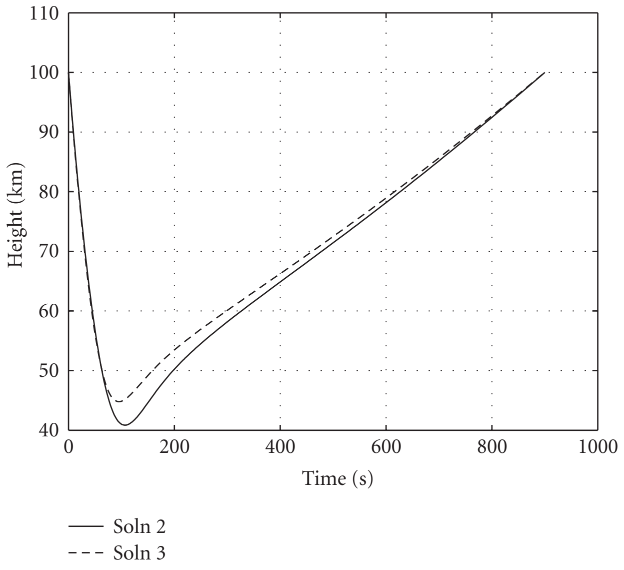


FIGURE 14: Soln 2 and Soln 3 refined solution: height profile.

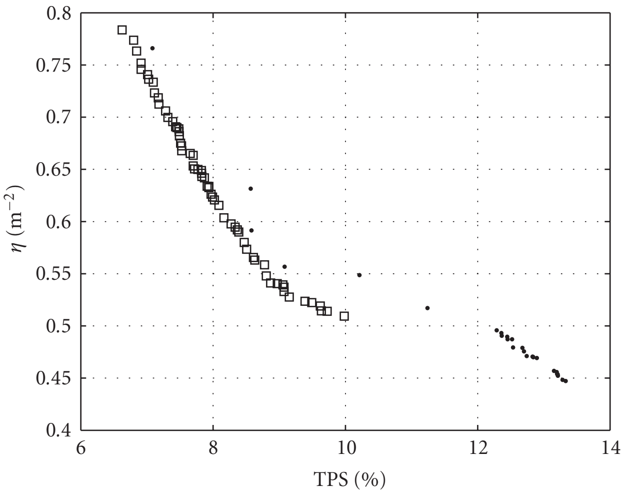


FIGURE 12: Two examples of local convergences ($w = 0.4$).

The scalar constraint assures that the maneuver begins at the planet atmosphere interface. The vectorial constraint states that the incoming velocity at the planet sphere of influence corresponding to the initial state $\mathbf{x}_e^-$, that is, $\mathbf{v}_\infty^-(\mathbf{x}_e^-)$, must be equal to the spacecraft incoming velocity $\mathbf{v}_\infty^-$, imposed by the heliocentric trajectory analysis. The optimizers exploits the two degrees of freedom on the initial state to choose the proper entry plane in the Martian atmosphere. The following three scalar constraints are enforced on the final position:

$$\begin{gathered} r_e^+ = r_e, \\ a\sin\left(\frac{\Gamma_{e,z}^+}{\Gamma_e^+}\right) = \frac{\pi}{2}, \\ -\frac{\mu}{v_e^{+2} - (2\mu/r_e^+)}\left(1 + \left\|\frac{\mathbf{v}_e^+ \times \mathbf{\Gamma}_e^+}{\mu} - \frac{\mathbf{r}_e^+}{r_e^+}\right\|\right) = r_a \end{gathered} \tag{26}$$

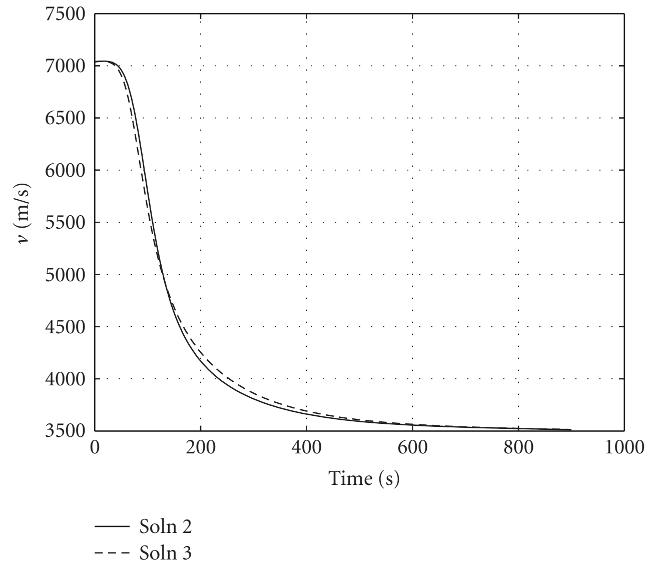


Figure 15: Soln 2 and Soln 3 refined solution: velocity magnitude.

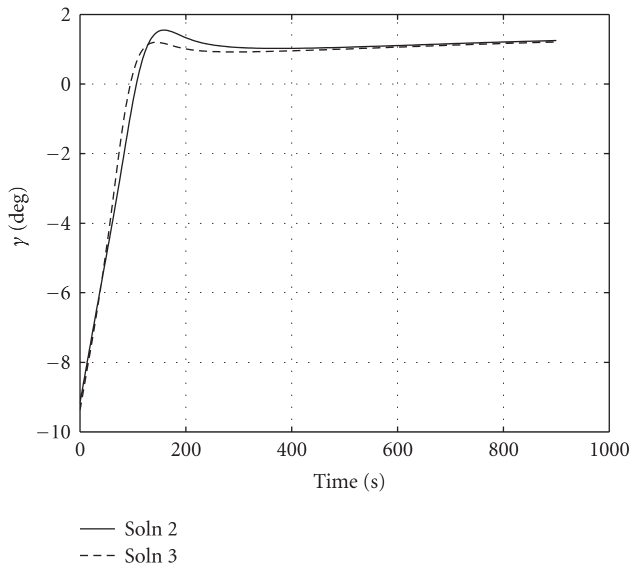


Figure 16: Soln 2 and Soln 3 refined solution: flight path angle.

The first constraint assures that the trajectory ends at the atmospheric boundary, whereas the remaining two guarantee the exit trajectory to be polar and with an apogee lying on the target orbit. If the additional constraint $\gamma_e^+ = 0$ is considered, thus achieving the theoretical minimum $\Delta v = 24.4$ m/s as explained in Section 3, the local optimizer fails to converge. More specifically, the tight relative tolerance of 10^{-8} on the constraints satisfaction is violated. This problem is avoided if the constraint on the final flight path angle is dropped and the Δv minimization is addressed. Thus the objective function is

$$J = \Delta v = \sqrt{\frac{\mu}{r_a}} - \sqrt{v_e^{+2} - 2\mu\left(\frac{1}{r_e^+} - \frac{1}{r_a}\right)}, \qquad (27)$$

in which the first term represents the target orbit velocity and the second one the vehicle's velocity at the apogee.

For each shape the TPS% is bounded by the value of the coupled shape-trajectory optimization. In order to evaluate the constraint, the stagnation heat load is added to the state vector and it is integrated along with the dynamics. The trajectory is split into four multiple shooting intervals, and the integration is performed adopting a 8th order fixed-step Runge-Kutta scheme with absolute and relative tolerances of 10^{-8}. The optimal solution is found using a sequential quadratic programming (SQP) optimizer.

The results obtained for the Soln 2 and Soln 3 are shown in Figures 14–17. For both the solutions the value of the minimum altitude is slightly lower than the one found in the previous section due to a steeper entry flight path angle. The optimizer completely changes the bank control law; the bank modulation is exploited to minimize the Δv and mainly to match the target orbital plane. Note that the maximum value of the aerodynamic forces on the trajectory is reached during the deepest phase in the atmosphere, when the bank angle is not far from the value found in the coupled shape-trajectory optimization. Furthermore, if most of the points describing

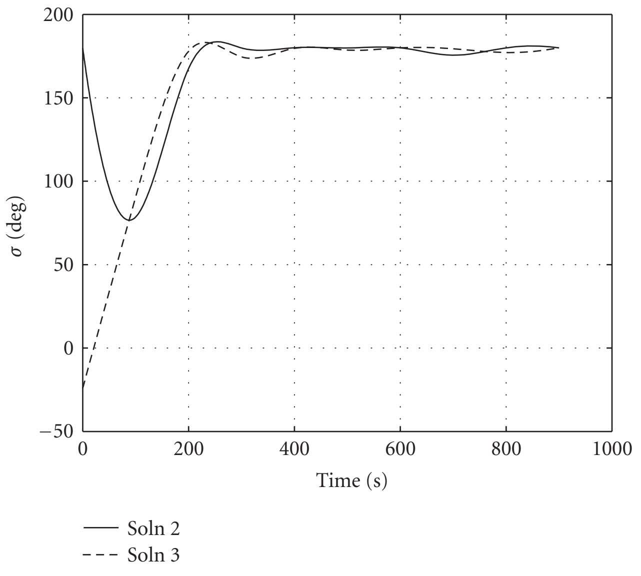


Figure 17: Soln 2 and Soln 3 refined solution: bank profile.

the optimal bank angle spline are substituted by the constant bank angle value computed in the shape-trajectory optimization, only a slight change in the trajectory is obtained. Moreover the trajectory refinement has a high convergence rate when the constant bank angle law is used as first guess solution. These considerations prove that the assumptions adopted in Section 4 deliver sufficiently accurate results.

The Δv is 39.22 m/s for Soln 2 and 38.25 m/s for Soln 3, close to the theoretical minimum of 24.4 m/s, and slightly lower than the 52.6 m/s constraint of the shape-trajectory optimization. Furthermore, the TPS mass ratio is lower compared to the one found in the shape-trajectory optimization for both of the solutions. This result is a consequence of the shorter permanence of the vehicles in the lower layers of Mars

Table 3: Aerocapture trajectory refinement: numerical results.

	$h_{\min}$ [km]	γ_e^- [deg]	TPS%	Δv [m/s]
Soln 2	40.85	−9.16	7.98	39.22
Soln 3	44.8	−9.38	7.05	38.25

atmosphere. Note that, although the trajectory refinement changes the value of the TPS mass, the trend highlighted in the previous section still holds, that is, a higher ballistic vehicle requires a higher TPS mass. The main results of the trajectory refinement are summarized in Table 3.

6. CONCLUSIONS

The use of an MOPSO for the optimization of an aerocapture maneuver at Mars from a multidisciplinary-multiobjective standpoint is presented. More specifically, the interaction among vehicle's shape, trajectory control, and TPS design is taken into account in the preliminary design of the maneuver. The aerocapture multiobjective optimization emphasizes the conflict between volumetric efficiency maximization and thermal protection system mass ratio minimization. Solutions that show a compromise between the two objective goals have shapes similar to those adopted in previous landing missions to Mars. Aerocapture maneuver is demonstrated to be a valid means for lowering the propellant required to accomplish interplanetary missions if compared to classical circularization maneuvers. The models adopted are suitable for Phase-A studies of future aerocapture missions, and the proposed method properly matches the requirements of concurrent engineering, which is nowadays the leading approach in aerospace field. The MOPSO, although of simple implementation, effectively compute the Pareto optimal solution set of a complex engineering problem using low number of function evaluations. The advantage of having a set of Pareto optimal solutions is of vital importance in aerospace field, where most of the design processes are characterized by the interaction of several subsystems and disciplines, and it is often impossible to identify a single performance index. Based on these considerations, in the authors belief, MOPSO could be successfully applied to a broad set of aerospace engineering problems, especially in system design and trajectory optimization fields.

REFERENCES

[1] M. I. Cruz, "The aerocapture vehicle mission design concept," in *Proceedings of the AIAA Joint Propulsion Conference*, 1979, AIAA papar 79-0893.

[2] M. Bello Mora and G. Dutruel-Lecohier, "Reentry and aeroassisted transfer trajectory optimal control: the gradient restoration algorithm," in *Proceedings of the 3rd International Conference on Space Guidance, Navigation, and Control Systems (ESTEC '96)*, pp. 95–101, Nooordwijk, The Netherlands, November 1996.

[3] K. D. Mease, "Optimization of aeroassisted orbit transfer: current status," *Journal of the Astronautical Sciences*, vol. 36, no. 1 part 2, pp. 7–33, 1988.

[4] N. X. Vinh, W. R. Johnson, and J. M. Longusky, "Mars aerocapture using bank modulation," in *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, Denver, Colo, USA, August 2000, collection of technical papers.

[5] E. Sigal and M. Guelman, "Optimal aerocapture with minimum total heat load," in *Proceedings of the 52nd International Astronautical Congress*, Toulouse, France, October 2001.

[6] D. Vaughan, H. C. Miller, B. Griffin, B. F. James, and M. M. Munk, "A comparative study of aerocapture missions with a Mars destination," in *Proceedings of the 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Tucson, Ariz, USA, July 2005.

[7] S. A. Whitmore, D. W. Banks, B. M. Andersen, and P. R. Jolley, "Direct—entry, aerobraking, and lifting aerocapture for human-rated lunar return vehicles," in *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exibit*, Reno, Nev, USA, January 2006.

[8] R. D. Braun and R. W. Powell, "Aerodynamic requirements of a manned aerobraking transfer vehicle," in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference*, Portland, Ore, USA, August 1990.

[9] K. J. Sudmeijer and E. Mooij, "Shape optimization for small experimental re-entry module," in *Proceedings of the 11th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, Orleans, France, September-October 2002.

[10] R. Armellin, M. Lavagna, R. P. Starkey, and M. J. Lewis, "Aerogravity-assist maneuvers: coupled trajectory and vehicle shape optimization," *Journal of Spacecraft and Rockets*, vol. 44, no. 5, pp. 1051–1059, 2007.

[11] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.

[12] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN '95)*, vol. vol. 4, pp. 1942–1948, Perth, Western Australia, November-December 1995.

[13] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence Journal*, vol. 1, no. 1, pp. 33–57, 2007.

[14] X. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. vol. 2, pp. 1677–1681, Honolulu, Hawaii, USA, May 2002.

[15] K. E. Parsoupulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of ACM Symposium on Applied Computing (SAC '02)*, pp. 603–607, Madrid, Spain, March 2002.

[16] T. Ray and K. M. Liew, "A swarm methaphor for multiobjective design optimization," *Engineering Optimization*, vol. 34, no. 2, pp. 141–153, 2002.

[17] C. A. Coello Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1051–1056, Honolulu, Hawaii, USA, May 2002.

[18] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[19] J. D. Anderson, *Hypersonic and High Temperature Gas Dynamics*, McGraw-Hill, New York, NY, USA, 1989.

[20] B. J. McBride and S. Gordon, "Computer program for calculation of complex chemical equilibrium compositions and

applications II. User's manual and program description," Tech. Rep. E-8017-1, NASA Lewis Research Center, Cleveland, Ohio, USA, 1996.

[21] K. A. Edquist, "Computations of viking lander capsule hypersonic aerodynamics with comparisons to ground and flight data," in *Proceedings of AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Keystone, Colo, USA, August 2006.

[22] B. Laub and E. Venkatapathy, "Thermal protection system technology and facility needs for demanding future planetary missions," in *Proceedings of International Workshop on Planetary Probe Atmospheric Entry and Descent Trajectory Analysis and Science*, Lisbon, Portugal, October 2003.

[23] M. Tauber and J. Bowles, "The use of atmospheric braking during mars missions," in *Proceedings of the 4th AIAA Thermophysics Conference*, Buffalo, NY, USA, June 1989.

[24] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, Aiaa Education Series, Reston, Va, USA, 1999.

[25] L. Quartapelle and S. Rebay, "Numerical solution of two-point boundary value problems," *Journal of Computational Physics*, vol. 86, no. 2, pp. 314–354, 1990.

[26] L. Quartapelle and A. Scandroglio, "Solution of the Falkner-Skan Equation to Determine Reverse Flow," DIA-SR 03-05, 2003.

[27] R. Armellin and F. Topputo, "A sixth-order accurate scheme for solving two-point boundary value problems in astrodynamics," *Celestial Mechanics and Dynamical Astronomy*, vol. 96, no. 3-4, pp. 289–309, 2006.

[28] J. Kennedy and R. C. Russel, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Fransisco, Calif, USA, 2001.

[29] K. Deb, "Evolutionary algorithms for multi-criterion optimization in engineering design," in *Evolutionary Algorithms in Engineering and Computer Science*, pp. 135–161, John Wiley & Sons, Chichester, UK, 1999.

[30] G. Coath and S. K. Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *Proceedings of the Congress on Evolutionary Computation (CEC '03)*, vol. vol. 4, pp. 2419–2435, Canberra, Australia, December 2003.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 316145, 11 pages
doi:10.1155/2008/316145

Research Article

A Hybrid PSO/ACO Algorithm for Discovering Classification Rules in Data Mining

Nicholas Holden and Alex A. Freitas

Computing Laboratory, University of Kent, Canterbury, Kent CT2 7NF, UK

Correspondence should be addressed to Nicholas Holden, nickpholden@gmail.com

Received 20 July 2007; Accepted 6 March 2008

Recommended by Jim Kennedy

We have previously proposed a hybrid particle swarm optimisation/ant colony optimisation (PSO/ACO) algorithm for the discovery of classification rules. Unlike a conventional PSO algorithm, this hybrid algorithm can directly cope with nominal attributes, without converting nominal values into binary numbers in a preprocessing phase. PSO/ACO2 also directly deals with both continuous and nominal attribute values, a feature that current PSO and ACO rule induction algorithms lack. We evaluate the new version of the PSO/ACO algorithm (PSO/ACO2) in 27 public-domain, real-world data sets often used to benchmark the performance of classification algorithms. We compare the PSO/ACO2 algorithm to an industry standard algorithm PART and compare a reduced version of our PSO/ACO2 algorithm, coping only with continuous data, to our new classification algorithm for continuous data based on differential evolution. The results show that PSO/ACO2 is very competitive in terms of accuracy to PART and that PSO/ACO2 produces significantly simpler (smaller) rule sets, a desirable result in data mining—where the goal is to discover knowledge that is not only accurate but also comprehensible to the user. The results also show that the reduced PSO version for continuous attributes provides a slight increase in accuracy when compared to the differential evolution variant.

1. INTRODUCTION

The focus of this paper is on supervised learning, more specifically, the classification task of data mining. In classification, the knowledge or patterns discovered in the dataset can be represented in terms of a set of rules. A rule consists of an antecedent (a set of attribute values) and a consequent (class):

$$\text{IF } \langle \text{attrib} = \text{value} \rangle \text{ AND} \cdots \text{AND } \langle \text{attrib} = \text{value} \rangle \text{ THEN } \langle \text{class} \rangle. \quad (1)$$

The consequent of the rule is the class that is predicted by that rule. The antecedent consists of a set of terms, where each term is essentially an attribute value pair. More precisely, a term is defined by a triple ⟨*attribute, operator, value*⟩, where value is a value belonging to the domain of *attribute*. The *operator* used in this paper is "=" in the case of categorical/nominal attributes, or "≤" and ">" in the case of continuous attributes. The knowledge representation in the form of rules has the advantage of being intuitively comprehensible to the user. This is important because the general goal of data mining is to discover knowledge that is not only accurate, but also comprehensible [1, 2].

We previously proposed a hybrid particle swarm optimisation [3, 4]/ant colony optimisation [5] (PSO/ACO) algorithm for the discovery of classification rules [6, 7] (the PSO/ACO2 classification algorithm is freely available on Sourceforge: http://sourceforge.net/projects/psoaco2). PSO has been explored as a mean for classification in previous work [8, 9] and shown to be rather successful. However, previous authors have never addressed the case, where PSO is used for datasets containing both continuous and nominal attributes. The same can be said for ACO, where no variants have been proposed that deal directly with continuous attributes [10].

ACO has been shown to be a powerful paradigm when used for the discovery of classification rules involving nominal attributes [11] and is considered state-of-the-art for many combinatorial optimisation problems [12].

Furthermore, ACO deals directly with nominal attributes rather than having to convert the problem first into a binary optimisation problem. When compared to other combinatorial optimisation algorithms (e.g., binary PSO), this reduces the complexity of the algorithm and frees the user from the issues involved in the conversion process. Note that, in the case of a nominal attribute containing more than two values the conversion of the nominal attribute into a binary one in order to use binary PSO is not trivial. For instance, consider the nominal attribute marital status taking on 4 values: "single, married, divorced, and widow." One could convert this attribute into four binary values—"yes" or "no" for each original nominal value—but this has the drawbacks of increasing the number of attributes (and so the dimensionality of the search space) and requiring a special mechanism to guarantee that, out of the 4 values, exactly one is "turned on" in each candidate classification rule. Alternatively, we could try to use a standard PSO for continuous attributes by converting the original nominal values into numbers, say "1, 2, 3, 4," but this introduces an artificial ordering in the values, whereas there is no such order in the original nominal values.

PSO/ACO2 uses ideas from ACO to cope directly with nominal attributes, and uses ideas from PSO to cope with continuous attributes, trying to combine "the best of both worlds" in a single algorithm.

We have shown [6, 7] in two of our previous papers that PSO/ACO is at least competitive with binary PSO in terms of a search mechanism for discovering rules. PSO/ACO is competitive with binary PSO in terms of accuracy, and often beats binary PSO when rule set complexity is taken into account. In this paper, we propose an improved PSO/ACO variant for classification rule discovery (PSO/ACO2) and provide a comprehensive comparison between it and an industrial standard classification algorithm (PART [1]) across 27 datasets (involving both continuous and nominal attributes). We also introduce and compare another PSO/ACO2 classification algorithm variant for continuous data based on differential evolution (DE) [13].

We propose several modifications to the original PSO/ACO algorithm. In essence, the proposed modifications involve changes in the pheromone updating procedure and in the rule initialisation method, as well as—significantly—the splitting of the rule discovery process into two separate phases. In the first phase, a rule is discovered using nominal attributes only. In the second phase, the rule is potentially extended with continuous attributes. This further increases the ability of the PSO/ACO algorithm in treating nominal and continuous attributes in different ways, recognising the differences in these two kinds of attributes (a fact ignored by a conventional PSO algorithm, as mentioned earlier).

The remainder of the paper is organised as follows. Section 2 describes in detail the workings of the modified algorithm (PSO/ACO2). Section 3 discusses the reasons for the modifications. In Section 4, we present the experimental setup and results. In Section 5, we draw some conclusions from the work and discuss possible future research. This paper is a significantly extended version of our recent workshop paper [14].

2. THE NEW PSO/ACO2 ALGORITHM

In this section, we provide an overview of the new version of the hybrid particle swarm optimization/ant colony optimization (PSO/ACO2) algorithm. PSO/ACO2 is a significant extension of the original PSO/ACO algorithm (here denoted PSO/ACO1) proposed in [6, 7]. The PSO/ACO1 algorithm was designed to be the first PSO-based classification algorithm to natively support nominal data—that is, to cope with nominal data directly, without converting a nominal attribute into a numeric or binary one and then applying a mathematical operator to the converted value, as is the case in [8]. The PSO/ACO1 algorithm achieves a native support of nominal data by combining ideas from ant colony optimisation [5] (the successful ant-miner classification algorithm [11]) and particle swarm optimisation [3, 8] to create a classification meta heuristic that supports innately both nominal (including binary as a special case) and continuous attributes.

2.1. PSO/ACO2's sequential covering approach

Both the original PSO/ACO1 algorithm and the new modified version (PSO/ACO2) use a sequential covering approach [1] to discover one classification rule at a time. The original PSO/ACO1 algorithm is described in detail in [6, 7], hereafter we describe how the sequential covering approach is used in PSO/ACO2 as described in Algorithm 1. The sequential covering approach is used to discover a set of rules. While the rules themselves may conflict (in the sense that different rules covering a given example may predict different classes), the "default" conflict resolution scheme is used by PSO/ACO2. This is where any example is only considered covered by the first rule that matches it from the ordered rule list, for example, the first and third rules may cover an example, but the algorithm will stop testing after it reaches the first rule. Although the rule set is generated on a per class basis, it is ordered according to rule quality before it is used to classify examples (to be discussed later in this paper).

The sequential covering approach starts by initialising the rule set (RS) with the empty set. Then, for each class the algorithm performs a WHILE loop, where TS is used to store the set of training examples the rules will be created from. Each iteration of this loop performs one run of the PSO/ACO2 algorithm, which only discovers rules based on nominal attributes, returning the best discovered rule (*Rule*), predicting examples of the current class (C). The rule returned by the PSO/ACO2 algorithm is not (usually) complete as it does not include any terms with continuous values. For this to happen, the best rule discovered by the PSO/ACO2 algorithm is used as a base for the discovery of terms with continuous values.

For the continuous part of the rule, a conventional PSO algorithm (applied only to numeric attributes) with constriction is used [4]. The vector to be optimised consists of two dimensions per continuous attribute, one for an upper bound (ub) and one for a lower bound (lb). At every particle evaluation, the vector is converted to a set of terms (rule conditions) and added to *Rule* produced by the PSO/ACO2

```
RS = ∅ /* initially, Rule Set is empty */
FOR EACH class C
    TS = {All training examples belonging to any class}
    WHILE (Number of uncovered training examples belonging to class C > MaxUncovExampPerClass)
        Run the PSO/ACO2 algorithm to discover the best nominal rule predicting class C called Rule
        Run the standard PSO algorithm to add continuous terms to Rule, and return the best discovered rule BestRule
        Prune BestRule
        RS = RS ∪ BestRule
        TS = TS −{training examples covered by discovered rule}
    END WHILE
END FOR
Order rules in RS by decending Quality
Prune RS removing unnecessary terms and/or rules
```

Algorithm 1: Sequential covering approach used by the hybrid PSO/ACO2 algorithm.

algorithm for fitness evaluation. For instance, if the dataset contained one nominal attribute A_{n0} and one continuous attribute A_{c0}, the PSO/ACO2 algorithm might produce a rule like IF $A_{n0} = \langle value \rangle$ THEN class C. The standard PSO algorithm would then attempt to improve this rule by adding the terms $x_{\mathrm{ub0}} > A_{c0}$ and $x_{\mathrm{lb0}} \leq A_{c0}$, which effectively corresponds to a term of the form $x_{\mathrm{ub0}} > A_{c0} > x_{\mathrm{lb0}}$. Where a single particle's position would be the vectors $\vec{x}_{\mathrm{lb}}, \vec{x}_{\mathrm{ub}}$. The rule for evaluation purposes would be

$$\begin{gathered}\text{IF } A_{n0} = \langle\text{value}\rangle \text{ AND } x_{\mathrm{ub0}} > A_{c0} \\ \text{AND } x_{\mathrm{lb0}} \leq A_{c0} \text{ THEN Class C.}\end{gathered} \tag{2}$$

If the two bounds cross over, (i.e., $x_{\mathrm{lb0}} \geq x_{\mathrm{ub0}}$) both terms are omitted from the decoded rule, but the *Personal Best* position is still updated in those dimensions:

PSO velocity update:

$$v_{\mathrm{id}} = \chi\left(v_{\mathrm{id}} + c_1\varphi_1\left(p_{\mathrm{id}} - x_{\mathrm{id}}\right) + c_2\varphi_2\left(p_{\mathrm{gd}} - x_{\mathrm{id}}\right)\right), \tag{3}$$

PSO position update:

$$x_{\mathrm{id}} = x_{\mathrm{id}} + v_{\mathrm{id}}. \tag{4}$$

To improve the performance of the PSO algorithm, the upper bound for each dimension is initialised (seeded) in the following manner. Each example in the training set is examined to find the lowest and highest values that each continuous attribute takes. From these values, the ranges of each continuous attribute are found. Then, each particle's initial position (for the upper bound dimension) is set to a uniformly distributed position between the value of a randomly chosen seed example's continuous attribute and that value added to the range for that attribute. For the lower bound, the same procedure is also conducted except that the position is initialised at a uniformly distributed position between an example's value (for that attribute) and an example's value minus the range for that attribute. This seeding procedure will likely produce some seeding positions outside the range of the values seen within the dataset. This is an intended feature as for some attributes it might never be beneficial to set lower or upper bounds on their values. The most likely place a particle will be seeded is around the lowest and highest values the seeding examples have (for lower and upper bounds, resp.). However, the seeding examples are from the class being predicted by the rule that the particle is encoding, so if the way in which the values from these examples are distributed is different from all the examples, then hopefully the search will be biased in a useful way. This idea is backed up by an improvement in performance observed in initial experiments.

While the standard PSO algorithm attempts to optimise the values for the upper and lower bounds of these terms, it is still possible that the nominal part of the rule may change. The particles in the PSO/ACO2 algorithm are prevented from fully converging using the Min-Max system (discussed in the next subsection) used by some ACO algorithms, so that an element of random search remains for the nominal part of the rule. This is helpful for the search, as in combination with the continuous terms, some nominal terms may become redundant or detrimental to the overall rule-quality. The exact mechanism of this partially random search is discussed in Section 2.2.

After the *BestRule* has been generated it is then added to the rule set after being pruned using a pruning procedure inspired by Ant-Miner's pruning procedure [11]. Ant-Miner's pruning procedure involves finding the term which, when removed from a rule, gives the biggest improvement in rule quality. When this term is found (by iteratively removing each term tentatively, measuring the rule's quality and then replacing the term) it is permanently removed from the rule. This procedure is repeated until no terms can be removed without loss of rule quality. Ant-Miner's pruning procedure attempts to maximise the quality of the rule in any class, so the consequent class of the rule may change during the procedure. The procedure is obviously very computationally expensive; a rule with n terms may require in the worst case $\sum_{i=1}^{n} i = n(n+1)/2$—that is, $O(n^2)$—rule quality evaluations before it is fully pruned. For this reason, the PSO/ACO2 classification algorithm only uses the Ant-Miner pruning procedure if a rule has less than 20 terms. If there are more than 20 terms then the rule's terms are iterated through once, removing each one if it is detrimental or unimportant for the rule's quality—that is, if the removal of the term does not decrease the classification accuracy of the rule on the training set. Also, for reasons of simplicity the rule's consequent class is fixed throughout the pruning procedure in PSO/ACO2. These alterations were observed (in initial experiments) to make little or no difference to rule quality.

After the pruning procedure, the examples covered by that rule are removed from the training set (TS). An example is said to be covered by a rule if that example satisfies all the terms (attribute value pairs) in the rule antecedent ("IF part"). This WHILE loop is performed as long as the number of uncovered examples of the class C is greater than a user-defined threshold, the maximum number of uncovered examples per class (*MaxUncovExampPerClass*). After this threshold has been reached TS is reset by adding all the previously covered examples. This process means that the rule set generated is unordered—it is possible to use the rules in the rule set in any order to classify an example without unnecessary degradation of predictive accuracy. Having an unordered rule set is important because after the entire rule set is created, the rules are ordered by their quality and not the order they were created in. This is a common approach often used by rule induction algorithms [15, 16]. Also, after the rule set has been ordered it is pruned as a whole. This involves tentatively removing terms from each rule and seeing if each term's removal affects the accuracy of the entire rule set. If that individual term's removal does not affect the accuracy then it is permanently removed. If it does affect the accuracy then it is replaced and the algorithm moves onto the next term, and eventually the next rule. After this process is complete, the algorithm also removes whole rules that do not contribute to the classification accuracy. This is achieved by classifying the training set using the rule list, if any rules do not classify any examples correctly then they are removed.

2.2. *The part of the PSO/ACO2 algorithm coping with nominal data in detail*

The PSO/ACO2 algorithm initially generates the nominal part of the rule, by selecting a (hopefully) near optimal combination of attribute value pairs to appear in the rule antecedent (the way in which rules are assessed is discussed in Section 2.4). The PSO/ACO2 algorithm generates one rule per run and so must be run multiple times to generate a set of rules that cover the training set. The sequential covering approach, as described in Section 2.1, attempts to ensure that the set of rules cover the training set in an effective manner. This section describes in detail the part of the PSO/ACO2 algorithm coping with nominal data, which is the part inspired by ACO. The part of the PSO/ACO2 algorithm coping with continuous data is essentially a variation of standard PSO, where each continuous attribute is represented by two dimensions, referring to the lower and upper bound values for that attribute in the rule to be decoded from the particle, as explained in Section 2.1.

To understand—in an intuitive and informal way—why the PSO/ACO2 algorithm is an effective rule discovery metaheuristic, it may be useful to first consider how one might create a very simple algorithm for the discovery of rules. An effective rule should cover as many examples as possible in the class given in the consequent of the rule, and as few examples as possible in the other classes in the dataset. Given this fact a good rule should have the same attribute-value pairs (terms) as many of the examples in the consequent class. A simple way to produce such a rule would be to use the intersection of the terms in all examples in the consequent class as the rule antecedent. This simple procedure can be replicated by an agent-based system. Each agent has the terms from one example from the consequent class (it is seeded with these terms), each agent could then take the intersection of its terms with its neighbours and then keep this new set. If this process is iterated, eventually all agents will have the intersection of the terms from all examples in the consequent class.

This simple procedure may work well for very simple datasets, but we must consider that it is highly likely that such a procedure would produce a rule with an empty antecedent (as no single term may occur in every example in the consequent class). Also, just because certain terms frequently occur in the consequent class does not mean that they will also not frequently occur in other classes, meaning that our rule will possibly cover many examples in other classes.

PSO/ACO2 was designed to "intelligently" deal with the aforementioned problems with the simple agent-based algorithm by taking ideas from PSO and ACO. From PSO: having a particle network, the idea of a best neighbour and best previous position. From ACO: probabilistic term generation guided by the performance of good rules produced in the past. PSO/ACO2 still follows the basic principle of the simple agent-based system, but each particle takes the intersection of its best neighbour's and previous personal best position's terms in a selective (according to fitness) and probabilistic way.

Each particle in the PSO/ACO2 population is a collection of n pheromone matrices (each matrix encodes a set of probabilities), where n is the number of nominal attributes in a dataset. Each particle can be decoded probabilistically into a rule with a predefined consequent class. Each of the n matrices has two entries, one entry represents an *off* state and one entry represents an *on* state. If the *off* state is (probabilistically) selected, the corresponding (seeding) attribute-value pair will not be included in the decoded rule. If the *on* state is selected when the rule is decoded, the corresponding (seeding) attribute-value pair will be added to the decoded rule. Which value is included in this attribute-value pair (term) is dependant on the seeding values.

The seeding values are set when the population of particles is initialised. Initially, each particle has its past best state set to the terms from a randomly chosen example from class C—the same class as the predefined consequent class for the decoded rule. From now on, the particle is only able to decode to a rule with attribute values equal to the seeding terms, or to a rule without some or all those terms. This may seem at first glance counter intuitive as flexibility is lost—each particle cannot be translated into any possible rule, the reasons for this will be discussed later.

Table 1

Colour		Has_fur		Swims	
(*on*) Colour = blue	*Off*	(*on*) Has_fur = false	*off*	(*on*) Swims = true	*off*
0.01	0.99	0.8	0.2	0.9	0.1

```
Initialise population
REPEAT for MaxInterations
    FOR every particle x
        /*Rule Creation */
        Set Rule $R_x$ = “IF ∅ THEN C”
        FOR every dimension d in x
            Use roulette selection to choose whether the state should be set to off or on. If it is on then the
            corresponding attribute-value pair set in the initialisation will be added to $R_x$; otherwise (i.e., if
            off is selected) nothing will be added.
        LOOP
        Calculate Quality $Q_x$ of $R_x$
        /*Set the past best position */
        P = x’s past best state
        $Q_p$ = P’s quality
        IF $Q_x > Q_p$
            $Q_p = Q_x$
            P = x
        END IF
    LOOP
    FOR every particle x
        P = x’s past best state
        N = the best state ever held by a neighbour of x according to N’s quality $Q_N$
        FOR every dimension d in x
            /*Pheromone updating procedure */
            IF $P_d = N_d$ THEN
                pheromone_entry corresponding to the value of $N_d$ in the current $x_d$ is increased by $Q_p$
            ELSE IF $P_d$ = off AND seeding term for $x_d \neq N_d$ THEN
                pheromone_entry for the off state in $x_d$ is increased by $Q_p$
            ELSE
                pheromone_entry corresponding to the value of $N_d$ in the current $x_d$ is increased by $Q_p$
            END IF
            Normalize pheromone_entries
        LOOP
    LOOP
LOOP
RETURN best rule discovered
```

Algorithm 2: The part of the PSO/ACO2 algorithm coping with nominal data.

Each pheromone matrix entry is a number representing a probability and all the entries in each matrix for each attribute add up to 1. Each entry in each pheromone matrix is associated with a minimum, positive, and nonzero pheromone value. This prevents a pheromone from dropping to zero, helping to increase the diversity of the population (reducing the risk of premature convergence).

For instance, a particle may have the following three pheromone matrices for attributes *Colour*, *Has_fur* and *Swims*. It was seeded with an example: *Colour* = *Blue*, *Has_fur* = *False*, *Swims* = *True*, *Class* = *Fish* as shown in Table 1.

The probability of choosing the term involving the attribute colour to be included in the rule is low, as the *off* flag has a high probability in the first pheromone matrix (0.99). It is likely that the term *Has_fur* = *False* will be included in the decoded rule as it has a high probability (0.8) in the second pheromone matrix. It is also likely that the term *Swims* = *True* will be included in the decoded rule.

The most likely rule decoded from this set of pheromone matrices is

$$\begin{gathered}\text{IF Has\_fur} = \text{False AND}\\ \text{Swims} = \text{True THEN}\\ \text{Class} = \text{Fish}.\end{gathered} \tag{5}$$

Algorithm 2 shows the modified PSO/ACO2 algorithm proposed in this paper and utilised in Algorithm 1. Firstly, the population of particles is initialised. Each particle is seeded with terms from a randomly selected example, as described earlier. Initially, in each dimension the pheromone

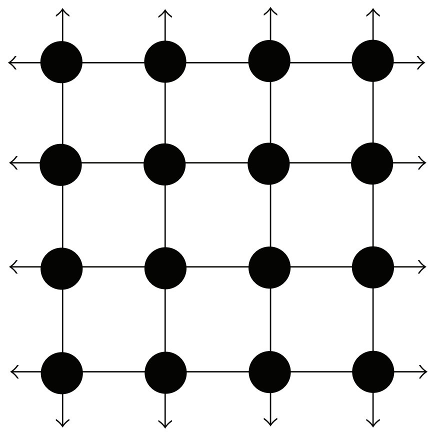


Figure 1: The Von-Neumann topology used by PSO/ACO2.

for the *on* state is set to 0.9 and the pheromone for the *off* state is set to 0.1. The first loop (REPEAT statement) iterates the whole population for *MaxIterations*. Then, for each particle x a rule is built probabilistically from its pheromone matrices. For each dimension d in x, roulette selection—a well-known selection method in evolutionary algorithms—is used to decide if the on or off state should be selected [6]. In PSO/ACO2 this simply involves the following rule:

```
IF rand() > pheromone in entry for on state THEN
        Select on state
ELSE                                                    (6)
        Select off state
```

where rand() returns a number from the interval $[0 \cdots 1]$ with a uniform probability distribution. If the *on* state is selected, then the corresponding term is added to the antecedent of R_x, this is an attribute value pair, where the attribute corresponds to the dimension d and the value corresponds to the initial seeding value. After this process has been repeated for every dimension, the quality Q_x of the rule is calculated. If the new Q_x is greater than the previous best Q_p, then the particle's state is saved as P.

After the rule creation phase, the pheromone is updated for every particle. Each particle finds its best neighbour's best state (N) according to Q_N (the quality of the best rule N has ever produced). The particles are in a static topology, so each particle has a fixed set of neighbour particles throughout the entire run of the algorithm. PSO/ACO2 uses Von-Neumann [17] topology, where the particles are arranged in a 2D grid, each particle having four neighbours. This topology was chosen as it consistently performed the best in initial experiments. This is likely due to the level of connectivity present in this particular topology, that is, enough connectivity but not too much (global topology showed to be suboptimal due to premature convergence to a local optimum in the search space).

The pheromone updating procedure is influenced by two factors, the best state a particle x has ever held (the state P), and the best state ever held by a neighbour particle N. As discussed previously, each dimension can take two values and so it has two corresponding pheromone entries, one for the *on* state and one for the *off* state. These states are examined in every dimension d and the following rules are applied. If P_d is the same as N_d, then an amount of pheromone equal to Q_p (the quality of P) is added to the pheromone entry in x_d corresponding to the value of P_d. The second pheromone updating rule is (the ELSE statement) if P_d is not the same as N_d then an amount of pheromone equal to Q_p is removed from the pheromone entry in x_p corresponding to the value of P_d. In other words, if a particle and its neighbour have both found that a particular attribute-value is good, then pheromone is added to the entry corresponding to it. If they disagree about that attribute value, then pheromone is removed from the corresponding entry.

There is a caveat in this pheromone updating procedure given by the "ELSE IF" statement in Algorithm 2. It states that if P_d is *off* and the current particle and its best neighbour do not have the same seeding terms, then increase the likelihood of choosing the *off* state (by adding pheromone to the pheromone entry corresponding to the *off* value). The reason for this is to maintain the idea from the simple agent-based system described earlier in this section. That is, when two particles have different seeding terms, then those terms should tend to be omitted. Without this caveat the opposite would happen, the probability of the term being omitted would become less, as pheromone is usually removed from P_d (*off*) if P_d and N_d do not agree. A more detailed examination of the effect of the pheromone updating rules can be seen in Table 2 with this caveat being shown in the second row from the bottom.

If after this process is completed any pheromone entry is less than a predefined minimum amount, then it is set to that amount (0.01). Importantly, this allows the pheromone entry that is not the best state to increase due to the normalisation procedure. This increase will occur if pheromone is removed from a state. If this happens, the amount of pheromone in the matrix becomes less than 1 and, as long as both entries have a greater than zero amount of pheromone, when the matrix is normalised both entries will increase. It also aids search in a conceptually similar way to mutation in GAs and the Min-Max system in the ACO literature [5].

In Table 2, the six possible scenarios for pheromone updating are described given the differing states of P_d, N_d and also the seeding term for x_d. These outcomes are controlled by the pheromone updating rules shown in Algorithm 2 (discussed previously). The first and last cases shown in the table are quite intuitive, if both P_d and N_d agree on a state that state is made more likely, this allows the algorithm to converge on a good solution that the particles agree on. Cases of the second type are shown in the second and fifth rows, where P_d and N_d have different seeding terms. In these cases, the particle makes it more likely that the conflicting term will be omitted from the decoded rule, by selecting the *off* state. This feature allows the particle to create rules that generalise well, covering more examples from the consequent class (discussed further in Section 3). Cases of the third type—which involve a disagreement between P_d and N_d about whether or not the seeded term should be used in the rule decoded from the current particle—are shown in the third and fourth rows. These cases bias the search towards N_d so that each particle tries to emulate its best neighbour. In the third row; if N_d = *off* then the probability of x_d decoding to *off* will be increased (by increasing the pheromone associated with the *off* state). In the fourth row; if $N_d = w$ (and N_d and x_d have the same seeing terms) the probability of x_d decoding to *on* will be increased. The cases of the third type allow the particles to come to a consensus about the best set of states. By trying to emulate its best neighbour, each particle has the potential to create (in future iterations) a new past best state (P) based on a mix of its own current P and N.

2.3. A differential evolution classification algorithm for continuous data

Differential evolution (DE) [13] has become a popular optimisation algorithm for continuous spaces. It has been shown to be very competitive with respect to other search algorithms [18]. It can be considered a kind of evolutionary algorithm whose population moves around the search space in a greedy manner. Each offspring is (usually) generated from the weighted combination of the best member of the population with two or more other members of the population. If this offspring has a higher fitness then it replaces its parent, if not it is discarded. There are many variations on this basic evolutionary method for DE, the ones used in this paper are the rand-to-best/1 and best/1 as they seemed to be good during initial experiments. The formulas for offspring generation in these DE variants are as follows:

DE update equation for best/1:

$$v = x_{\text{best}} + F(x_{r1} - x_{r2}), \tag{7}$$

DE update equation for rand-to-best/1

$$v = x_i + F(x_{\text{best}} - x_i) + F(x_{r1} - x_{r2}), \tag{8}$$

where v is the offspring vector, x_{best} is the best member of the current population. x_{r1} and x_{r2} are randomly-selected members of the population, x_i is the parent vector, and F is the weighting factor. The other parameters for the algorithm are: *CR*—a crossover likelihood factor and *NP*—the population size.

As we already have the framework for a PSO-based classification algorithm using sequential covering (as detailed in Section 2.1), the DE optimiser can essentially be "plugged-in," replacing standard PSO as the rule discovery algorithm (for continuous data). To make any comparisons fair the same seeding mechanism and procedure to deal with crossed over attribute value bounds is used both in the PSO and DE classification algorithm.

2.4. Quality measures

It is necessary to estimate the quality of every candidate rule (decoded particle). A measure must be used in the training phase in an attempt to estimate how well a rule will perform in the testing phase. Given such a measure, it becomes possible for an algorithm to optimise a rule's quality (the fitness function). In our previous work [6], the quality measure used was *Sensitivity* × *Specificity* (9) [19].

Quality Measure used by PSO/ACO1 [6]:

$$\text{Sensitivity} \times \text{Specificity} = \text{TP}/(\text{TP}+\text{FN}) \times \text{TN}/(\text{TN}+\text{FP}), \tag{9}$$

where TP, FN, FP and TN are, respectively, the number of true positives, false negatives, false positives, and true negatives associated with the rule [1]:

(i) true positives (TP) are the number of examples that match the rule antecedent (attribute values) and also match the rule consequent (class). These are desirable correct predictions;

(ii) false positives (FP) are the number of examples that match the rule antecedent, but do not match the rule consequent. These are undesirable incorrect predictions;

(iii) false negatives (FN) are the number of examples that do not match the rule antecedent but do match the rule consequent. These are undesirable uncovered cases and are caused by an overly specific rule;

(iv) true negatives (TN) are the number of examples that do not match the rule antecedent and do not match the rule consequent. These are desirable and are caused by a rule's antecedent being specific to its consequent class.

In the new PSO/ACO2 classification algorithm proposed in this paper, the quality measure is *Precision* with Laplace correction [1, 20], as per (10). In initial experiments this quality measure was observed to lead to the creation of rules that were more accurate (when compared to the original quality measure shown in (9).

New Quality Measure used by PSO/ACO2:

$$\text{Laplace-corrected Precision} = (1+\text{TP})/(1+\text{TP}+\text{FP}), \tag{10}$$

We observed that in some cases (when using (10) as a quality measure), rules would be generated covering very few examples. These cases were likely due to the way in which the *Laplace-Corrected Precision* measure penalises false positives very severely (when compared to *Sensitivity* × *Specificity*). To stop this less than ideal situation we added the following conditional statement to the new quality measure:

```
IF TP < MinTP
    Rule Quality = Laplace-Corrected Precision*0.1,
ELSE
    Rule Quality = Laplace-Corrected Precision,
END IF
                                                    (11)
```

where *MinTP* is the least number of correctly covered examples that a rule has to cover before it is given a "normal" value, as computed by (10). When a rule covers too few examples the quality is severely reduced (by a factor of 100).

TABLE 2: Different pheromone updating scenarios.

Seeding Term for x_d	P_d	N_d	Outcome for entries in x_d
$\langle$value$\rangle = w$	(on) $\langle$value$\rangle = w$	(on) $\langle$value$\rangle = w$	*on* pheromone increased *off* pheromone decreased
$\langle$value$\rangle = w$	(on) $\langle$value$\rangle = w$	(on) $\langle$value$\rangle \neq w$	*on* pheromone increased *off* pheromone decreased
$\langle$value$\rangle = w$	(on) $\langle$value$\rangle = w$	*off*	*on* pheromone increased *off* pheromone decreased
$\langle$value$\rangle = w$	*off*	(on) $\langle$value$\rangle = w$	*on* pheromone increased *off* pheromone decreased
$\langle$value$\rangle = w$	*off*	(on) $\langle$value$\rangle \neq w$	*on* pheromone increased *off* pheromone decreased
$\langle$value$\rangle = w$	*off*	*off*	*on* pheromone increased *off* pheromone decreased

TABLE 3: An example single class dataset, R's are records, A_n's are nominal attributes.

	A_{n1}	A_{n2}	A_{n3}
R_1	a	a	a
R_2	a	a	b
R_3	a	a	b
R_4	b	b	b
R_5	b	b	b
R_6	b	b	b

This procedure reduces the quality below the quality of any normal rule, but still allows the particles covering fewer than *MinTP* examples to compare their solutions effectively. In our experiments, we set *MinTP* to 10, but any comparably small number will have a similar effect.

3. MOTIVATIONS FOR PSO/ACO2 AND DISCUSSION

The modified algorithm (PSO/ACO2) proposed in this paper differs from the original algorithm (PSO/ACO1) proposed in [6, 7] in five important ways. Firstly, PSO/ACO1 attempted to optimise both the continuous and nominal attribute values present in a rule antecedent at the same time, whereas PSO/ACO2 takes the best nominal rule built by PSO/ACO2 and then attempts to add continuous attributes to it using a conventional PSO algorithm. Secondly, the original algorithm used a type of rule pruning to create seeding terms for each particle, whilst PSO/ACO2 uses all the terms from an entire training example (record). Thirdly, in PSO/ACO1 it was possible for a particle to select a value for an attribute that was not present in its seeding terms, whilst in PSO/ACO2 only the seeding term values may be added to the decoded rule. Fourthly, the pheromone updating rules have been simplified to concentrate on the optimisation properties of the original algorithm. In PSO/ACO1 pheromone was added to each entry that corresponded to the particle's past best state, its best neighbour's best state, and the particle's current state in proportion to a random learning factor. Now, pheromone is only added to a pheromone matrix entry in the current particle when N_d and P_d match, or taken away when they do not. Fifthly, the algorithm now prunes the entire rule set after creation, not simply on a per rule basis.

In PSO/ACO2, the conventional PSO for continuous data and the hybrid PSO/ACO2 algorithm for nominal data have been separated partially because they differ quite largely in the time taken to reach peak fitness. It usually takes about 30 iterations (depending on the complexity of the dataset) for the pheromone matrices to reach a stable state in PSO/ACO2, whilst it tends to take considerably longer for the standard PSO algorithm to converge. Due to this fact, the standard PSO algorithm's particles set past best positions in quite dissimilar positions, as their fitness is dependant on the quickly converging part of the PSO/ACO2 algorithm coping with nominal data. This causes high velocities and suboptimal search, with a higher likelihood of missing a position of high fitness. Therefore, separating the rule discovery process into two stages—one stage for the part of the PSO/ACO2 algorithm coping with nominal data and one stage for the part of the PSO/ACO2 algorithm coping with continuous data (essentially a variation of a standard PSO)—provides more consistent results.

Secondly, in the PSO/ACO1 algorithm, sets of seeding terms were pruned before they were used. This aggressive pruning algorithm used a heuristic to discard certain terms. This is less than ideal as the heuristic does not take into account attribute interaction, and so potentially useful terms are not investigated.

Table 4: Accuracy of labelled approaches in UCI datasets, with standard deviation and Student's t-test shadings.

Data set	Accuracy		Average rule size		Average rule set length	
	PSO/ACO2	PART	PSO/ACO2	PART	PSO/ACO2	PART
Autos	76.63 ± 8.36	79.83 ± 11.43	2.8 ± 0.17	2.54 ± 0.24	16.0 ± 1.25	14.2 ± 2.74
Balance scale	82.72 ± 4.77	79.38 ± 7.81	2.56 ± 0.17	3.13 ± 0.16	26.6 ± 1.07	38.9 ± 3.25
Breast cancer	72.62 ± 6.84	69.7 ± 7.8	1.73 ± 0.26	1.91 ± 0.18	12.4 ± 2.27	17.3 ± 4.72
Breast w	93.42 ± 3.79	93.7 ± 4.05	1.17 ± 0.09	1.01 ± 0.03	9.9 ± 1.6	10.4 ± 3.03
Credit-a	85.31 ± 4.14	84.23 ± 3.35	2.94 ± 0.31	2.46 ± 0.34	22.7 ± 2.0	30.8 ± 9.66
Credit-g	67.9 ± 5.82	69.7 ± 4.4	4.23 ± 0.19	3.01 ± 0.25	54.3 ± 1.89	77.0 ± 4.57
Crx	85.6 ± 2.84	84.54 ± 2.8	2.94 ± 0.28	2.44 ± 0.31	22.5 ± 3.1	29.9 ± 8.67
Diabetes	72.67 ± 4.98	74.36 ± 4.51	3.88 ± 0.29	1.88 ± 0.23	33.4 ± 1.43	7.1 ± 1.52
Glass	70.95 ± 7.5	65.43 ± 11.45	3.11 ± 0.18	2.7 ± 0.28	20.4 ± 1.35	16.1 ± 1.6
Heart-c	77.38 ± 5.45	78.72 ± 5.92	3.33 ± 0.19	2.42 ± 0.21	12.6 ± 0.84	19.9 ± 2.42
Heart-statlog	81.11 ± 6.16	78.15 ± 6.64	3.17 ± 0.44	2.88 ± 0.34	9.7 ± 1.34	18.4 ± 1.9
Ionosphere	88.06 ± 4.91	90.04 ± 4.68	3.33 ± 0.79	2.35 ± 0.43	3.6 ± 0.97	8.9 ± 1.91
Iris	94.67 ± 5.26	90.67 ± 7.17	0.93 ± 0.14	1.02 ± 0.05	3.0 ± 0.0	4.3 ± 1.42
Iris_d	94.67 ± 6.13	94.0 ± 5.84	0.68 ± 0.04	0.76 ± 0.06	3.2 ± 0.42	4.4 ± 0.97
Lymph	83.05 ± 6.67	83.19 ± 9.47	1.89 ± 0.15	2.26 ± 0.42	14.7 ± 2.0	10.0 ± 1.25
Mmushroom	99.9 ± 0.11	100.0 ± 0.0	1.86 ± 0.18	1.55 ± 0.02	8.7 ± 0.48	12.8 ± 0.42
Promoters	81.0 ± 12.12	83.91 ± 7.91	1.02 ± 0.05	1.02 ± 0.14	5.1 ± 0.32	6.9 ± 1.2
Segment	96.67 ± 1.17	96.67 ± 0.84	2.8 ± 0.27	3.07 ± 0.17	21.9 ± 0.99	26.3 ± 1.7
Sonar	75.05 ± 9.11	72.52 ± 10.57	2.6 ± 0.63	2.23 ± 0.49	4.4 ± 1.58	7.4 ± 1.17
Soybean	87.01 ± 6.53	90.57 ± 3.96	2.08 ± 0.21	2.66 ± 0.16	24.2 ± 1.03	32.1 ± 3.21
Tic-tac-toe	100.0 ± 0.0	93.85 ± 2.7	2.67 ± 0.0	2.65 ± 0.11	9.0 ± 0.0	38.3 ± 3.06
Vehicle	73.05 ± 4.45	73.29 ± 2.77	3.85 ± 0.18	3.84 ± 0.38	37.8 ± 1.2	34.0 ± 3.02
Vowel	86.16 ± 3.47	85.05 ± 5.79	4.2 ± 0.25	3.55 ± 0.21	29.0 ± 0.82	50.5 ± 3.57
Wisconsin	94.87 ± 2.53	94.43 ± 2.06	1.21 ± 0.07	1.02 ± 0.03	10.2 ± 1.87	9.9 ± 3.11
Kr-vs-kp	99.47 ± 0.51	99.37 ± 0.29	2.25 ± 0.15	3.03 ± 0.35	18.7 ± 2.0	22.7 ± 1.34
Zoo	97.18 ± 6.25	94.18 ± 6.6	1.14 ± 0.18	1.48 ± 0.12	7.1 ± 0.32	7.6 ± 0.52
Splice	93.48 ± 1.24	92.79 ± 1.65	3.0 ± 0.07	2.65 ± 0.1	88.0 ± 2.91	99.6 ± 6.1

Table 5: Overall performance of PSO/ACO2 against PART according to WEKA's Student's t-test (out of 27 datasets).

	Accuracy	Average rule size	Average rule length
Total	1	−6	14

To understand the reasons behind the last two modifications, it is important to understand how the algorithms find good rules. In both PSO/ACO1 and PSO/ACO2, sets of terms are generated by mixing together the experiences of the particles and their neighbours. As the entries in the pheromone matrices converge and reach one (and zero), better rules should be generated more often. In PSO/ACO1, the levels of the pheromone in the matrices are influenced by three factors (current state, past best state, and best neighbours' best state) [6]. If these factors do not agree, then the pheromone matrix will be slow to converge. Slow convergence can sometimes be advantageous as the algorithm should not prematurely converge to a local maximum. However, in PSO/ACO1 the result of this slow convergence is usually destructive, as incompatible terms can be mixed together over and over again. Incompatible terms are terms that do not cover any of the same examples. For instance, in Table 3, incompatible terms are $A_{n1} = a$ and $A_{n2} = b$. A rule including both these terms would have a quality of zero as it would not cover any examples. This problem is addressed by the third modification in PSO/ACO2, now incompatible terms will not be mixed. This modification also ensures a particle will always cover at least one example (the seeding example) even if all the terms are included in the decoded rule. This was not the case in PSO/ACO1 as at the beginning of the search many incompatible terms could be mixed, creating many particles with zero fitness.

In PSO/ACO2, the pattern being investigated by the particles will likely include relatively general terms—an example might be a rule including the term $A_{n3} = b$ in Table 3. It is the job of the PSO/ACO2 algorithm to find terms that interact well to create a rule that is not only general to the class being predicted (covering many examples of that class) but also specific to the class (by not covering examples in other classes). It is also the job of the PSO/ACO2 algorithm to turn off terms that limit the generality of the rule without adding specificity to it. This trade-off between specificity and generality (or sensitivity) is calculated by the rule quality measure. It is clear, in Table 3, that including values for A_{n1} and A_{n2} will not ever lead to the most general rule (the optimal rule only has one term, $A_{n3} = b$). Due to the new pheromone updating procedures a particle would choose the *off* state for these conflicting attributes quickly.

Table 6: Accuracy of labelled approaches in UCI datasets containing only continuous attributes, with standard deviation and Student's t-test shadings.

Data set	PSO/ACO2 using standard PSO	PSO/ACO2 using differential evolution (P) (rand-to-best/1)	PSO/ACO2 using differential Evolution (P) (best/1)
Balance-scale	81.46 ± 6.33	81.94 ± 5.1	80.33 ± 3.4
Diabetes	76.19 ± 3.79	74.11 ± 5.82	74.04 ± 5.39
Glass	67.68 ± 11.27	66.73 ± 10.98	68.45 ± 7.69
Heart-statlog	79.26 ± 6.34	76.3 ± 8.41	76.3 ± 10.36
Ionosphere	84.33 ± 6.51	84.33 ± 3.85	80.33 ± 7.09
Iris	87.33 ± 11.95	94.67 ± 5.26	93.67 ± 5.54
Segment	95.89 ± 0.82	93.9 ± 1.38	94.2 ± 0.85
Sonar	69.71 ± 10.02	76.9 ± 7.89	77.4 ± 10.29
Vehicle	70.1 ± 5.75	65.72 ± 6.24	66.01 ± 3.94

4. RESULTS

For the experiments, we used 27 datasets from the well-known UCI dataset repository [21]. We performed 10-fold cross validation [1], and run each algorithm 10 times for each fold for the stochastic algorithms (PSO/ACO2 and the DE algorithm).

Both the part of the PSO/ACO2 algorithm coping with nominal data and the standard PSO algorithm (i.e., the part of PSO/ACO2 coping with continuous data) had 100 particles, and these two algorithms ran for a maximum of 100 iterations (MaxIterations) per rule discovered. In all experiments, constriction factor $\chi = 0.72984$ and social and personal learning coefficients $c1 = c2 = 2.05$ [4]. PSO/ACO2 is freely available on sourceforge.

A freely available java implementation of DE by Mikal Keenan and Rainer Storn was used in all experiments presented in this paper [22]. The default values (as stated in the JavaDoc) of $F = 0.5$, CR $= 1$ for rand-to-best/1 and $F = 0.5$, CR $= 1$ for best/1 were used, so as not to delve into the realms of parameter tweaking. To maintain consistency with the PSO algorithm, a population size of 100 was used and the number of fitness evaluations was kept the same as the PSO variant. As to not bias the comparison, the PSO/ACO2 and DE classification algorithms were only compared on continuous attribute only datasets. This was done in an attempt to prevent any bias that might occur from the interaction with the nominal PSO/ACO2 part of the algorithm.

MaxUncovExampPerClass was set to 10 as this is standard in the literature [11]. As mentioned previously, PSO/ACO2 uses Von-Neumann topology, where each particle has four neighbours, with the population being connected together in a 2D grid. The corrected WEKA [1] statistics class was used to compute the standard deviation of the predictive accuracies and to apply the corresponding corrected two-tailed Student's t-test—with a significance level of 5%—in the results presented in Tables 4, 5, and 6.

The algorithms compared in Table 4 are PSO/ACO2 and PART. PART is WEKA's improved implementation of C4.5 rules [1]. PART extracts rules from decision trees created by J48 (WEKA's implementation of C4.5). We compared PSO/ACO2 against this algorithm as it is considered an industry standard.

The first two columns (not including the dataset column) in Table 4 show the percentage predictive accuracy of both algorithms. The second two columns show the average rule size (number of terms, or attribute-value pairs) for the rules generated for each dataset. The third two columns show the average rule set size for each dataset; this is simply the average number of rules in each rule set. The measures of average rule size and average rule set size give an indication of the complexity (and so comprehensibility) of the rule sets produced by each algorithm. The shading in these six columns denotes a statistically significant win or a loss (according to the corrected WEKA two-tailed Student's t-test), light grey for a win and dark grey for a loss against the baseline algorithm (PART). Table 5 shows the overall score of the PSO/ACO2 classification algorithm against PART, considering that a significant win counts as "+1" and a significant loss counts as a "−1," and then calculating the overall score across the 27 datasets.

It can be seen from Tables 4 and 5 that in terms of accuracy PART and PSO/ACO2 are quite closely matched. This is not completely surprising as PART is already considered to be very good in terms of predictive accuracy. Furthermore, there is only one result that is significant in terms of accuracy; the accuracy result for the tic-tac-toe dataset. However, if one scans through the accuracy results it is clear that often one algorithm outperforms the other slightly. In terms of rule set complexity, the algorithms are much less closely matched. When the average rule size results are taken as a whole, PSO/ACO2 generates longer rules in 6 cases overall. Although the average rule size results are significant, the real impact of having a rule that is under one term longer is arguable (as is found in many cases). The most significant results by far are in the rule set size columns. PSO/ACO produces significantly smaller rule sets in 14 cases overall, sometimes having tens of rules less than PART. These improvements have a tangible effect on the comprehensibility of the rule set as a whole.

The reduced PSO/ACO2 classification algorithm (for continuous data only) using standard PSO (i.e., coping only with continuous attributes) is compared with the PSO/ACO2 classification algorithm using two DE variants in Table 6. Although there is one significant loss for each DE variant against the PSO variant, both algorithms seem to slightly outperform each other on certain datasets. Also, there is no clear winner between the different DE variants.

5. DISCUSSION AND CONCLUSIONS

We have conducted our experiments on 27 public domain "benchmark" datasets used in the classification literature, and we have shown that PSO/ACO2 is at least competitive with PART (an industry standard classification algorithm) in terms of accuracy, and that PSO/ACO2 often generates *much* simpler (smaller) rule sets. This is a desirable result in data mining—where the goal is to discover knowledge that is not only accurate but also comprehensible to the user.

At present, PSO/ACO2 is partly greedy in the sense that it builds each rule with the aim of optimising that rule's quality individually, without directly taking into account the interaction with other rules. A less greedy, but possibly more computationally expensive way to approach the problem would be to associate a particle with an entire rule set and then to consider the quality of the entire rule set when evaluating a particle. This is known as the "Pittsburgh approach" in the evolutionary algorithm literature, and it could be an interesting research direction. Also the nominal part of the rule is always discovered first and separately from the continuous part, it could be advantageous to use a more "coevolved" approach.

REFERENCES

[1] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2nd edition, 2005.

[2] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: an overview," in *Advances in Knowledge Discovery and Data Mining*, pp. 1–34, AAAI, Menlo Park, Calif, USA, 1996.

[3] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann/Academic Press, San Francisco, CA, USA, 2001.

[4] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, Honolulu, Hawaii, USA, April 2007.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*, The MIT Press, Cambridge, Mass, USA, 2004.

[6] N. Holden and A. A. Freitas, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 100–107, Pasadena, Calif, USA, June 2005.

[7] N. Holden and A. A. Freitas, "Hierarchical classification of G-protein-coupled receptors with a PSO/ACO algorithm," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '06)*, pp. 77–84, Indianapolis, Ind, USA, May 2006.

[8] T. Sousa, A. Silva, and A. Neves, "Particle swarm based data mining algorithms for classification tasks," *Parallel Computing*, vol. 30, no. 5-6, pp. 767–783, 2004.

[9] I. De Falco, A. Della Cioppa, and E. Tarantino, "Facing classification problems with particle swarm optimization," *Applied Soft Computing*, vol. 7, no. 3, pp. 652–658, 2007.

[10] A. A. Freitas, R. S. Parpinelli, and H. S. Lopes, "Ant colony algorithms for data classification," in *Encyclopedia of Information Science and Technology*, M. Khosrou-Pour, Ed., pp. 420–424, IGI Publishing, Hershey, Pa, USA, 2nd edition, 2005.

[11] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321–332, 2002.

[12] M. Dorigo and K. Socha, "An introduction to ant colony optimization," in *Handbook of Approximation Algorithms and Metaheuristics*, T. F. Gonzalez, Ed., pp. 26.1–26.14, Chapman & Hall/CRC, Boca Raton, Fla, USA, 2007.

[13] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[14] N. Holden and A. A. Freitas, "A hybrid PSO/ACO algorithm for classification," in *Proceedings of the 9th Genetic and Evolutionary Computation Conference Workshop on Particle Swarms: The Second Decade (GECCO '07)*, pp. 2745–2750, ACM Press, London, UK, July 2007.

[15] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, Calif, USA, 1993.

[16] G. L. Pappa, *Automatically evolving rule induction algorithms with grammar-based genetic programming*, Ph.D. thesis, Computing Laboratory, University of Kent, Canterbury, UK, 2007.

[17] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '02)*, pp. 1671–1676, Honolulu, Hawaii, USA, May 2002.

[18] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06)*, vol. 1, pp. 485–492, ACM Press, Seattle, Wash, USA, July 2006.

[19] D. J. Hand, *Construction and Assessment of Classification Rules*, John Wiley & Sons, New York, NY, USA, 1997.

[20] P. Clark and R. Boswell, "Rule induction with CN2: some recent improvements," in *Proceedings of the 5th European Working Session on Learning (EWSL '91)*, pp. 151–163, Porto, Portugal, March 1991.

[21] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, "UCI repository of machine learning databases," 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[22] Differential Evolution Java Implementation, July 2007, http://www.icsi.berkeley.edu/~storn/code.html#java.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 514879, 10 pages
doi:10.1155/2008/514879

Research Article

An Improved Particle Swarm Optimizer for Placement Constraints

Sheng-Ta Hsieh,[1, 2] Tsung-Ying Sun,[1] Chan-Cheng -Liu,[1] and Cheng-Wei Lin[1, 3]

[1] *Department of Electrical Engineering, National Dong Hwa University, Hualien 97401, Taiwan*
[2] *Department of Communication Engineering, Oriental Institute of Technology, Taipei County 22061, Taiwan*
[3] *Silicon Motion Technology Corporation, Jhubei City, Hsinchu County 302, Taiwan*

Correspondence should be addressed to Tsung-Ying Sun, sunty@mail.ndhu.edu.tw

Received 20 July 2007; Revised 28 November 2007; Accepted 14 January 2008

Recommended by Alex Freitas

This paper presents a macrocell placement constraint and overlap removal methodology using an improved particle swarm optimization (PSO). Several techniques have been proposed to improve PSO, such as methods to prevent the floorplan from falling into the local minimum and to assist in finding the global minimum. The proposed method can deal with various kinds of placement constraints and can process them simultaneously. Experiments employing MCNC and GSRC benchmarks show the difference in the efficiency and robustness of proposed method in the exploration for more optimal solutions through restricted placement and overlap removal compared with other methods. The proposed approach exhibits rapid convergence and leads to more optimal solutions than other related approaches; furthermore, it displays efficient packing with all the constraints satisfied.

1. INTRODUCTION

VLSI floorplanning is an important aspect in chip design. It involves the placement of a set of rectangular circuit modules (macrocells) on a chip to minimize the total area and the total interconnecting wire length and without overlap between two modules since larger chip sizes increase production cost while longer wire lengths increase power consumption and decrease system performance.

The physical placement of circuits in VLSI chips or *system on chips* (SoC's) has been given constant attention in recent years. Early research on the placement problem applied force to reduce the overlap between cells [1]. However, [2–4] show the generation of overlap-free placements and [5, 6] show the generated layouts with cell overlaps. While allowing overlaps during the process of placement was shown to obtain better floorplanning solutions, this process could not guarantee the entire elimination of overlaps. Later, Vijayan and Tsay [7] proposed the topological overlap removal method, an approach that removes a redundant edge from two critical paths and repeats the process continuously until it makes a significant improvement on the layout area. However, the drawback to this approach was the random selection, if there were two or more edges qualified for removal, it cannot predict which removal will lead to a better placement. Asato and Ali [8] improved this method by removing all redundant edges from one of the constraint graphs following the iteration.

The recent approach, conducted by Alupoaei and Katkoori, proposed a macrocell overlap removal algorithm that was based on the ant colony optimization (ACO) method [9]. Each ant in the colony generated a placement based on the macrocells' relative positions and information regarding the optimal placement obtained by previous colonies. The disadvantage to this macrocell movement procedure was that the initial relationship between macrocells would influence the final result directly, plus the result could be trapped in the local minimum.

All the approaches mentioned above have their advantages and disadvantages; however, none of them is able to cover placement constraint problems. Due to in the analog design, designers are also interested in a particular kind of placement constraint called symmetry, some recent literature on this problem can be found in [10, 11]. The floorplanner in [12] can handle alignment constraints which may appear in bus-based routing. The floorplanners in [4, 13, 14] can

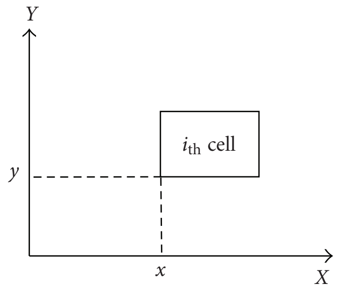


Figure 1: Cell definition.

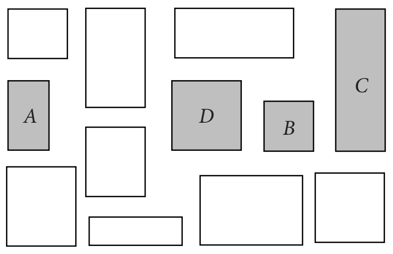


Figure 2: Example of alignment constraint.

handle preplaced constraints, where some modules are fixed in position. Different approaches are used to handle the different kinds of constraints, but there are no unified methods that can handle all constraints simultaneously.

As opposed to these previously mentioned methods,this paper utilizes particle swarm optimization (PSO) with an overlap detection and removal mechanism to search for the optimal placement solution. PSO is a swarm intelligence method that roughly models the social behavior of swarms. The advantage with this behavioral model is the search process which allows stochastic return of particles towards previously successful regions in the search space. This method has been proven to be efficient on a plethora of problems in science and engineering.

In order to make PSO more capable when dealing with placement problems, a *turnaround factor* [15] is adopted to extend the particles' search space. A disturbance mechanism [16] is also introduced to prevent the solution from falling into the local minimum. Furthermore, the proposed method can handle several kinds of placement constraints simultaneously, including boundary, preplaced, range, abutment, alignment, and clustering. Users can dictate a mixed set of constraints and their preferred way of arrangement for the assigned macrocell. The proposed floorplanner will then be able to place these macrocells based on the given criteria.

Section 2 contains the definition of various placement constraint problems; Section 3 describes the original PSO methodology; Section 4 presents the proposed methods; and Section 5 presents the experimental results, which aside from the comparisons with ACO and B*-tree in unconstrained conditions, also contains the experiment results of situations involving different kinds of randomly selected multi-constraints in floorplanning. Section 6 of the paper contains the conclusion.

2. PLACEMENT CONSTRAINTS

2.1. Cell definition

During floorplanning, information is given to a set of macrocells; the information includes their width, length, and cell numbers. In this paper, the floorplanning problem is addressed with a number of placement constraints, other than the module information, some placement constraints are given between the modules. The goal of proposed method is to plan all macrocells' positions on a chip such that all the placement constraints can be satisfied and the area and interconnection cost minimized.

In this paper, the notation $c(i, x, y)$ is used to denote the ith cell's location, where x and y are presented, ith cell's position on x-axis and y-axis, respectively. Note that, the cell positions are defined as the cell's lower left corner. Figure 1 illustrates these definitions.

2.2. Relative and absolute constraints

In general, placement constraints can be classified as relative and absolute. A relative placement constraint describes the relationship between two modules, and an absolute placement constraint describes the relationship between a module and the chip. Using relative placement constraints, users can restrict the horizontal or vertical distance between two modules to a certain range of values; using absolute placement constraints, the placement of a module is modified with respect to the whole chip, like restricting a cell to a certain distance from the boundary of the chip. Modules classified under either kind of constraints can be set as restriction conditions for particle movement during the PSO evolution process. Details of the definition and applications will be described in the following section.

2.3. General use placement constraints

Using the above specifications for absolute and relative placement criteria, many different kinds of placement constraints can be created. In this section, a few commonly used constraints will be picked up and show how each can be specified using a combination of the relative and absolute placement constraints.

2.3.1. Alignment

To align modules A, B, C, and D horizontally (Figure 2), the following constraints can be imposed:

$$y_A = y_B = y_C = y_D. \tag{1}$$

The horizontal axes of each module to be restricted the same; they will thus align horizontally. A similar definition can be applied to their vertical alignment.

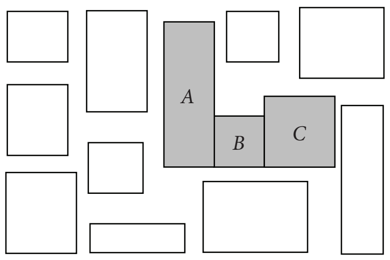


FIGURE 3: Example of abutment constraint.

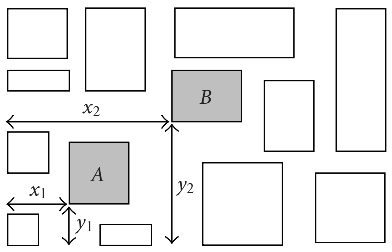


FIGURE 4: Example of preplace constraint.

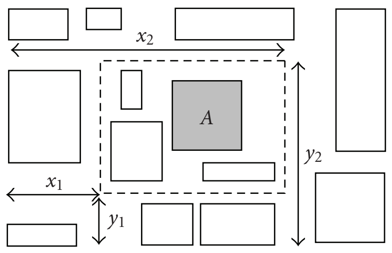


FIGURE 5: Example of range constraint.

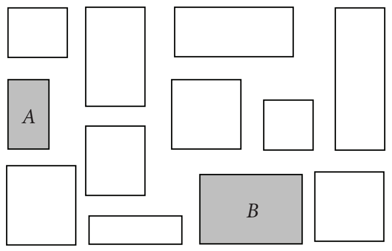


FIGURE 6: Example of boundary constraint.

2.3.2. Abutment

To abut modules A, B, and C horizontally (Figure 3), the following constraints can be imposed:

$$\begin{aligned} x_B &= x_A + w_A, \\ x_C &= x_B + w_B, \\ y_A &= y_B = y_C, \end{aligned} \tag{2}$$

where w_A and w_B are the widths of modules A and B, respectively. In this formulation, the horizontal axes of each module are the same; so they will align horizontally. It also restricts module B to being placed next to module A on the right-hand side, and so on. The end result is their horizontal abutment.

2.3.3. Preplace constraint

To preplace module A with its lower left corner at axis (x_1, y_1) and module B with its lower left corner at axis (x_2, y_2) (Figure 4), the following constraints can be imposed:

$$\begin{aligned} x_A &= x_1, \qquad y_A = y_1, \\ x_B &= x_2, \qquad y_B = y_2. \end{aligned} \tag{3}$$

Module A is restricted to be x_1 units from the left boundary and y_1 units from the bottom boundary. A similar definition can be applied to module B. Each restricted module will be preplaced with its lower left corner in the final packing.

2.3.4. Range constraint

To restrict the position of module A in the range $\{(x_A, y_A) \mid x_1 \le x_A \le x_2,\ y_1 \le y_A \le y_2\}$ (Figure 5), the following constraints can be imposed:

$$x_A = [x_1, x_2], \qquad y_A = [y_1, y_2]. \tag{4}$$

In this formulation, module A is restricted to be x_1 to x_2 units from the left boundary and to be y_1 to y_2 units from the bottom boundary, restricting module A to the designated rectangular region.

2.3.5. Boundary constraint

To place module A along the left boundary and place module B along the bottom boundary in the final packing (Figure 6), the following constraints can be imposed:

$$x_A = 0, \qquad y_B = 0. \tag{5}$$

In this formulation, module A is restricted to be 0 units from the left boundary, so module A will be abut with the left boundary in the final packing. Module B is restricted to be 0 units from the bottom boundary, so module B will abut with the bottom boundary as required.

2.3.6. Clustering constraint

To cluster modules A and B around C at a distance of most units away vertically (Figure 7), the following constraints can be imposed:

$$\begin{aligned} x_A &= x_C + w_C, \qquad y_A = [y_C, y_C + h_C], \\ x_B &= x_C + w_C, \qquad y_B = [y_C, y_C + h_C]. \end{aligned} \tag{6}$$

In this formulation, the vertical distances of A and B from C are restricted to be at most a certain unit in vertical directions, so they will cluster around C at a vertical distance of at most h_C units away. The horizontal distances of A and B from C can also be restricted in a similar way.

3. PARTICLE SWARM OPTIMIZATION (PSO)

The PSO is a population-based optimization technique that was proposed by Eberhart and Kennedy [17] in 1995, in

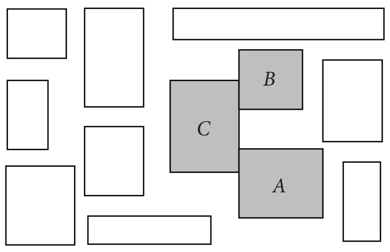


FIGURE 7: Example of cluster constraint.

which the population is referred to as a *swarm*. The particles exhibit the ability of fast convergence to local and/or global optimal positions over a small number of generations.

A swarm in PSO consists of a number of particles. Each particle represents a potential solution to the optimization task. All of the particles iteratively discover a probable solution. Each particle generates a position according to the new velocity and the previous positions of the cell. This is compared with the best position generated by previous particles in the cost function and the best one is kept; each particle accelerates in the direction of not only the local best solution but also the global best position. If a particle discovers a new probable solution, other particles will move closer to explore the region in more details [18].

Let s denote the swarm size. In general, there are three attributes, current position x_i, current velocity v_i, past best position $Pbest_i$, and global best position $Gbest$, as the main guidelines for each particle in the search space. Each particle in the swarm is iteratively updated according to the aforementioned attributes. Assuming that the objective function f is to be minimized so that the particle contains N dimensions, the new velocity of every particle is updated using

$$\begin{aligned} v_{i,j}(t+1) = {} & wv_{i,j}(t) + c_1 r_{1_{i,j}}(t)\left[Pbest_{i,j}(t) - x_{i,j}(t)\right] \\ & + c_2 r_{2_{i,j}}(t)\left[Gbest_j(t) - x_{i,j}(t)\right]. \end{aligned} \tag{7}$$

For all $j \in 1, \dots, N$, $v_{i,j}$ is the velocity of the jth dimension of the ith particle, w is the inertia weight of velocity [19], c_1 and c_2 denote the *acceleration coefficients*, r_1 and r_2 are elements from two uniform random sequences in the range $(0, 1)$, and t is the number of generations. The new position of the ith particle is calculated as follows:

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1). \tag{8}$$

The past best position of each particle is updated by

$$Pbest_i(t+1) = \begin{cases} Pbest_i(t), & \text{if } f\left(x_i(t+1)\right) \geq f\left(Pbest_i(t)\right), \\ x_i(t+1), & \text{otherwise.} \end{cases} \tag{9}$$

The global best position $Gbest$ found by all particles during previous three steps is defined as

$$Gbest(t+1) = \arg\min_{Pbest_i} f\left(Pbest_i(t+1)\right), \quad 1 \leq i \leq s. \tag{10}$$

4. PSO ALGORITHM FOR MACROCELL OVERLAP REMOVAL AND PLACEMENT

The first step to using PSO as a way to handle the macrocell overlap removal and placement is defining each macrocell's position as a dimension of a particle. That is, 20 dimensional particles can be used to optimize placement problems whose circuit contains 20 macrocells. The definition of the macrocells' position is stated in the previous section. While a swarm consists of a number of particles, the particle's movement will lead the module to find another potential or superior solution for placement. For the initial state of the placement, each module was randomly generated in the floorplanning and overlap was allowed. After a number of generations, the distance between each of the modules will shrink, meaning the chip size will get smaller. The overlap between the two modules will be eliminated by proposed overlap detection and removal mechanism.

4.1. *Handling placement constraints by PSO*

Both relative and absolute placement constraints can be set as a particle's movement constraints and can be included in the PSO algorithm. Because of the combination of horizontal and vertical vectors into the particles' moving vector, in this paper, a constraints handling process is instituted in the proposed floorplanner to guarantee that each macrocell's arrangement will follow the placement constraints. Therefore, each particle can move according to the original PSO algorithm, but any moving vector that violates the setting rules (placement constraints) will be stopped by the constraints handling process; the movement of particles could be limited to one or both directions and the moving vectors could be reduced in each move to comply with the placement constraints.

Here, the two general kinds of placement constraints, absolute and relative, are taken into consideration. For relative placement constraint, users can restrict the horizontal or vertical distance between two modules to a certain value, or to a certain range of values.

In this paper, a master-slave concept is introduced to define the relationship between cells. For a set constrained macrocells, one of them will be defined as *master*; the others will be *slaves*. All the slave cells will be moved only after the master cell has moved. Furthermore, the movement strategy of slave cells will also obey the cell's relationship defined by the constraints. For example, if cell A and cell B are both constrained to vertical alignment, and cell A is defined as the master cell, then cell B will be moved only after cell A has moved, and the x-axis of cell B will be set according to cell A's current position. Absolute placement constraint is similarly specified except it does not involve or need the master-slave concept.

4.2. *Overlap detection and removal mechanism*

Before the global best position $Gbest$ is updated to move the macrocell to a new position, the proposed floorplanner should detect if the current macrocell overlaps with any

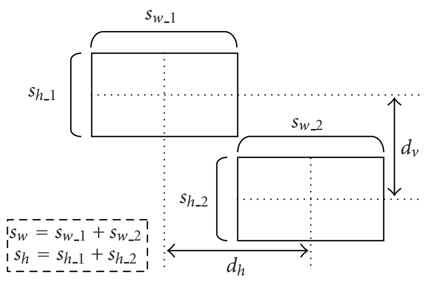


FIGURE 8: The distance measure between two macrocells mechanism.

existing macrocells. The horizontal distance d_h and vertical distance d_v between two macrocells depicted in Figure 8 are measured via the macrocells' center. Then, they are compared with the half sum of two the macrocells' height s_h and half sum of the two macrocells' width s_w to estimate the occurrence of overlap using

$$\text{Overlap} = \begin{cases} 1, & \text{if } d_h < 0.5s_w,\ d_v < 0.5s_h, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

The area with an existing macrocell will be regarded as a forbidden region. If a macrocell is moved to intersect a forbidden region (overlaps with another macrocell), this macrocell will be eliminated and restored to its former location. The macrocells' new position will be updated until it is overlap-free. This process requires more time for computation (particle movement), but it will guarantee that each macrocell's placement is overlap-free.

4.3. *Turnaround factor for particles' movement*

In the original PSO, the moving vector (velocity) of each particle is decided according to its past best solution and the global best solution. This procedure makes sense in evolutional computing, where all new generations would inherit past generations' experience or ability and move all particles toward the global best solution.

In some applications, such as placement, the optimal solution will change in each time slot. The original PSOs will ignore some of the better solutions that are beyond the particles in the searching space, and continue to drive particles toward a specific direction according to previous experiences. Particles will miss the optimal solution in the current slot and spend more time on turning particles in the right direction, and toward to the global optimal solution. Figure 9 shows the particles' searching behavior, driven by (7) in a two-dimensional search space. It would lead particles to some searching spaces and skip the areas which have already been searched before in previous generations. This searching behavior is more suited to applications whose optimal solution is fixed at a specific position, but not time varying or dynamic. In different applications, the algorithm should be modified to be more suitable under the circumstances.

For example, in order to prevent macrocells from overlapping, all areas with existing macrocells will be regarded as

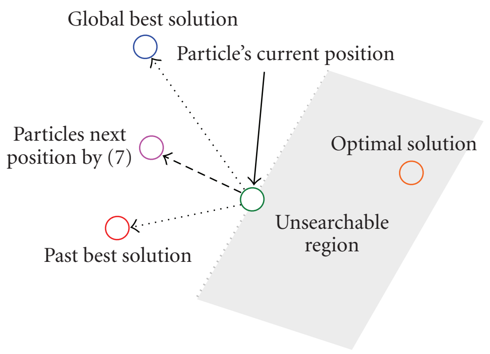


FIGURE 9: Particle movement behavior.

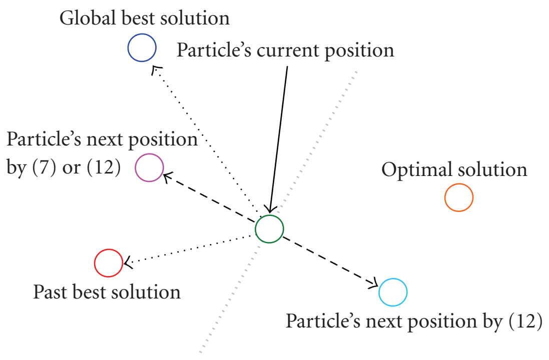

Global best solution
Particle's current position
Particle's next position by (7) or (12)
Optimal solution
Past best solution
Particle's next position by (12)

FIGURE 10: Extended searching space.

a forbidden region; that is, each rearrangement of a macrocell must be overlap-free. Thus, particles may not be able to find feasible solution by moving forward (according to previous experience) in the current generation. Therefore, to search for better placement solutions, particles should not move only forwards, but also backwards to find possible solutions.

To enhance the particles' searching ability and save evolution time, the velocity update equation is modified as follows:

$$\begin{aligned} v_{i,j}(t+1) = T\{&wv_{i,j}(t) + c_1 r_{1_{i,j}}(t)[Pbest_{i,j}(t) - x_{i,j}(t)] \\ &+ c_2 r_{2_{i,j}}(t)[Gbest_j(t) - x_{i,j}(t)]\}, \end{aligned} \tag{12}$$

where T denotes the *turnaround factor* [15]. Normally, the T would be set at 1 (moving forward). The particle's movement will follow the original PSO. If the particle can not find a feasible solution however, the T of the regenerated macrocell will be switched to -1 (moving backward) for this generation. After feasible solutions have been found, the particles will then follow in the current direction in its successive generations to find better solutions. Thus, T must be restored to 1. Particles will then continue the solution searching by moving according to the original PSO. Figure 10 illustrates the extended searching space possible with proposed method.

The turnaround factor can extend the particles' searching space from the usual frontal 180 degrees to a full 360 degrees. It can stop particles from missing the solutions located behind them or located in already explored areas.

```
if disturbance mechanism activated
    random select a particle (macrocell) and place it randomly
    in the searching space (enclosing rectangle of the
    floorplanning)
    if overlap
        restore the particle's position
    else
        update the particle's position
    endif
endif
```

Algorithm 1: Pseudocode of disturbance mechanism for placement.

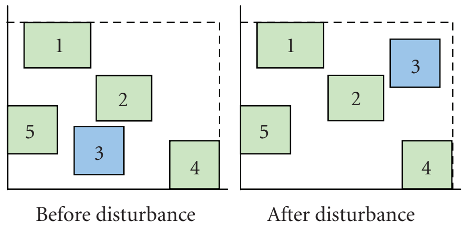


Figure 11: Macrocell replaced by disturbance.

4.4. *Disturbance mechanism*

In this paper, the PSO placement procedure is incorporated with a disturbance mechanism [16], the mutation-like evolutionary strategy, to prevent particles from being trapped in the local optimum.

Because each macrocell has a unique length-height ratio, after numbers of generations, the gap and free space between each macrocell should become smaller as each cell moves closer together. But it is almost unavoidable for the solution to fall into the local minimum during each particle's search for a better placement solution.

Hence, a disturbance mechanism was added into the PSO process to prevent the search from falling into the local minimum and to escape from the trap to find the global minimal. The disturbance mechanism will pick up a particle randomly and disturb its position information; a randomly selected macrocell will be reordered in the enclosed rectangle of the floorplan while the disturbance mechanism is active. The placement process was then redone to achieve a better placement solution. Figure 11 shows that the macrocell has been replaced by disturbance mechanism.

The selected macrocell will keep following the PSO iteration and continue to find new placement solutions. The disturbance mechanism will cause a slight rearrangement of the floorplan and prevent the solution from falling into the local minimum.

It takes many tries for the particles to find the best solution for the following generation. After the disturbance mechanism activates, the removed macrocell will create an empty space for other macrocells, so it can also reduce the PSO iteration time and help other macrocells find the next solution. The disturbance mechanism provides two advantages with a single process. The pseudocode of the disturbance mechanism for placement is presented Algorithm 1.

4.5. *Objective function*

In general, the objective functions are dependent on the application's demands. In this paper, the focus will be smaller chip size and shorter wire length also. Just as chip size gets smaller, the wire length should also be shorter. Therefore, we defined the objective function f as follows:

$$f = \begin{cases} \sqrt{D_h^2 + D_v^2}, & \text{if } i = 0, \\ D_h + D_v + |D_h - D_v|, & \text{if } i = 1, \end{cases} \tag{13}$$

where D_h and D_v are the horizontal distance and the vertical distance with coordinates $(0, 0)$, respectively, and i is produced by a random integer generator with a result of either 1 or 0. The function $\sqrt{D_h^2 + D_v^2}$ can make cells get as close as possible, while the function value gets smaller. The function $D_h + D_v + |D_h - D_v|$ can arrange cells in a uniform distribution. Both cost functions are integral parts and must work alternately in PSO evolution. Adopting the first one only will form a tight cell arrangement but the side effect is a placement that forms an arced contour, on the other hand, adopting the second one only will create results with better contours but will lead to loose cells arrangements.

4.6. *PSO for placement constraint and overlap removal*

The dimension numbers N denote the number of macrocells and the particle number is defined as $1 \leq i \leq s$. The macrocells were randomly placed on the floorplan and overlap was allowed initially. The x_i represents the macrocells' current positions and the initial state of $v_{i,j}$, $Pbest_i$, and $Gbest$ were set to 0. After that, particles are moved by (11) and (8); vectors violating movement restraints are cut off with the constraints handling process. The overlap detection and removal mechanism will also eliminate any cells that overlap with other macrocells. Then, the new local best position would be updated with (9) and the global best position would be updated with (10). In order to guarantee that each constrained macrocell would not violate placement constraints, all constrained modules are set to have

```
Create and initiate a 2*N-dimensional PSO: P
Repeat:
    Disturbance mechanism
    Execute PSO to update P by (11) and (8)
    Constraints handling process
    Overlap detection and removal by (12)
    for each particle $i \in [1, \dots, s]$
        if $f(x_i) < f(Pbest_i)$
            then $Pbest_i = x_i$
        if $f(Pbest_i) < f(Gbest)$
            then $Gbest = Pbest_i$
    endfor
Until termination condition is reached
```

Algorithm 2: Pseudocode.

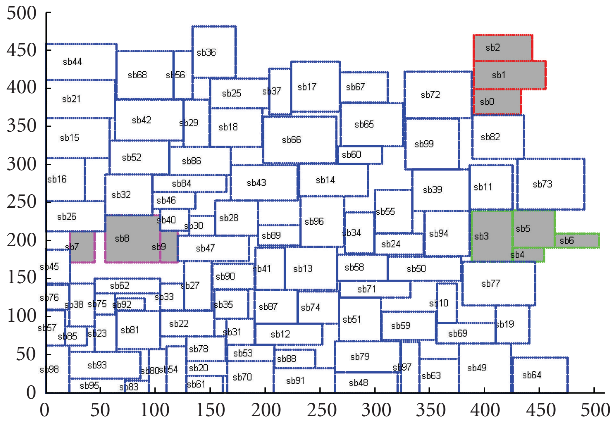


Figure 12: Cells 0–3 abut vertically, cells 4, 6, and 7 cluster around cell 5, and cells 7–9 align horizontally.

higher priority in each movement during the earlier generations to ensure that all set constraints were reached, in other words, if a nonconstrained module overlaps with a constrained module, this nonconstrained module would be removed and regenerated. Thus, all the particles would keep moving to find a better solution until it reached the goal or met the termination condition. The pseudocode of improved PSO for macrocell overlap removal and placement is presented in Algorithm 2. The dimension of each particle will be 2^*N since there are 2 parameters (x- and y-axes) per cell.

4.7. *Time complexity*

In this paper, the time complexity of the proposed floorplanner is $O(N_d/N_p(n-1)+d+c)$, where N_d is the dimensions of particles, N_p the number of particles in the swarm, n is the number of macrocells, $(n-1)$ is overlap detection and removal (compare one macrocell with others), d is the disturbance mechanism, and c is the constraints handling process. Thus, N_d equals to n. The number of particles is typically less than n, since the number of particles in a swarm has no need to surpass a constant limit, and the activating probability of disturbance mechanism is much smaller than n^2. Thus, the overall time complexity of the algorithm is $O(n^2)$.

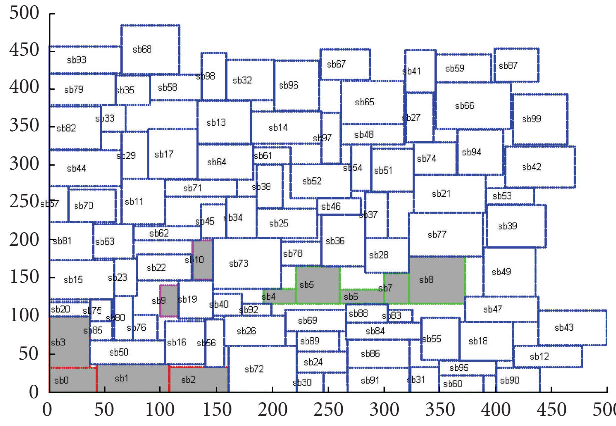


Figure 13: Cells 0–3 cluster at lower left corner of the chip, cells 4–8 abut horizontally, and cells 9-10 are placed at range 100–200 in both axes.

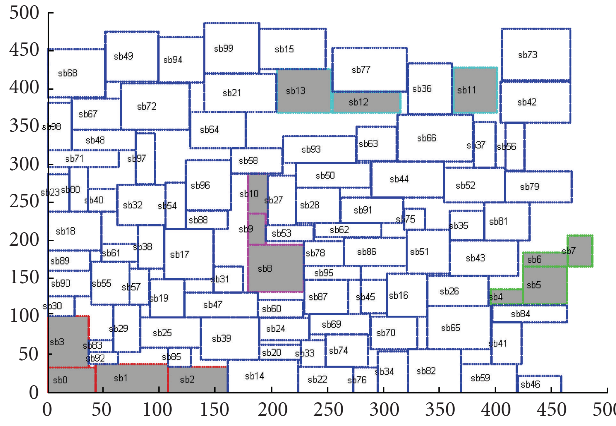


Figure 14: Cells 0–3 cluster at lower left corner of the chip, cells 4, 6, and 7 cluster around cells 5, cells 8–10 abut vertically, and cells 11-12 align horizontally.

5. EXPERIMENTS

The experiments in this study employed GSRC and MCNC benchmarks [20] for the proposed placement constraints and overlap removal procedures and compared the results with other related researches. All the macrocells were set as hard IP modules and without rotation. The simulation programs were written in Matlab [21], and the results were obtained on Pentium 4 1.7 GHz with 512 MB RAM. The initial parameters of w, c_1, and c_2 for the proposed method and [19] were empirically set as 0.12, 0.25, and 0.25, respectively. Both their particle numbers were set as ten and the termination condition was set at 100 generations. The algorithm and parameters of ACO and B*-tree were complete following the original setting of [9] and [4], respectively. Each floorplanner was run 25 times for each of the benchmarks and their average outcomes of wire length, chip area, and run time were recorded.

Table 1: Compared proposed method with other approaches without placement constraints.

Circuits	Cells	Proposed method			PSO [19]			Ant Colony [9] 100 Colony 200 Ants			B*-tree [4]		
		Area (mm^2)	Wire (mm)	Run time	Area (mm^2)	Wire (mm)	Run time	Area (mm^2)	Wire (mm)	Run time	Area (mm^2)	Wire (mm)	Run time
ami49	49	51.46	179.28	1m52s	58.75	212.75	1m57s	63.32	218.36	7m58s	50.71	208.38	1m49s
n_50	50	0.27	12.8	1m40s	0.39	17.85	1m50s	0.41	18.01	7m12s	0.27	14.97	1m40s
n_100	100	0.24	24.17	3m43s	0.41	35.07	3m51s	0.45	40.36	15m01s	0.27	31.75	11m39s
n_200	200	0.24	47.74	8m48s	0.42	71.93	8m55s	0.46	73.68	45m59s	0.28	84.01	1h36m12s
n_300	300	0.38	89.84	15m43s	0.40	132.91	15m59s	0.57	119.43	1h24m1s	0.49	204.44	5h26m50s

Table 2: The proposed method with different placement constraints.

Circuits	Cells	4 macrocells in 2 random selected constraints			8 macrocells in 4 random selected constraints			12 macrocells in 4 random selected constraints		
		Area (mm^2)	Wire (mm)	Run time	Area (mm^2)	Wire (mm)	Run time	Area (mm^2)	Wire (mm)	Run time
ami49	49	50.7	186.29	2m30s	51.91	177.672	2m51s	52.21	186.82	3m2s
n_50	50	0.28	13.47	2m26s	0.29	13.65	3m21s	0.3	13.86	3m56s
n_100	100	0.27	25.91	5m5s	0.27	26.17	5m23s	0.28	26.42	5m50s
n_200	200	0.28	52.11	10m59s	0.28	52.47	11m18s	0.28	52.95	12m25s
n_300	300	0.42	97.12	18m58s	0.43	97.09	20m6s	0.43	97.79	20m35s

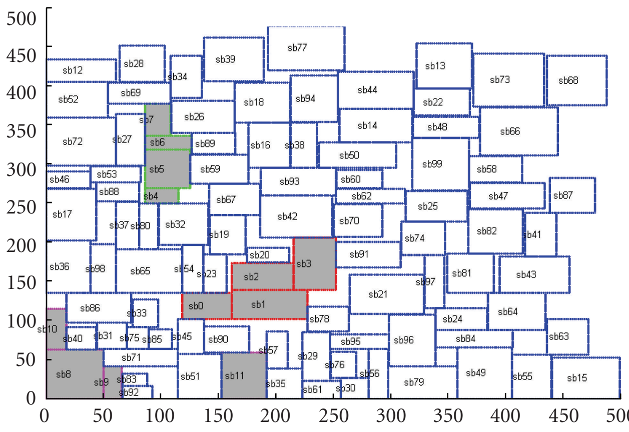


Figure 15: Cells 0–3 clustering, cells 4–7 abut vertically, cells 8–10 cluster at lower left corner of the chip, and cell 11 places at bottom boundary.

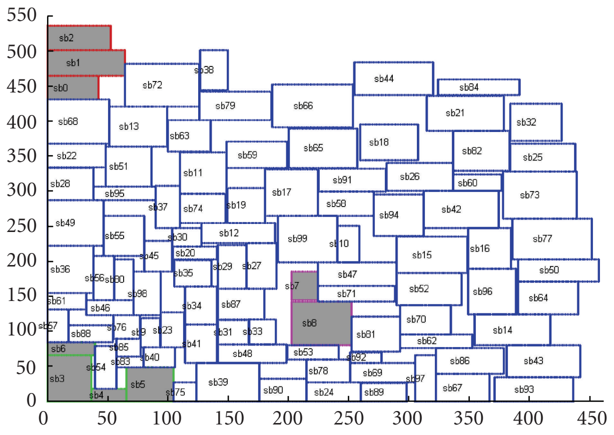


Figure 16: Cells 0–2 abut vertically and place at left boundary, cells 3–6 cluster at lower corner of the chip, and cells 7-8 align vertically.

The experimental results of placements are shown in Table 1. Compared with related methods, the proposed method is more efficient in finding solutions with respect to chip area or wire length and can prevent solutions fall into the local minimum. The convergence of [19] is quite acceptable. Good solutions can be found after reasonable computation time. The ACO method allows overlap in the initial stage; this means it not only has to execute the ACO procedure, detect and remove overlaps, but must also deal with constraint graphs for each macrocell. However, the ACO method shows impressive results for the floorplan. In the B*-tree structure, all macrocells are overlap-free, saving detection time to find and remove overlapping cells, but it would then spend more time arranging the cells after one cell was removed, exchanged, or placed. In a minority of cases, it displays more efficiency with placement. Once the cell number was increased, however, the computation burden of this method would increase nonlinearly, furthermore, it would likely fall into the local minimum. In unconstrained cases, the proposed method expressed more optimal results for wire length and chip area, and shorter run times than ACO [9], PSO [19], and B*-tree [4].

Because there are hundreds of combinations for placement constraints mentioned in Section 2, it was not possible to present every case here. In placement constraints simulation, the constraint types were selected by a random number generator. The constrained macrocells are selected by its serial number. For example, if the user wants to have 8 constrained macrocells in the GSRC n100 benchmark, cells sb0 to sb7 would be chosen. The placement constraint experimental results are shown in Table 2. Even when restricted to different numbers of macrocells during placement, the proposed method exhibited reasonable outcomes for chip size, wire length and run times, and fit all selected constraints. Furthermore, the proposed method has no cases of constraint violations in each experiment. Only little fluctuation in computation times in the proposed method when using different restricted cells can also be observed. The proposed floorplanner is more robust than related research in being more adaptable to various placement requirements. The final five multiconstraint floorplanning results in GSRC n100 were illustrated in Figures 12–16.

6. CONCLUSIONS

This paper presents a method to handle various placement constraints in floorplanning. Two techniques are introduced, which are the turnaround factor and the disturbance mechanism, to improve capability of the algorithm. When it was applied to problems without constraints, the proposed method exhibited more efficiency and robustness when searching for the solution than ACO, PSO, and B*-tree. In order to perform a wide optimal solution search, the overlap between macrocells was allowed in the initial stage, the proposed method can guarantee that all the overlaps will be eliminated in the final floorplan. The experimental results proved that the proposed method can lead to more optimal solutions within reasonable computation times for hard IP modules in unconstrained placement. Furthermore, the propose method also has ability to deal with constrained placement problems. Several benchmarks were adopted for testing and the results were very reliable. Placements with all the constraints satisfied can be obtained efficiently by the proposed method.

7. FUTURE WORKS

The authors' future works will focus on finding ways to apply their method to different architecture in order to enhance the efficiency of floorplanning, and deal with soft IP module placement problems.

REFERENCES

[1] N. Quinn and M. A. Breuer, "A forced directed component placement procedure for printed circuit boards," *IEEE Transactions on Circuits and Systems*, vol. 26, no. 6, pp. 377–388, 1979.

[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.

[3] P.-N. Guo, T. Takahashi, C.-K. Cheng, and T. Yoshimura, "Floorplanning using a tree representation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 2, pp. 281–289, 2001.

[4] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu, "B*-trees: a new representation for nonslicing floorplans," in *Proceedings of the 37th on Design Automation Conference (DAC '00)*, pp. 458–463, Los Angeles, Calif, USA, June 2000.

[5] G. Sigl, K. Doll, and F. M. Johannes, "Analytical placement: a linear or a quadratic objective function?" in *Proceedings of the 28th Design Automation Conference (DAC '91)*, pp. 427–432, San Francisco, Calif, USA, June 1991.

[6] F. Mo, A. Tabbara, and R. K. Brayton, "A force-directed macrocell placer," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '00)*, pp. 177–180, San Jose, Calif, USA, November 2000.

[7] J. Vijayan and R.-S. Tsay, "A new method for floor planning using topological constraint reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 12, pp. 1494–1501, 1991.

[8] B. Asato and H. H. Ali, "An improved floor planning algorithm using topological constraint reduction," in *Proceedings of the IEEE 38th Midwest Symposium on Circuits and Systems (MWSCAS '95)*, pp. 310–313, Rio de Janeiro, Brazil, August 1995.

[9] S. Alupoaei and S. Katkoori, "Ant colony system application to macrocell overlap removal," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1118–1123, 2004.

[10] F. Balasa and K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 7, pp. 721–731, 2000.

[11] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy, "Efficient solution space exploration based on segment trees in analog placement with symmetry constraints," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '00)*, pp. 497–502, San Jose, Calif, USA, November 2002.

[12] X. Tang and D. F. Wong, "Floorplanning with alignment and performance constraints," in *Proceedings of the 39th Design Automation Conference (DAC '02)*, pp. 848–853, New Orleans, La, USA, June 2002.

[13] H. Murata, K. Fujiyoshi, and M. Kaneko, "VLSI/PCB placement with obstacles based on sequence-pair," in *Proceedings of the International Symposium on Physical Design (ISPD '97)*, pp. 26–31, Napa, Calif, USA, April 1997.

[14] F. Y. Young and D. F. Wong, "Slicing floorplans with pre-placed modules," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '98)*, pp. 252–258, San Jose, Calif, USA, November 1998.

[15] C.-L. Lin, S.-T. Hsieh, T.-Y. Sun, and C.-C. Liu, "PSO-based learning rate adjustment for blind source separation," in *Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS '05)*, pp. 181–184, Hong Kong, December 2005.

[16] T.-Y. Sun, S.-T. Hsieh, and C.-W. Lin, "Particle swarm optimization incorporated with disturbance for improving the efficiency of macrocell overlap removal and placement," in *Proceedings of the International Conference on Artificial Intelligence (ICAI '05)*, pp. 122–125, Las Vegas, Nev, USA, June 2005.

[17] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.

[18] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training

algorithms for neural networks," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 110–117, Indianapolis, Ind, USA, April 2003.

[19] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE World Congress on Computational Intelligence*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[20] http://vlsicad.cs.binghamton.edu/benchmarks.html.

[21] http://www.mathworks.com/.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 574613, 14 pages
doi:10.1155/2008/574613

Research Article

Inverse Parameter Identification Technique Using PSO Algorithm Applied to Geotechnical Modeling

Joerg Meier,[1] Winfried Schaedler,[1] Lisa Borgatti,[2] Alessandro Corsini,[2] and Tom Schanz[1]

[1] *Labor für Bodenmechanik, Bauhaus-Universität Weimar, Coudraystraße 11 C, 99421 Weimar, Germany*
[2] *Dipartimento di Scienze della Terra, Università degli Studi di Modena e Reggio Emilia, Largo Sant' Eufemia 19, 41100 Modena, Italy*

Correspondence should be addressed to Winfried Schaedler, winfried.schaedler@uni-weimar.de

Received 24 July 2007; Accepted 14 January 2008

Recommended by Jim Kennedy

This paper presents a concept for the application of particle swarm optimization in geotechnical engineering. For the calculation of deformations in soil or rock, numerical simulations based on continuum methods are widely used. The material behavior is modeled using constitutive relations that require sets of material parameters to be specified. We present an inverse parameter identification technique, based on statistical analyses and a particle swarm optimization algorithm, to be used in the calibration process of geomechanical models. Its application is demonstrated with typical examples from the fields of soil mechanics and engineering geology. The results for two different laboratory tests and a natural slope clearly show that particle swarms are an efficient and fast tool for finding improved parameter sets to represent the measured reference data.

1. INTRODUCTION

When compared to most other engineering tasks, geotechnical problems are often characterized by the following peculiarities. The materials involved are geological materials, that is, soils and rocks, which are inhomogeneous and consist of various phases in different states of aggregation. Initial and boundary conditions tend to be complex and heterogeneous. Furthermore, in real geotechnical field problems, the exact geometry is usually not known, with the available geometry information being limited to topographical surface data and punctual outcrops or soundings. For this reason, in geotechnics, there is always a need for a high level of simplification and abstraction. Frequently, continuum methods are used to calculate deformations in soil or rock, and the material behavior is simulated by means of constitutive models, which require a certain set of material parameters.

Normally, in geotechnical engineering, the values of these parameters are set based on the results of laboratory experiments, literature data, or even just experience values are used. The results of the calculation, a forward calculation, are then compared to measurement data obtained in the laboratory or in the field. Provided that the simplifications made and the constitutive model chosen are appropriate and provided that the performed calculation gives plausible results, parameter values are then varied by trial and error in order to reach an improved fit of the calculation results to the measured data, the reference data.

Though this is done based on the experience of the geoscientist, the procedure remains to a certain extent arbitrary or at least subjective.

In recent years, due to the availability of sufficiently fast computer hardware, there has been a growing interest in the application of inverse parameter identification strategies and optimization algorithms to geotechnical modeling in order to make this procedure automated [1–5] and thus more traceable and objective. Furthermore, this approach provides statistical information, which can be used to quantify the calibration quality of the developed geotechnical model.

Applications of optimization procedures in geotechnics were described by many authors, for example, in the calibration process of geotechnical models [1, 2], or to identify hydraulic parameters from field drainage tests [6]. Already in 1996, Ledesma et al. [7] and Gens et al. [8] applied gradient methods to a synthetic and a real example of a tunnel drift simulation. Also during the excavation of a cavern in the

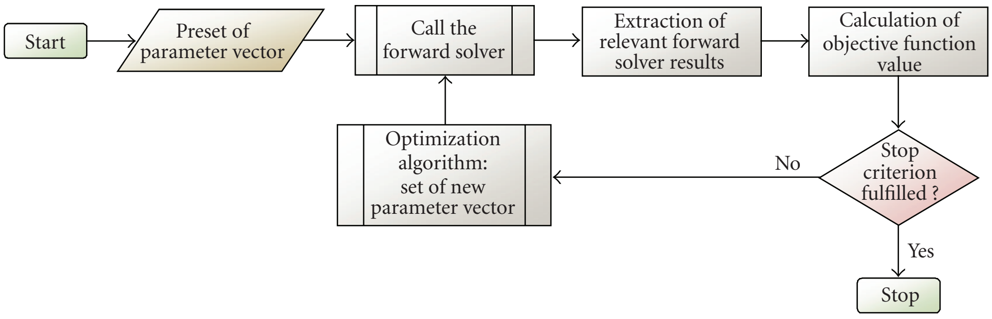

Start
Preset of parameter vector
Call the forward solver
Extraction of relevant forward solver results
Calculation of objective function value
Stop criterion fulfilled ?
No
Optimization algorithm: set of new parameter vector
Yes
Stop

Figure 1: Flowchart of the adopted iterative procedure.

Spanish Pyrenees the above mentioned group of authors was applying gradient methods for the identification of geotechnical parameters [9]. Malecot et al. [10] used inverse parameter identification techniques for analyzing pressuremeter tests and finite-element simulations of excavation problems. For the identification of soil parameters, also genetic algorithms were studied [11, 12]. Feng et al. [13] used an inverse technique for the determination of the parameters of viscoelastic constitutive models for rocks, based on genetic programming and a particle swarm optimization algorithm. In the field of geoenvironmental engineering, Finsterle [14] examined the potential use of standard optimization algorithms for the solution of aquifer remediation problems in three-phase and three-component flow and transport simulations of contamination plumes. As a different aspect of parameter identification, Cui and Sheng [15] determined the minimum parametric distance to the limit state of a strip foundation by optimizing a reliability index. In 2006, J. Meier and T. Schanzin [5] applied particle swarm optimization techniques to geotechnical field projects and laboratory tests, namely, a multistage excavation and the desaturation of a sand column.

All cited references agree on the fact that back-calculation of model parameters by means of optimization routines is possible in the field of geotechnics, if an appropriate forward calculation depending on adequately realistic model assumptions is provided, for example, Calvello and Finno [1, 2]. In this context, particle swarms represent a powerful tool for finding parameter sets that best represent the reference data, with acceptable calculation effort and time consumption.

2. WORKING SCHEME OF THE ADOPTED PARAMETER IDENTIFICATION STRATEGY

The starting point of the parameter identification strategy presented in this study is given by an ordinary geotechnical modeling-task, the so-called forward calculation. This forward problem consists of a specified geometry with given initial and boundary conditions and a material model, which requires a set of material parameters to be determined. It is generally also possible to identify geometrical parameters [4], but this issue will not be discussed in this article. For the first run of the forward calculation, the user presets the unknown parameters, for example, entering estimated values, or the preset of the parameter vector is done by a random generator within values margins specified by the user. The relevant results of the forward calculation are then read out and their deviation from a set of reference data, usually measured data, is determined by means of an objective function. This procedure is repeated many times, while at any one time, an optimization algorithm, based on the parameter combinations and the values of the objective function during the previous forward calculations, identifies an improved parameter set to be used in the next forward calculation.

This sequence of cycles, illustrated in Figure 1, is interrupted when one of the following stop criteria is fulfilled:

(i) a maximum number of runs or maximum calculation time is reached;

(ii) the deviation from the reference dataset, described by the value of the objective function, falls below a specified limit;

(iii) the deviation could not be lowered during a certain number of cycles.

Hence we use a direct approach as described by Cividini et al. [16] to solve a back-analysis problem. In an iterative procedure, the trial values of the unknown parameters are corrected by minimizing an error function. It is therefore not necessary to formulate the inverse problem itself, the desired solution is obtained by combining the results of numerous forward calculations with an optimization routine.

To quantify the deviation between the reference data and the modeling results, we chose the frequently used and relatively simple method of least squares. In this method, the objective function $f(x)$ for more than one reference dataset is defined as

$$f(x) = \sum_{g} \left[w_g f'_g(x) \right] \tag{1}$$

with

$$f'_g(x) = \frac{1}{m} \sum_{h=1}^{m} w_h \left(y_{h,g}^{\text{calc}}(x) - y_{h,g}^{\text{meas}} \right)^2. \tag{2}$$

In (1) and (2), x denotes the parameter vector to be estimated and w_g are positive weighting factors associated

```
“initialization of swarm”
create particle list
for each particle
        set initial particle position and velocity
next particle
“processing loop”
do
        “get global best particle position”
        determine best position in past of all particles
        “determine current particle positions”
        for each particle
            calculate and set new velocity based on a corresponding equation
            calculate and set new position based on a corresponding equation
        next particle
        “parallelized calculation of objective function values”
        for each particle
            start forward calculation and calculate objective function value
        next particle
        “join all calculations”
        wait for all particle threads
        “post-processing”
        for each particle
            if current objective function is less than own best in past:
                save position as own best
            end if
        next particle
loop until stop criterion is met
```

Algorithm 1: Pseudocode of the used particle swarm optimizer.

correspondingly with the error measure $f_g'(x)$. Via the weights w_g, the different series g can be scaled to the same value range and different precisions can be merged, for example, a series of measuring data possessing a higher precision is included with a higher weighting factor compared to more uncertain data. The particular numbers for the weights have to be given manually respecting the engineer's experience and they have to be specified depending on the optimization problem. The weighting factors w_h are used to provide a possibility for considering different precisions and measurement errors within one and the same data series. The dimensions of the weighting factors can be taken in the way to obtain a dimensionless objective function quantity. For minimizing the objective function, we use a particle swarm optimization algorithm described by Eberhart and Kennedy [17, 18].

A computer program developed by the first author of this paper implements this algorithm and disposes of interfaces to several commercial finite-element packages used in geotechnical engineering. A short pseudocode of the implemented PSO used in this study is presented in Algorithm 1.

3. CONCEPT FOR THE APPLICATION TO GEOTECHNICAL PROBLEMS

If the parameter identification strategy described above is to be applied to geotechnical tasks, the geotechnical model of the forward problem usually has to be adapted for its use in the optimization routine. The runtime of a single forward calculation has to be minimized in order to allow for a high number of calls. The number of calls needed depends furthermore on the number of parameters to be identified and, of course, on the used optimization algorithm.

While the reduction of calculation time demands simplification and abstraction, the model still should be sufficiently complex to reproduce the reference data with the required accuracy. Furthermore, the number of required forward calculations can be reduced by applying hypersurface approximation methods [4].

As a next step, it is essential to select the parameters to be identified and to decide on the upper and lower limits of their plausible values margins. The values of some of the parameters might be fixed with the aid of previous knowledge. These specifications must be done with care and require the experience of the geoscientist, since they influence the obtained results. However, due to the application of a particle swarm optimizer instead of, for example, a gradient method, no initial guess for the parameter set is necessary because initial positions are generated randomly within the parameter value margins.

Due to the inhomogeneities of the geological materials involved and the uncertainties related to the initial conditions and the geometrical boundary conditions, geotechnical problems tend to be underdetermined. In order to improve and ensure the efficiency of the back-analysis, it is of significant importance to check if the set of parameters to be identified may be reduced and if for the prescribed trusted zone the optimization problem is well posed. For this purpose, a statistical analysis is done based on the results of a Monte Carlo procedure including a sufficient parameter set. Figure 2 shows the principle scheme of the matrix plot used here to visualize the results of the Monte Carlo simulation. A standard mathematical tool for examining multidimensional datasets is the scatter plot matrix (see [19]), whic is included in the matrix plot presented in Figure 2, where each nondiagonal element shows the scatter plots of the respective parameters. The matrix is symmetric. Matrix element *D*-*B*, for example, may suggest that the involved parameters *B* and *D* are not independent, but strongly correlated. The diagonal matrix plot elements (*A*-*A*, ..., *D*-*D*) show plots where the value of the objective function is given over the parameter which is associated with the corresponding column. These plots are called hereafter objective function projections. If the problem is well posed, each of these plots of the objective function projections has to present one firm extreme value as it is the case in the diagram *D*-*D*. Otherwise, the respective parameter could not be identified reliably. By filtering out data points that have objective function values larger than a certain threshold level, the distribution of the remaining points gives a rough idea of the size and shape of the extreme value (solution) space. For further statistical analyses, the well-known linear 2D correlation coefficient can be calculated from the individual scatter plots. Referring to [20] in the analysis of our case studies, we consider variables with a correlation coefficient of less than 0.5 as “noncorrelated”.

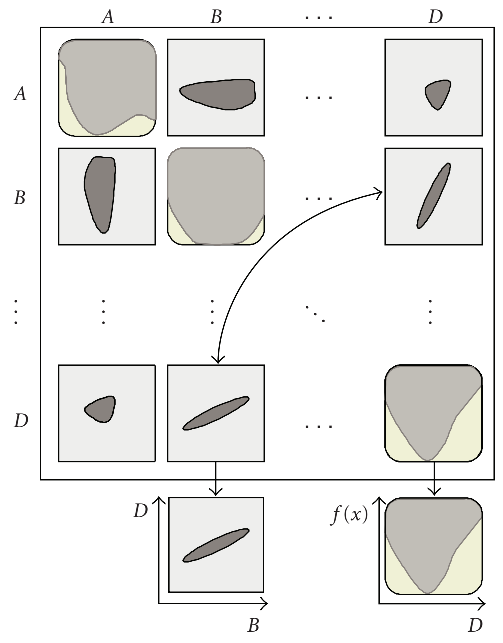


Figure 2: Statistical analyses via matrix plot, principle scheme.

Table 1: Properties of the studied soil.

Percentage of clay particles (diameter < 2 μm)	26–34%
Bulk density (g/cm^3)	1.82
Dry density (g/cm^3)	1.29
Density of grains (g/cm^3)	2.76
Porosity n	53%
Lime content	36%
Loss of ignition	6.5%
Liquid Limit W_L	0.50
Plastic Limit W_P	0.27
Water content	0.41

4. APPLICATIONS

4.1. Description of the studied geological material and the adopted constitutive model

In the following examples of applications of the presented parameter identification technique using PSO, we are modeling the mechanical behavior of a natural soil. It is of geotechnical interest because it favors the development of numerous landslides, namely, rotational soil slips, earth slides, and earthflows. The studied material is clay that results from the weathering of structurally complex geological formations, the San Cassiano formation (Kassianer Schichten), and the La Valle formation (Wengener Schichten) of the Alpine Trias. These rock formations are made up by interbedded strata of marls, tuffites, claystones, limestones, dolomites, and sandstones. Like its source rocks, the soil is characterized by a high clay and silt content. In the field, it consists of a clayey matrix with coarser components, of diameters from centimeters up to meters, floating in it without mutual support. Therefore, it is not possible to sample the material as a whole representatively. As it has been completely remolded by earthflow phenomena, no preferred orientation of the components can be observed. For this study, only the fraction smaller than 2 mm was taken, as it is considered to determine the relevant mechanical properties of the entire soil and it allows for the manufacturing of reproducible samples and test specimens. Some properties of the studied material are listed in Table 1. Unless otherwise expressly stated, all tests and classifications were carried out according to the German standard DIN.

From this description of the material, it becomes clear that its mechanical behavior is expected to be very complex and therefore it is only possible to model some important aspects of this behavior. Like many soils with a high clay and silt content, the studied material is highly compressible and exhibits a significant amount of creep deformations, thus its behavior is strongly time-dependent. As a constitutive model, we chose the soft soil creep model, which was developed by Vermeer and Neher [21] to account particularly for these phenomena. The soft soil creep model requires the following material parameters to be specified (see Table 2).

A set of three parameters $(c, \varphi \text{ and}, \psi)$ is needed to model failure according to the Mohr-Coulomb criterion. Two further parameters are used to model the amount of elastic and plastic strains and their stress dependency. The modified compression index (λ^*) represents the slope of the normal consolidation line during one-dimensional or isotropic logarithmic compression. In the same manner, the modified swelling index (κ^*) is related to the unloading or swelling line. The modified creep index (μ^*) serves as a measure to simulate the development of volumetric creep deformations with the logarithm of time.

In the modeling examples of this article, we will not take into account the development and the influence of water pressures, which would be also possible but imply a considerable increase in calculation effort; and in the case of the slope example in Section 4.4, more reference data would be needed. All forward calculations were carried out applying the finite-element method using the commercial code PLAXIS (Version 8.2, professional, update-pack 8, build 1499) and considering the effect of large deformations by means of an updated Lagrangian formulation (updated mesh analysis) [22].

4.2. *Oedometer test*

A one-dimensional compression test was conducted by MFPA Weimar (Germany) [23] in a fixed oedometer ring with an inner diameter of around 7 cm (71.45 mm) and a height of around 2 cm (20.21 mm). Drainage was allowed on the top and at the bottom of the soil sample. All load steps were applied vertically, while the sample was held radially, impeding horizontal displacements. First, the sample was preloaded with 9 kPa during two days and with 13 kPa during one day. Then the load was doubled successively, with each load step lasting 24 hours, loading the sample with 25, 50, 100, 200, 400, and 800 kPa. After that, it was unloaded at 400, 200, 100, and 50 kPa, and finally it was reloaded again with 100, 200, 400, and 800 kPa (last step took 43 hours). The displacements of the sample top were recorded continuously.

For the numerical model of the test setting, we used an axisymmetric geometrical configuration with the exact dimensions of the test specimen. In order to minimize calculation runtime, the discretization was done with two six-node triangular elements only, which is the minimum possible number, as the software offers only triangular elements. Thereby, the duration of a forward calculation could be reduced to less than one minute on an ordinary personal computer. The accuracy of the deformation results was checked by carrying out comparative analyses with finer meshes. Horizontal fixities were assigned to the lateral boundary and to the rotation axis, simulating the stiff oedometer ring and vertical fixities were attributed to the basal boundary, representing the fix filter plate at the bottom. After generating the initial stress state by applying the soil self-weight (gravity loading procedure), distributed loads were applied perpendicular to the top boundary analog to the laboratory conditions.

Three parameters of the material model (λ^*, κ^*, and μ^*) can be determined directly from the oedometer test. As the material is known to show no dilatancy, ψ can be set to 0° in all calculations. Laboratory data from shear tests on similar soil samples reported by Panizza et al. [24] is shown in Table 3.

Table 2: Parameters of the soft soil creep model.

Parameter	Description	(Unit)
c	Effective cohesion	(kPa)
φ	Effective friction angle	(°)
ψ	Dilatancy angle	(°)
λ^*	Modified compression index	(dimensionless)
κ^*	Modified swelling index	(dimensionless)
μ^*	Modified creep index	(dimensionless)

Table 3: Shear-test data reported by Panizza et al. 2006 [24].

	Effective friction angle φ' (°)	Effective cohesion c' (kPa)
Direct shear tests	18	20
	18	10
	20	49
	18	39
	16	49
	14	69
	18	25
	20	20
Triaxial tests	19	7
	28	14

We averaged these values, giving double weight to the triaxial test data, which we assumed to be more precise, coming out with an average friction angle of 20° and an average cohesion of 27 kPa. The set of experimental parameter values is shown by Table 4 and the results of a forward calculation using these parameters are presented in Figure 3 comparing them with the reference data of the oedometer test.

The graph shows that the deformations are underestimated by the simulation. In order to test the ability of the PSO algorithm to find good parameter combinations, wide search areas were chosen for the five parameters. A statistical analysis (see Section 3) comprising 2000 calls of the forward calculation was then performed varying these parameters. Their value margins are displayed in Table 5.

The scatter plot matrix of all data points with objective function values lower than 10^{-7} is given in Figure 4. For the parameters λ^*, κ^*, and μ^*, logarithmized margins of the search intervals were used in order to avoid overrepresentation of high parameter values. Furthermore, a parameter constraint was prescribed, demanding for $\lambda^* > \kappa^*$, which has to apply for all materials. The objective function projections of c, κ^*, and λ^* indicate that the data points showing good model fits seem to concentrate in quantifiable value ranges of these parameters, whereas μ^* and φ cannot be identified. The modified compression index (λ^*) and the cohesion (c) appear to be correlated (correlation coefficient of 0.88), to a lesser extent, this holds true also for λ^* and κ^* (correlation

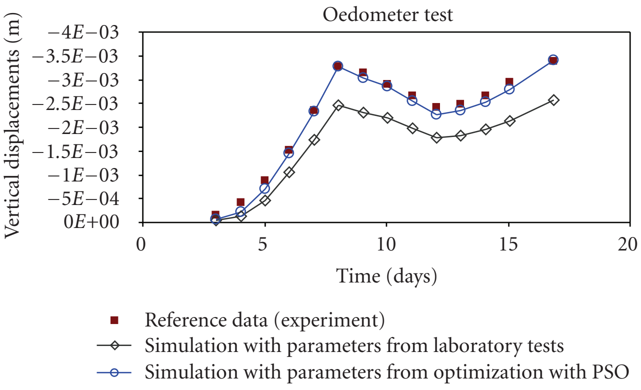


Figure 3: Oedometer test calculation results versus reference data.

Table 4: Parameters obtained from experiments and identified parameters using PSO.

	Experimental parameter values	Identified values PSO
λ^*	0.064	0.082
κ^*	0.035	0.051
μ^*	1.46E–03	—
φ (°)	20	—
c (kPa)	27	25
ψ (°)	0	—

Table 5: Search intervals for the parameters of the oedometer test.

	Maximum	Minimum	ln (max.)	ln (min.)
φ	30	8	—	—
c	100	0.001	—	—
λ^*	1	0.002	0.00	−6.21
κ^*	0.5	0.001	−0.69	−6.91
μ^*	0.75	0.00001	−0.29	−11.51

coefficient of 0.64). According to these findings, λ^*, κ^*, and c were selected for the optimization procedure. The friction angle (φ) and the modified creep index (μ^*) were fixed on their experimental values.

After 159 cycles (1590 calls) the particle swarm optimizer had reduced the deviation to $5*10^{-9}$, which was found to be a sufficiently low value to stop the optimization routine. The identified best parameter set and the corresponding calculation results can be seen also in Table 4 and Figure 3. It becomes clear that the identified parameter set represents the measurement data much better than the available laboratory parameters.

4.3. *Isotropic compression test*

An isotropic compression test was performed in the triaxial apparatus on a cylindrical soil sample with a diameter of 5 cm (49.88 mm) and a height of 10 cm (98.55 mm). Drainage was allowed on top and at the bottom of the specimen. All load steps were performed isotropically, applying a hydrostatic cell pressure. The sample was preloaded at 30 kPa for three hours and, after that, at 50 kPa for 17 hours. Then it was gradually loaded to 800 kPa in one hour, increasing the load by steps of 100 kPa. After reaching this target load, the stress level was left constant for 3 weeks. Top displacements and volume change of the sample were recorded during the whole test. A reference dataset for the horizontal displacements was calculated from the vertical displacements and the volume change, assuming the shape of the specimen to remain exactly cylindrical until the end of the test.

For the numerical model of the test setting, an axisymmetric geometrical configuration with the exact dimensions of the test specimen was used. Again, the model was discretized only with two six-node triangular elements, to save calculation time. The accuracy of the deformation results was checked by carrying out several comparative analyses with finer meshes. Horizontal fixities were assigned to the rotation axis. Vertical fixities were attributed to the basal boundary, representing the fix filter plate at the bottom. After generating the initial stress state by applying the soil self-weight (gravity loading procedure), two independent and identical distributed loads were applied, one perpendicular to the upper boundary (vertically) and the other one perpendicular to the lateral boundary (radially). Loading was carried out the same way as in the laboratory, but instead of the stepwise application of the 800 kPa target load, this load was applied directly after the 50 kPa load step. For this reason, the 50 kPa load step in the model was prolonged in such a manner that the integral of the load as a function of time equals the test conditions.

In the example of the isotropic compression test, except for the relatively short phases before reaching the target load, only one load step (800 kPa) is applied. Therefore, of the model parameters only μ^* can be determined directly from the test. This value $(1.3*10^{-3})$ is very similar to the one obtained from the oedometer test. Figure 5 shows the results of a forward calculation using this value together with the laboratory values of Section 4.2.

Also in this example, the deformations are underestimated by the simulation. A statistical analysis with 1820 calls was carried out varying the parameters c (cohesion), λ^* (modified compression index), and μ^* (modified creep index) within the boundaries given in Table 7, logarithmic values were used for the search intervals of the latter two parameters.

As the modeled test contains no unloading phases, it makes no sense to identify the modified swelling index κ^*. Its value was therefore linked to the value of λ^* by multiplying this parameter by 0.5, which is the typical κ^*/λ^* ratio, we observed in our laboratory tests performed on this material and similar materials.

The results of the statistical tests presented in Figure 6 suggest that good fits can be obtained for cohesion values between 20 kPa and 90 kPa, but apart from this, the cohesion value seems to have no influence on the quality of the model calibration. Whereas for the modified compression

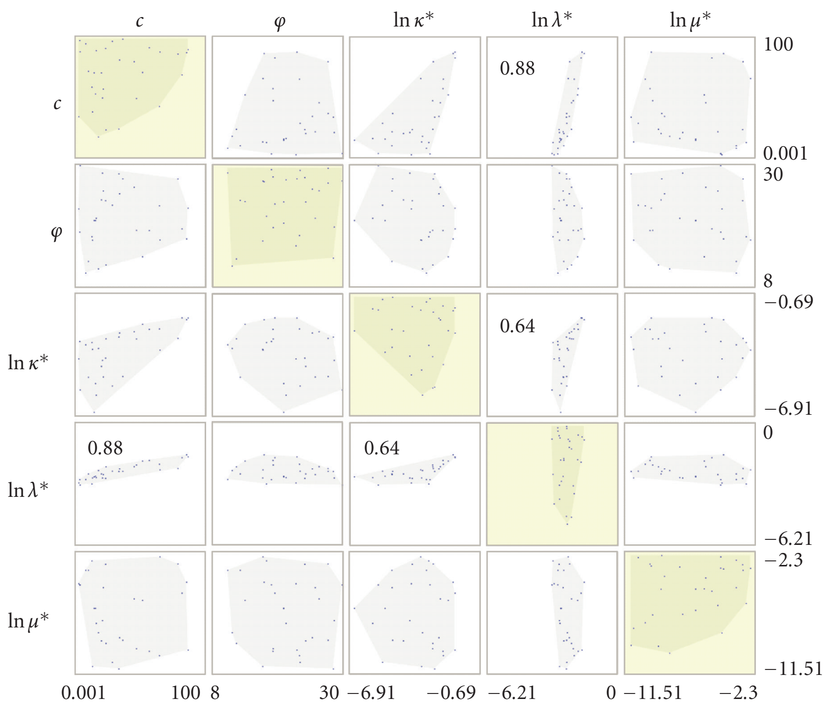
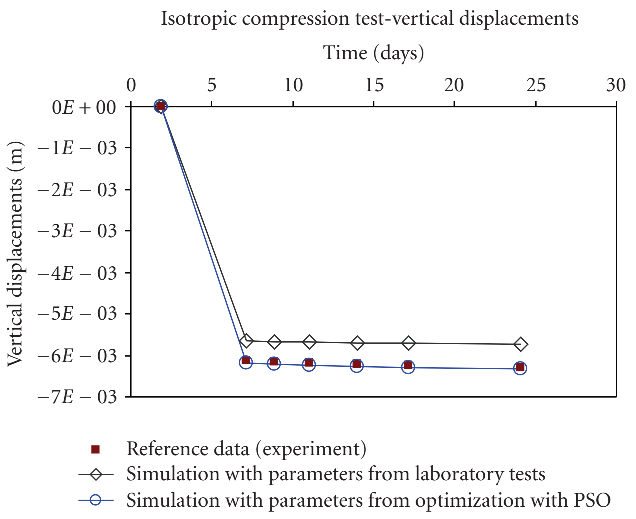
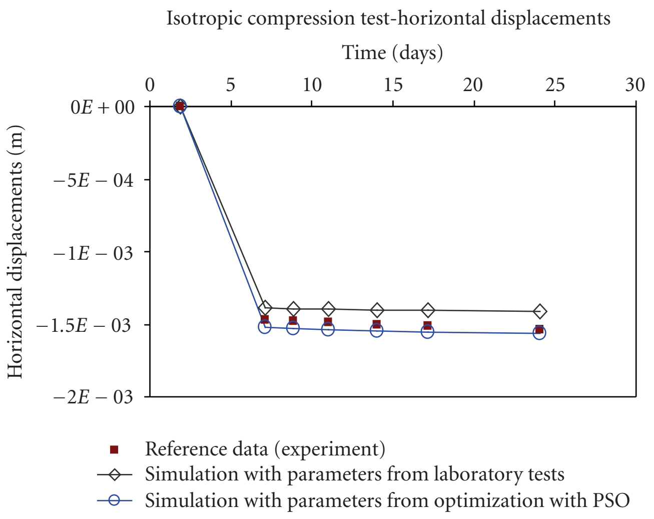


Figure 4: Scatter plot matrix for the oedometer test.



(a)



(b)

Figure 5: Isotropic compression test calculation results versus reference data.

index (λ^*) and the modified creep index (μ^*), the objective function projections suggest a preferred value range for the data points with low deviation values.

Furthermore, it can be concluded that λ^* and c might be quite closely related to each other (correlation coefficient 0.96), this means, for a given λ^*, an appropriated cohesion value could be computed by the equation given also in Figure 6. This linear relationship seems to be valid for cohesion values between 20 and 90 kPa and λ^* values between 0.064 and 0.165.

Therefore, only the parameters λ^* and μ^* were selected for the optimization procedure via PSO. We stopped this procedure after 500 calls (50 cycles). The identified values are shown in Table 6. In Figure 5, the calculation results are

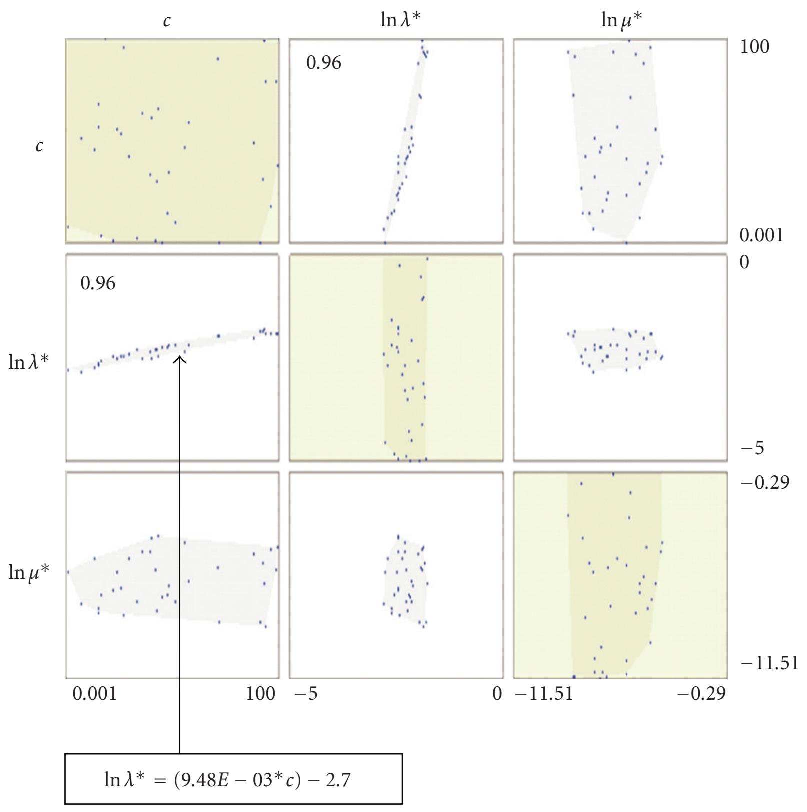


FIGURE 6: Scatter plot matrix for the isotropic compression test.

TABLE 6: Parameters from laboratory tests and identified parameters using PSO.

	Experimental values	Identified values (PSO)
λ^*	0.064	0.089
κ^*	0.035	—
μ^*	1.30*E*–03	2.33*E*–03
φ (°)	20	—
c (kPa)	27	—
ψ (°)	0	—

TABLE 7: Isotropic compression test search intervals for the varied parameters.

	Maximum	Minimum	ln (max.)	ln (min.)
c (kPa)	100	0.001	—	—
λ^*	1.00	6.74*E*–03	0.00	−5.00
μ^*	0.75	0.00001	−0.29	−11.51

compared to the reference dataset and the results obtained by using only laboratory data.

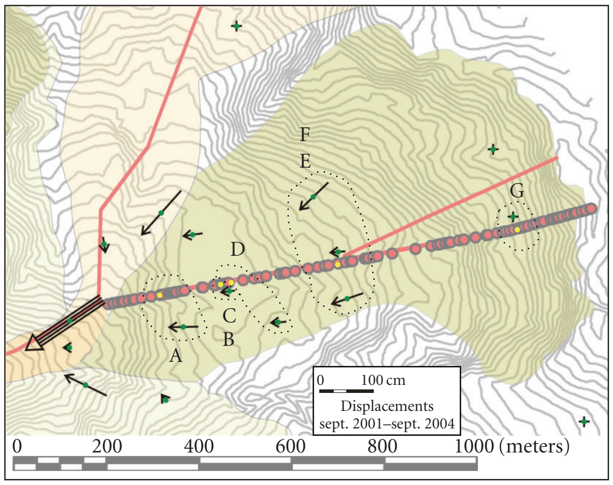
Like for the oedometer test, an improved parameter set could be found also for the isotropic compression test. It can be observed that the results are much better for the vertical displacements, although a weighting factor of one had been assigned to both datasets.

This may be due to the fact that the precision of the reference data is lower for the horizontal displacements, since the volume change of the sample could not be measured with the same accuracy as the top displacements. In addition to this, small inhomogeneities of the material or a small frictional resistance at the sample top could have caused a slight distortion of the cylindrical shape of the specimen which was assumed to calculate the horizontal displacements.

4.4. *Deformations along a shear zone in a natural slope*

4.4.1. *Analyzed section and reference data*

The presented parameter identification technique was also applied to the 2D-model of a natural slope which is located in the municipality of Corvara in the Dolomites (Italy). It shows continuous creep deformations at the basis of a 20–40 m thick soil cover consisting mainly of the above studied material and very similar materials. As slopes of this type are known to show potential acceleration phases that can



● Field point
● Model point

Figure 7: Site map with the location of the section and the GPS points.

endanger human settlements, a detailed study of the geology and geomorphology as well as comprehensive monitoring were carried out by the University of Modena and Reggio Emilia and by the National Research Council's Group for Hydrogeological Catastrophes Defense (CNR-GNDCI) [25] by order of the Autonomous Province of Bolzano/South-Tyrol [24].

In this context, the displacements of several surface points were observed regularly by means of global positioning system (GPS) measurements. In our study, a subarea of the slope was modeled along a representative 2D section. A site plan with the location of the section and the GPS points is given in Figure 7, whereas Figure 8 depicts a field survey of this section and shows the abstracted geometry model.

4.4.2. *Geotechnical model*

The geometry was determined using all the information available, that is, a core drilling near section-point B, the local geomorphology, refraction seismics, and direct current resistivity (DC-resistivity) [24]. As exposed in Figure 9, the vertical profile of the slope was divided into three layers interpreting various inclinometer profiles reported by Corsini et al. [25]; the illustrated one is located near section-point B. The uppermost layer, the soft soil cover, which is showing little internal deformations, was only considered in the form of its weight acting on the intermediate layer, the shear zone. Therefore, the displacement vectors of section-points C and F are presumed to be equal to those of section-points B and E, respectively. The shear zone is assumed to be a thin, soft, and highly plastic layer, exhibiting a pronounced time dependency in its mechanical behavior. The third layer is given by the underlying weathered bedrock, which is supposed to be stable. The earth pressure at the foot of the slope was assumed to be slightly lower than the earth pressure at rest; this means that there is no support obtained from the soil layer further downslope. A description of the assumptions made for the foot load is shown in Figure 10.

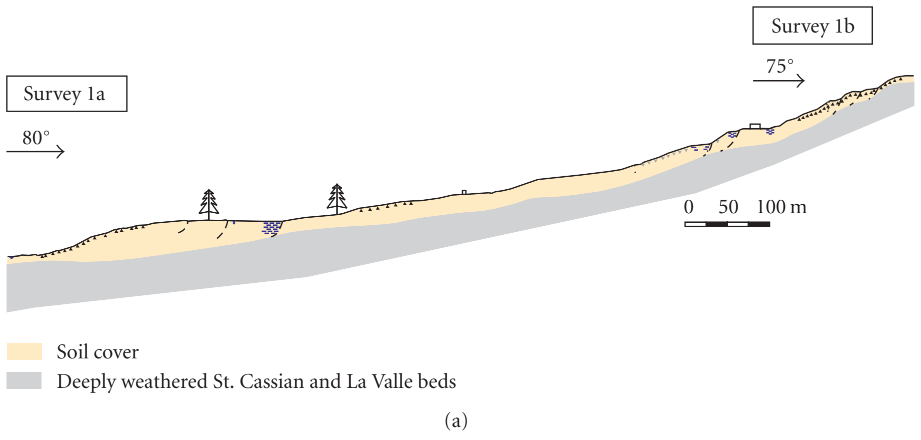
The actual main detachment zone is modeled as an open crack. No tensile forces are acting across it onto the downslope section of the sliding body, which is moving as a whole. Around section-point A, the soil body below the secondary shear zone is assumed to move at the same velocity as point B. The material properties of the secondary shear zone, which is not subject of this study, were fixed to comply with this criterion. At present, the displacement rates observed along the slope are more or less constant, being superposed only by seasonal variations attributed to fluctuations of the groundwater conditions which are not modeled in this study. Therefore, our geotechnical model features displacement rates remaining constant with time.

4.4.3. *Numerical model*

Figure 11 shows the characteristics of the numerical model of the studied slope. A plane strain geometrical configuration with the real dimensions of the slope was used. The model was discretized with 1070 triangular six-node elements. To save calculation time, the number of elements was reduced by modeling only the uppermost 20 m of the bedrock layer. The upper and the lower layers were meshed with the automatic meshing procedure of the software and using a very coarse setting. A linear elastic material model was assigned to them. Finally, one forward calculation took approximately three minutes on an ordinary personal computer. The intermediate layer was meshed manually by predefining geometry points in order to assure a sufficiently fine mesh and a suitable orientation of the triangles in order to go against an excessive distortion of the element shapes by the calculated deformations. For the same reason, the updated mesh option was only used in the last three years of the simulation (which are compared to the reference dataset). The detachment zones were modeled by means of interfaces on both sides of their geometry lines. The interfaces were modeled with a Mohr-Coulomb material model, with negligible cohesion, the same friction angle as the basal shear zone and with a constant reference stiffness in the order of magnitude of the shear zone stiffness. The accuracy of the calculated deformation results was checked by carrying out several comparative analyses with finer meshes and also with a horizontal basal boundary. Horizontal fixities were assigned to the lateral edges of the model, which extends over a total length of 1080 m. Vertical fixities were attributed to the basal boundary, representing the stable bedrock.

4.4.4. *Calculation phases*

In the first calculation phase, all three layers are made up by the bedrock material. An initial stress state is generated by applying the self-weight of this material (gravity loading procedure). In the second calculation phase, the two upper layers are replaced by the weaker material of the soil cover. The third calculation phase marks the starting point of the



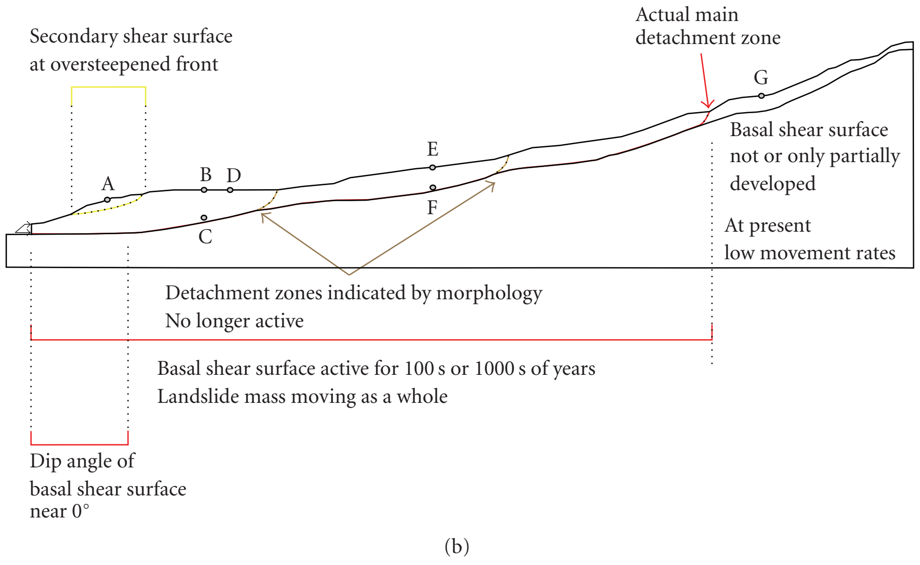

Secondary shear surface at oversteepened front
Actual main detachment zone
G
E
A
B D
F
C
Basal shear surface not or only partially developed
At present low movement rates
Detachment zones indicated by morphology
No longer active
Basal shear surface active for 100 s or 1000 s of years
Landslide mass moving as a whole
Dip angle of basal shear surface near 0°
(b)

FIGURE 8: Field survey along the section and derived geometry model.

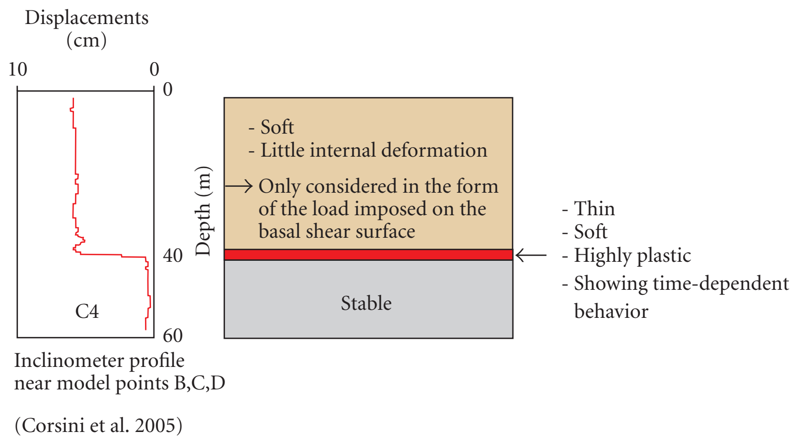
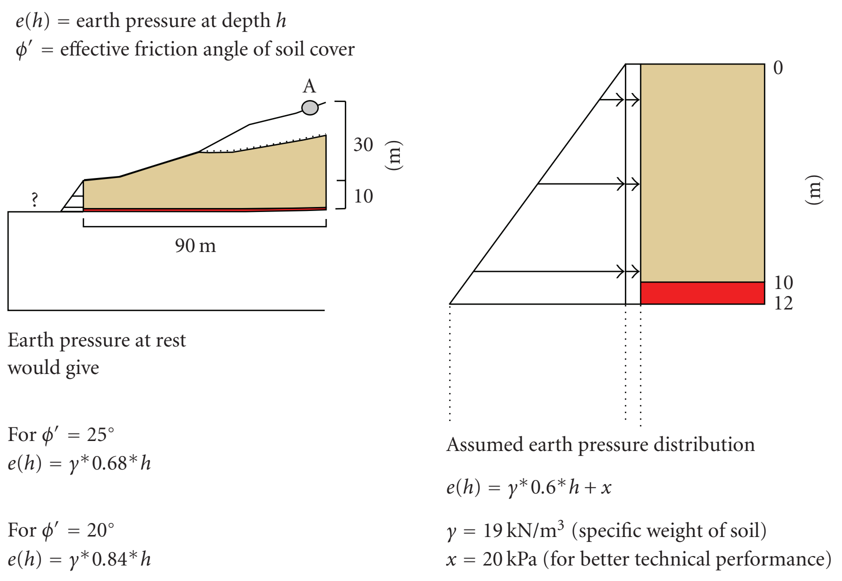

Displacements (cm)
10
0
0
40
60
C4
Depth (m)
- Soft
- Little internal deformation
Only considered in the form of the load imposed on the basal shear surface
Stable
- Thin
- Soft
- Highly plastic
- Showing time-dependent behavior
Inclinometer profile near model points B,C,D
(Corsini et al. 2005)

FIGURE 9: Interpretation of the inclinometer data from Corsini et al. 2005 [25].



$e(h)$ = earth pressure at depth h

ϕ' = effective friction angle of soil cover

Earth pressure at rest would give

For $\phi' = 25°$
$e(h) = \gamma * 0.68 * h$

For $\phi' = 20°$
$e(h) = \gamma * 0.84 * h$

Assumed earth pressure distribution

$e(h) = \gamma * 0.6 * h + x$

$\gamma = 19\,\mathrm{kN/m^3}$ (specific weight of soil)
$x = 20\,\mathrm{kPa}$ (for better technical performance)

Figure 10: Earth pressure assumptions made for the foot of the slope.

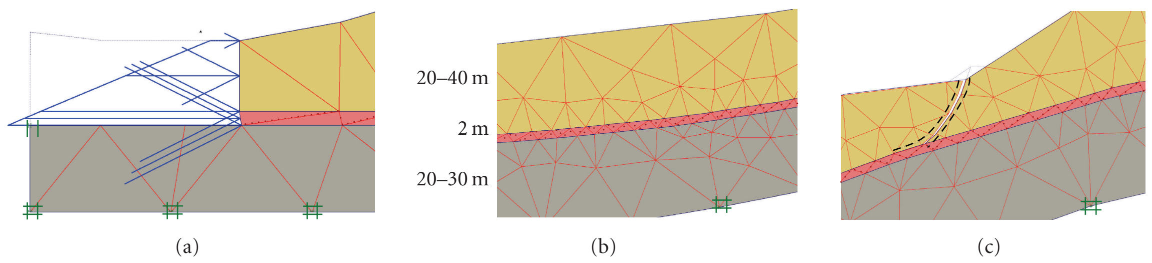


Figure 11: Discretization of the slope model: foot zone, vertical profile, and detachment zone.

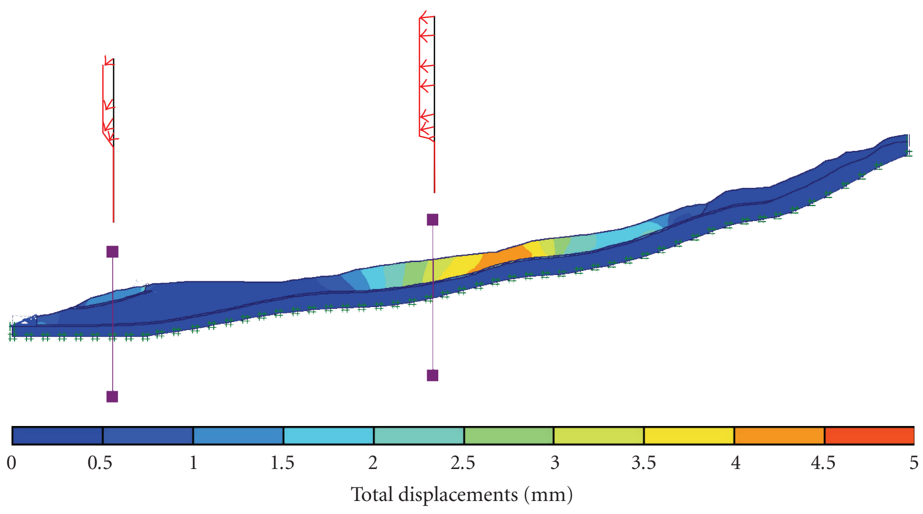


Figure 12: Results of a forward calculation using laboratory values of the parameters.

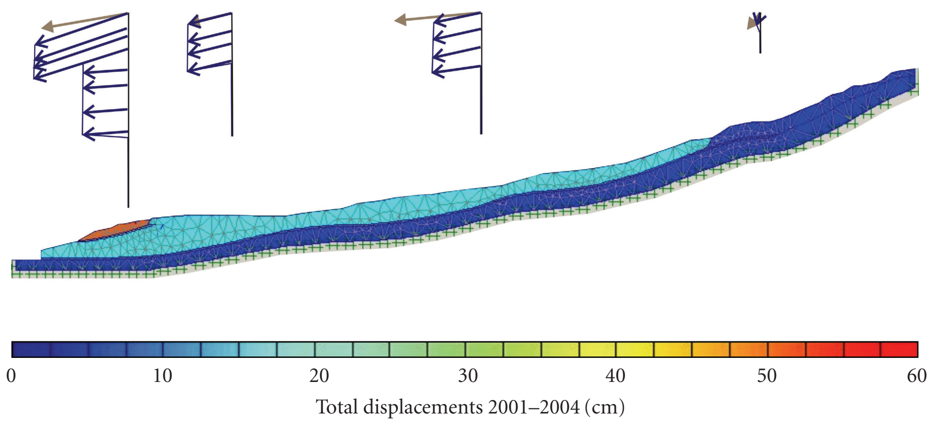


Figure 13: Calibrated slope model—comparison of modeled displacement vectors (blue) with reference data derived from the GPS measurements (brown).

slope instability in the model. The shear zone material with time-dependent mechanical behavior is inserted and the horizontal load at the foot is set. After that, the model is left creeping with unchanged boundary conditions during a period of 33 years. As the loading history of the shear zone material is unknown, this time period had to be chosen arbitrarily to reach constant displacement rates as they are presently observed along the natural slope.

4.4.5. *Results of initial model using parameters derived from experiments*

In a first trial forward calculation, for λ^*, κ^*, and μ^*, the parameters calculated from the laboratory experiments were used as input values. As the deformations along the shear zone are known to persist since hundreds or thousands of years [26], the shear strength of this zone has decreased to a residual value that is characteristic for the soils originated by the weathering of the San Cassiano and La Valle beds outcropping in the whole slope area. Therefore, cohesion was assumed to be negligible (0.01 kPa) and a friction angle of 10° was adopted, according to the average slope inclination observed in nearby areas which were formed since the Late Glacial by the studied processes (earth slides and earthflows) and covered by comparable soil covers [27]. The stiffness of the uppermost layer was set equal to the stiffness modulus observed in the oedometer test during unloading and reloading between the load steps 400 kPa and 800 kPa. For Poisson's ratio of this layer, we used 0.35, a value that is considered to be characteristic of clayey soils. The experimental parameter values are shown in Table 8 and the deformations calculated on their basis for the last three years of the creep phase are presented by Figure 12. The latter are only in the range of millimeters, and thus not representing the actual situation in the field, where between September 2001 and September 2004, displacements from several centimeters to several decimeters were measured.

Table 8: Laboratory values of the parameters used for the slope example.

	Experimental parameter values
λ^*	0.064
κ^*	0.035
μ^*	0.00146
c (kPa)	0.01
φ (°)	10
G (kPa)	5560
ν	0.35

4.4.6. *Results of statistical analysis and optimization procedure*

A statistical analysis was carried out (which will not be reported in detail here). One interesting finding of this analysis was that the friction angle and the modified creep index appeared to be closely correlated (coefficient of 0.92). The parameters λ^*, κ^*, and μ^*; the friction angle; and the stiffness of the uppermost layer (represented by its shear modulus G) were chosen for the optimization procedure during which they were varied within the intervals specified in Table 9.

After 82 cycles, each of them consisting of 10 forward calculations, the procedure was stopped because, from then on, the deviation could no longer be reduced significantly. The resulting parameter set is also given in Table 9. Figure 13 depicts the calculated deformations using the identified parameter combination, together with the displacement vectors of the GPS measurement points.

It can be observed that the identified parameter set is able to reproduce the field measurements qualitatively. Because of the simplifications made in the model, no exact fit of the displacement vectors is possible. The presented back analysis procedure gives one of a number of possible approximate solutions to the geotechnical problem and the result returned

Table 9: Search intervals and identified parameters for the slope example.

Parameter	Fixed parameters	Varied parameters		Identified values (PSO)
		Maximum	Minimum	
ν	0.35	—	—	—
G (kPa)	—	20000	200	5160
c (kPa)	0.01	—	—	—
ψ ($^\circ$)	0	—	—	—
κ^*	—	1	0.005	0.60
λ^*	—	2	0.01	1.42
μ^*	—	1.5	0.001	0.145
φ ($^\circ$)	—	16	8	10.7

by the particle swarm optimizer can be seen as a parameter set that best represents the reference data.

5. CONCLUSIONS

A back analysis procedure for the identification of material parameters of constitutive models applied to geotechnical problems was presented. This procedure represents a direct approach based on the method of least squares, correlation analyses, and a particle swarm optimization algorithm. The applicability and suitability of the technique was demonstrated by means of three examples from the fields of soil mechanics and engineering geology. The studied material was a natural soil. Besides being way more objective and less arbitrary than the conventional trial and error procedure, the outlined method provides valuable information on the quality of the model calibration, the uniqueness of an obtained solution, or the determinateness of the problem. In all three examples, the particle swarm optimizer was able to identify an improved parameter set after a justifiable amount of forward calculations. Further research should also concentrate on the identification of the geometrical parameters of geotechnical problems.

ACKNOWLEDGMENTS

The German Academic Exchange Service (DAAD) and the Association of the Rectors of the Italian Universities (CRUI) are acknowledged for funding traveling expenses through a VIGONI exchange project. The second author acknowledges the support by the Konrad-Adenauer-Foundation via a postgraduate scholarship. The work of the first author was funded by the German Research Foundation (DFG) via the projects SCHA 675/7-2 and SCHA 675/11-2 "Geomechanical modeling of large mountainous slopes".

REFERENCES

[1] M. Calvello and R. J. Finno, "Calibration of soil models by inverse analysis," in *Numerical Models in Geomechanics NUMOG VIII*, G. Pande and S. Pietruszczak, Eds., pp. 107–116, Balkema, Rotterdam, The Netherlands, 2002.

[2] M. Calvello and R. J. Finno, "Selecting parameters to optimize in model calibration by inverse analysis," *Computers and Geotechnics*, vol. 31, no. 5, pp. 411–425, 2004.

[3] J. Carrera, A. Alcolea, A. Medina, J. Hidalgo, and L. Slooten, "Inverse problem in hydrogeology," *Hydrogeological Journal*, vol. 13, no. 1, pp. 206–222, 2005.

[4] J. Meier, S. Rudolph, and T. Schanz, "Effektiver Algorithmus zur Lösung von inversen Aufgabenstellungen—Anwendung in der Geomechanik," *Bautechnik*, vol. 83, no. 7, pp. 470–481, 2006.

[5] T. Schanz, M. M. Zimmerer, M. Datcheva, and J. Meier, "Identification of constitutive parameters for numerical models via inverse approach," *Felsbau*, vol. 24, no. 2, pp. 11–21, 2006.

[6] Z. F. Zhang, A. L. Ward, and G. W. Gee, "Estimating soil hydraulic parameters of a field drainage experiment using inverse techniques," *Vadose Zone Journal*, vol. 2, no. 2, pp. 201–211, 2003.

[7] A. Ledesma, A. Gens, and E. E. Alonso, "Estimation of parameters in geotechnical backanalysis—I. Maximum likelihood approach," *Computers and Geotechnics*, vol. 18, no. 1, pp. 1–27, 1996.

[8] A. Gens, A. Ledesma, and E. E. Alonso, "Estimation of parameters in geotechnical backanalysis—II. Application to a tunnel excavation problem," *Computers and Geotechnics*, vol. 18, no. 1, pp. 29–46, 1996.

[9] A. Ledesma, A. Gens, and E. E. Alonso, "Parameter and variance estimation in geotechnical backanalysis using prior information," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 20, no. 2, pp. 119–141, 1996.

[10] Y. Malecot, E. Flavigny, and M. Boulon, "Inverse analysis of soil parameters for finite element simulation of geotechnical structures: pressuremeter test and excavation problem," in *Proceedings of the Symposium on Geotechnical Innovations*, R. B. J. Brinkgreve, H. Schad, H. f. Schweiger, and E. Willand, Eds., pp. 659–675, Verlag Glückauf, Essen, Germany, 2004.

[11] S. Levasseur, Y. Malecot, M. Boulon, and E. Flavigny, "Soil parameter identification using a genetic algorithm," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 32, no. 2, pp. 189–213, 2007.

[12] S. Levasseur, Y. Malecot, M. Boulon, and E. Flavigny, "Soil parameter identification from in situ measurements using a genetic algorithm and a principle component analysis," in *Proceedings of the 10th International Symposium on Numerical Models in Geomechanics (NUMOG '07)*, Rhodes, Greece, April 2007.

[13] X.-T. Feng, B.-R. Chen, C. Yang, H. Zhou, and X. Ding, "Identification of visco-elastic models for rocks using genetic programming coupled with the modified particle swarm optimization algorithm," *International Journal of Rock Mechanics and Mining Sciences*, vol. 43, no. 5, pp. 789–801, 2006.

[14] S. Finsterle, "Demonstration of optimization techniques for groundwater plume remediation using iTOUGH2," *Environmental Modelling & Software*, vol. 21, no. 5, pp. 665–680, 2006.

[15] L. Cui and D. Sheng, "Genetic algorithms in probabilistic finite element analysis of geotechnical problems," *Computers and Geotechnics*, vol. 32, no. 8, pp. 555–563, 2005.

[16] A. Cividini, L. Jurina, and G. Gioda, "Some aspects of 'characterization' problems in geomechanics," *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, vol. 18, no. 6, pp. 487–503, 1981.

[17] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[18] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, IEEE Press, Piscataway, NJ, USA, November-December 1995.

[19] B. F. J. Manly, *Multivariate Statistical Methods: A Primer*, Chapman & Hall/CRC, Boca Raton, Fla, USA, 3rd edition, 1944.

[20] J. Will, D. Roos, J. Riedel, and C. Bucher, "Robustness analysis in stochastic structural mechanics," in *NAFEMS Seminar: Use of Stochastics in FEM Analyses*, Wiesbaden, Germany, May 2003.

[21] P. A. Vermeer and H. P. Neher, "A soft soil model that accounts for creep," in *Proceedings of the International Symposium. Beyond 2000 in Computational Geotechnics. 10 Years of PLAXIS Inetrnational*, pp. 55–58, Balkema, Amsterdam, The Netherlands, March 1999.

[22] K.-J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood-Cliffs, NJ, USA, 1982.

[23] K. Lemke, H. Lorenz, A. Klimitsch, J. Koeditz, T. Schaefer, and K. J. Witt, "Results of laboratory study on earthflow materials," Tech. Rep., Material Research and Testing Institute MFPA, Weimar, Germany, 2006.

[24] M. Panizza, S. Silvano, A. Corsini, et al., "Definizione della pericolosità e di possibili interventi di mitigazione della frana di Corvara in Badia. Provincia Autonoma di Bolzano—Alto Adige," Autonome Provinz Bozen—Südtirol, http://www.provincia.bz.it/opere-idrauliche/attivita6_i.htm, 2006.

[25] A. Corsini, A. Pasuto, M. Soldati, and A. Zannoni, "Field monitoring of the Corvara landslide (Dolomites, Italy) and its relevance for hazard assessment," *Geomorphology*, vol. 66, no. 1–4, pp. 149–165, 2005.

[26] M. Soldati, A. Corsini, and A. Pasuto, "Landslides and climate change in the Italian Dolomites since the Late glacial," *Catena*, vol. 55, no. 2, pp. 141–161, 2004.

[27] A. Corsini, "L'influenza dei fenomeni franosi sull'evoluzione geomorfologica post-glaciale dell'Alta Val Badia e della Valparola (Dolomiti)," Ph.D. thesis, University of Bologna, Bologna, Italy, 2000.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 728929, 10 pages
doi:10.1155/2008/728929

Review Article

Particle Swarm Optimization for Antenna Designs in Engineering Electromagnetics

Nanbo Jin and Yahya Rahmat-Samii

Department of Electrical Engineering, University of California, Los Angeles, CA 90095-1594, USA

Correspondence should be addressed to Nanbo Jin, jnnb@ee.ucla.edu

Received 17 July 2007; Accepted 17 December 2007

Recommended by Riccardo Poli

This paper presents recent advances in applying particle swarm optimization (PSO) to antenna designs in engineering electromagnetics. By linking the PSO kernel with external electromagnetic (EM) analyzers, the algorithm has the flexibility to handle both real and binary variables, as well as multiobjective problems with more than one optimization goal. Three examples, including the designs of a dual-band patch antenna, an artificial ground plane of a surface wave antenna, and an aperiodic antenna array, are presented. Both simulation and measurement results are provided to illustrate the effectiveness of applying the swarm intelligence to design antennas with desired frequency response and radiation characteristics for practical EM applications.

1. INTRODUCTION

As access to previously unimaginable computational resources has become commonplace, many aspects of electromagnetic (EM) design have undergone titanic shifts. In particular, the fast and accurate simulation of an antenna design on a PC or parallel platforms has opened the door for stochastic optimizers to augment design processes in a large variety of engineering EM problems. As a novel evolutionary algorithm proposed in mid 1990s [1–3], particle swarm optimization (PSO) has been introduced into the EM community by one of the authors [4, 5], and its applications have received enormous attention in recent years. Unlike genetic algorithms (GAs) [6, 7], which rely on Darwin's theory of natural selection and the competition between individual chromosomes, the swarm intelligence in the nature is modelled by fundamental Newtonian mechanics in PSO for optimization purposes. This corporative scheme manifests PSO the concise formulation, the ease in implementation, and many distinct features in different types of optimizations.

In this paper, representative examples of PSO-optimized antennas developed at UCLA antenna laboratory are collected from the authors' previous publications and current research activities, in order to present the recent progress of applying swarm intelligence in practical engineering EM problems. Basic concepts in antenna design problems are introduced in Section 2, with unique advantages of applying PSO in these problems discussed. The implementation of a PSO engine for antenna optimizations is briefly described in Section 3, and three concrete design examples are presented in the following sections. The useful design guidelines provided by PSO are validated by both simulation and measurement results. The paper is summarized in Section 7.

2. ANTENNA DESIGN AS AN OPTIMIZATION PROBLEM: WHY PSO?

In transmitting or receiving systems, an antenna is a transducer of guided EM waves into propagating waves. As shown in Figure 1, the guided wave is injected into an antenna, which generates equivalent electric current $\vec{J}_{\text{eq}}$ and magnetic current $\vec{K}_{\text{eq}}$ over its enclosure. The specific configuration of antenna is not necessarily the horn as illustrated in Figure 1, and it may belong to other paradigms such as patch antennas, wire antennas, reflector antennas, or antenna arrays which assemble multiple antenna elements, and so forth. The radiation pattern of an antenna is obtained by Fourier transforming $\vec{J}_{\text{eq}}$ and $\vec{K}_{\text{eq}}$ from the spatial do-

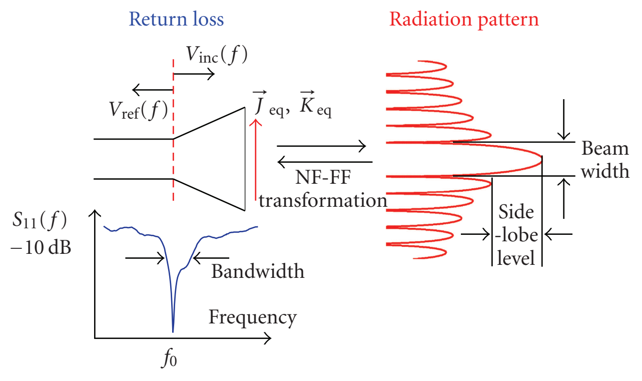


Figure 1: An antenna operates as a transducer of guided waves into propagating waves. The syntheses of frequency-dependent return loss and angular-dependant radiation pattern are formulated as optimization problems.

main into the angular domain. In general, the figures of merit of an antenna include the following.

Return loss

Usually the incident power is partially reflected due to the imperfect matching between the antenna and the front-end circuits. The return loss is measured by the reflection coefficient S_{11}, which is the ratio between the amplitudes of reflected wave and incident wave at the feeding point:

$$S_{11}(f) = 20\log_{10}\frac{V_{\mathrm{inc}}(f)}{V_{\mathrm{ref}}(f)}\ (\mathrm{dB}), \tag{1}$$

where S_{11}, V_{inc}, and V_{ref} are all frequency-dependent. A standard definition of operating bandwidth is the frequency range in which $S_{11} < -10$ dB with over 90% incident power delivered to the antenna.

Radiation pattern

The radiation characteristics of an antenna, that is, the beamwidth, the sidelobe level (SLL), the directivity, and so forth, need to be synthesized to concentrate radiated power into desired directions.

Designing an antenna at a certain operating frequency f_0 (or within a bandwidth from f_L to f_H) is inherently an optimization problem. In particular, by optimizing the antenna configuration, a desired return loss can be obtained by minimizing $S_{11}(f_0)$, and a desired radiation pattern can be achieved by minimizing the difference between the actual antenna pattern and the desired pattern. The fitness functions may have different forms according to the specific application, while most antenna design problems can be categorized into these two scenarios. The evaluations of S_{11} and radiation pattern are performed by solving Maxwell equations under different boundary conditions. As a stochastic global optimizer, PSO is a good candidate to address the significant nonlinearity and multimodal effect induced by the full-wave analysis.

In antenna optimizations, the coordinates of each point in the solution space are mapped into a candidate antenna configuration. This mapping may occur from either a continuous or a discrete space to the actual design space. In many design problems, the basic antenna configuration is determined by the designer's a priori EM knowledge, while specific geometrical parameters of the antenna need to be fine-tuned to achieve the desired performance, which is a continuous optimization. In contrast, if the operating scheme of a desired antenna is relatively unknown, the design space is typically discretized into pixels and the basic antenna topology needs to be explored by optimizing a binary string to fill in the pixels. Therefore, another advantage of applying PSO to antenna designs arises from the ease and flexibility in implementing PSO for both real and binary variables. Since the only major difference between real-number and binary PSO algorithms lies in their position updating equations, real and binary variables can be even hybridized to represent a candidate design [8].

The recent popularity of PSO in antenna designs is also attributed to its capability to efficiently handle multiple design goals. In single-objective optimizations with conventional weighted aggregation (CWA), fitness functions related to different design goals are often weighted and summed. However, the weighting coefficients need to be carefully selected via trial-and-error, which is impractical for antenna optimizations that are computationally expensive. On the other hand, multiobjective PSO (MOPSO) provides an efficient way of exploiting the best tradeoff possibilities between multiple design goals. The output of MOPSO is the Pareto front that consists of a set of nondominated designs, and one has the freedom in selecting any of them according to the specific design requirement.

3. IMPLEMENTATION OF THE PSO ENGINE

With the unique advantages of applying PSO to antenna design problems discussed above, a PSO engine is implemented at UCLA with the full capability to address continuous, binary, single- and multiobjective optimizations [9]. Different EM analyzers are linked with the PSO kernel via a user-oriented interface, in order to simulate the performance of candidate antenna designs and to evaluate the fitness function. For each particular design, the user needs to select a proper optimization scenario and an EM analyzer according to the modelling of problem. A flowchart of the PSO engine is sketched in Figure 2. In this section, the implementation of different optimization scenarios is briefly described by following canonical PSO algorithms proposed in [10–12]. Optional parameters are also specified similar to the values suggested in these literature to maximize the optimizer's performance.

3.1. *Real-number PSO (RPSO)*

A RPSO algorithm proposed in [10] is implemented in the PSO kernel for continuous optimizations. The gbest swarm topology is used as well as in other optimization scenarios. Assume M agents are used in an N-dimensional problem,

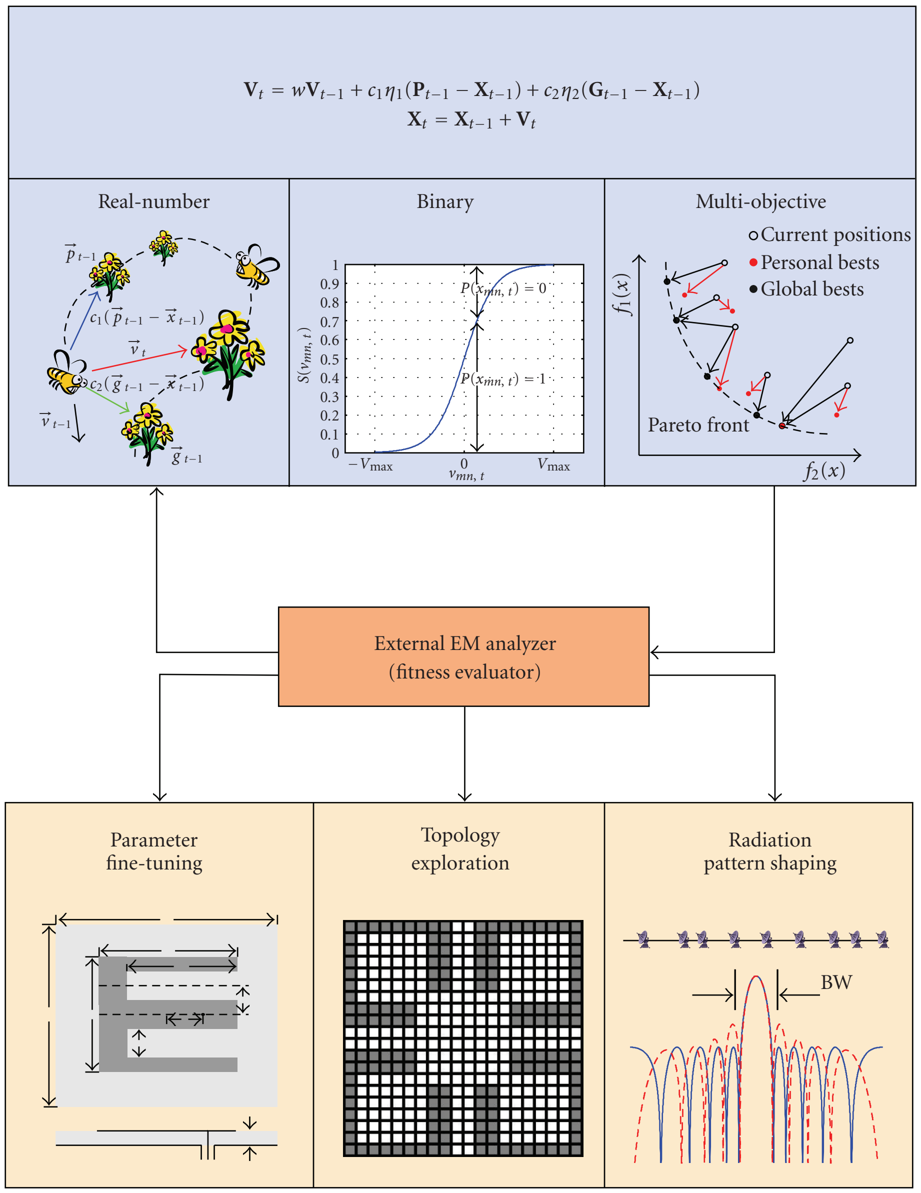

$\mathbf{V}_t = w\mathbf{V}_{t-1} + c_1\eta_1(\mathbf{P}_{t-1} - \mathbf{X}_{t-1}) + c_2\eta_2(\mathbf{G}_{t-1} - \mathbf{X}_{t-1})$
$\mathbf{X}_t = \mathbf{X}_{t-1} + \mathbf{V}_t$
Real-number
Binary
Multi-objective
Current positions
Personal bests
Global bests
Pareto front
External EM analyzer (fitness evaluator)
Parameter fine-tuning
Topology exploration
Radiation pattern shaping
BW

FIGURE 2: A flowchart of the PSO engine implemented at UCLA. The PSO kernel has the capability to handle continuous, binary, single- and multiobjective optimizations. Different EM analyzers are linked with the PSO kernel as external fitness evaluators.

at the tth iteration, the agents' velocities and positions are updated by

$$\mathbf{V}_t = w_t\mathbf{V}_{t-1} + c_1\eta_1 \otimes (\mathbf{P}_{t-1} - \mathbf{X}_{t-1}) + c_2\eta_2 \otimes (\mathbf{G}_{t-1} - \mathbf{X}_{t-1}), \tag{2}$$

$$\mathbf{X}_t = \mathbf{X}_{t-1} + \mathbf{V}_t. \tag{3}$$

Here, $\mathbf{V}$, $\mathbf{X}$, $\mathbf{P}$, and $\mathbf{G}$ are all $M \times N$ matrices that store the agents' velocities, positions, personal bests, and the global best, respectively. The notation $\otimes$ represents a component-wise multiplication. As suggested by [10], in the velocity updating (2), a time-varying inertia weight w_t is applied by changing its value from 0.9 at the beginning of optimization to 0.4 towards the end. Each component in matrices η_1 and η_2 has a uniform distribution in $(0, 1)$ to inject the randomness. Constants c_1 and c_2 both take a value of 2.0 to balance the cognitive and social influences.

The maximum velocity, $V_{\max}$, is imposed to prevent an agent from flying out of the physically meaningful solution space too often [13]. The value of each component of $V_{\max}$ is selected to be equal to the dynamic range of that dimension. However, this limitation does not always constrain the agent in the solution space. In [4, 14], three basic boundary handling techniques, the *absorbing* wall, the *reflecting* wall, and the *invisible* wall, are illustrated and compared. In RPSO, the invisible boundary condition is used due to its distinct advantage in reducing the computational cost.

3.2. *Binary PSO (BPSO)*

The BPSO subroutine in the PSO kernel follows an algorithm proposed by Kennedy and Eberhart [11]. In BPSO, the velocity update has a similar form to (2), while the agents' positions are updated via the sigmoid limiting transformation instead of directly applying (3). In particular, for the nth dimension of the mth agent, the sigmoid limiting transformation is defined as

$$S(v_{mn,t}) = \frac{1}{1+e^{-v_{mn,t}}}, \tag{4}$$

where $v_{mn,t}$ is the agent's velocity component in the nth dimension at the tth iteration. The associated position component, $x_{mn,t}$, is updated by

$$x_{mn,t} = \begin{cases} 1, & r_{mn,t} < S(v_{mn,t}), \\ 0, & r_{mn,t} \geq S(v_{mn,t}), \end{cases} \tag{5}$$

with $r_{mn,t}$ a random number uniformly distributed in $(0, 1)$. A maximum velocity of $V_{\max} = 6.0$ is used as suggested by [11]. The inertia weight is kept as 1.0 throughout the optimization since the time varying inertia weight is observed to be harmful for the swarm's convergence in a binary solution space [9]. Boundary conditions are not necessary to be applied due to the fact that an agent will be always located in the binary solution space during the optimization.

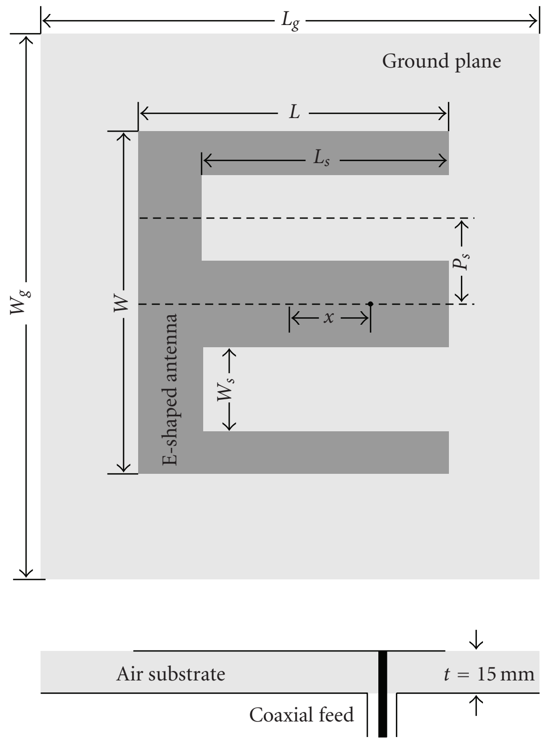


Figure 3: The topology of an E-shaped patch antenna. Each candidate design is represented by six geometrical parameters.

3.3. *Multi-objective PSO (MOPSO)*

Among several existing MOPSO algorithms [12, 15–17], the directed MOPSO (d-MOPSO) proposed by Fieldsend and Singh is selected due to its potential for handling more than two design objectives. In MOPSO, an archive that stores all nondominated solutions is updated at each iteration by following an algorithm proposed in [18]. The velocity and position updating equations have the same form as in single-objective PSO algorithms, while the global best of each agent is selected as the closest nondominated solution to that agent on the Pareto front in the objective space. A turbulence term is added into the velocity updating, and the magnitude of each turbulence component is one-tenth of the dynamic range of that dimension.

Before applying the PSO engine to practical antenna design problems, the algorithms described above are well examined using classic functional testbeds provided in [2] (for RPSO), [11] (for BPSO), and [15] (for MOPSO). The testing results and parameter tuning of these algorithms will not be presented here in order to keep the paper in a reasonable length, while interested readers are encouraged to refer to [14, 19] for relevant details that validate the PSO engine.

4. REAL-NUMBER PSO: A DUAL-BAND E-SHAPED PATCH ANTENNA

The first example of PSO-optimized antenna is presented in this section by applying RPSO in a parameter refinement problem [20]. In order to cover frequency bands for cellular communication systems such as the digital communication system (DCS, 1.71–1.88 GHz) and wireless local area networks (WLAN, 2.40 GH–2.48 GHz), a base station antenna is proposed in [21] for dual-band or wideband applications. The configuration of the antenna is plotted in Figure 3, which resembles a capital letter "E" and the design is therefore referred as an E-shaped antenna. Since the operating frequencies of this resonant-type antenna are closely related to current path lengths in the patch, the six geometrical parameters in Figure 3, including the patch length L, the patch width W, the slot length L_s, the slot width W_s, the slot position P_s, and the feed position x are optimized to design a dual-band antenna with resonant frequencies at 1.8 GHz and 2.4 GHz.

As discussed above, optimization of S_{11} can be formulated as a minimax problem. In other words, the relatively worse S_{11} at two desired frequencies is minimized. The fitness functions is therefore defined as

$$f = 50 + \max\{S_{11}(1.8\text{ GHz}), S_{11}(2.4\text{ GHz})\}. \tag{6}$$

In (6), $S_{11}(f)$ is evaluated by the finite difference time domain (FDTD) algorithm and each FDTD simulation takes about 7 minutes. The six geometrical parameters are optimized by being subjected to (unit: mm)

$$\begin{aligned} &L \in (30, 96), && W \in (30, 96), && L_s \in (0, 96), \\ &W_s \in (0, 96), && P_s \in (0, 96), && x \in (-48, 48). \end{aligned} \tag{7}$$

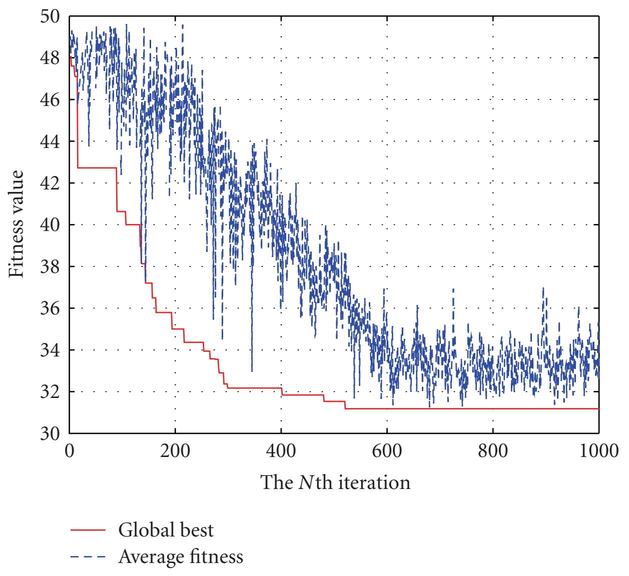


FIGURE 4: Convergence curves of the RPSO optimization by using a 10-agent swarm for 1000 iterations and applying the fitness function defined in (6).

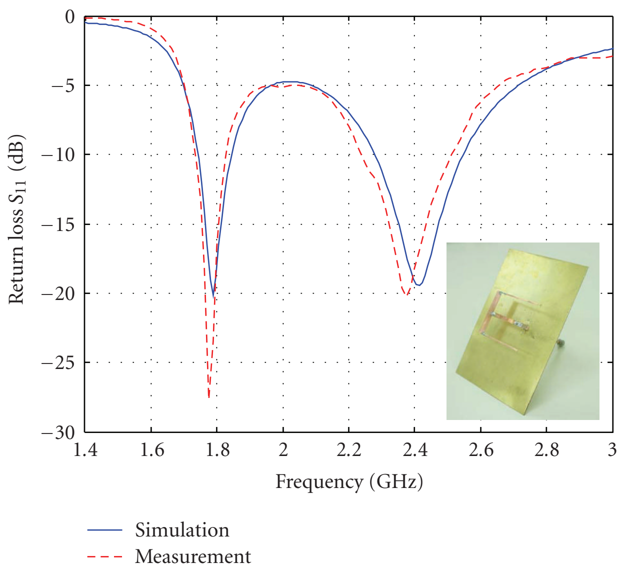


FIGURE 5: Simulated and measured S_{11} curves of the optimal dual-band antenna, which is prototyped and shown by the inset. Measurement results show a −15 dB return loss at both 1.8 GHz and 2.4 GHz.

For each candidate design, the following equations also need to be satisfied to maintain the E-shape of the patch and to retain the desired dual-band performance:

$$L_s < L, \qquad P_s > \frac{W_s}{2}, \qquad P_s + \frac{W_s}{2} < \frac{W}{2}, \qquad |x| < \frac{L}{2}. \tag{8}$$

Before evaluating the fitness function at each iteration, the six geometrical parameters are checked against (8), and the agent is treated as out-of-boundary if any equation is not satisfied.

In order to overcome the significant computational cost induced by FDTD simulations, the algorithm is implemented on 4 parallel Intel Xeon 3.0 GHz processors at UCLA Advanced Technology Service's Beowulf Linux cluster. The message passing interface (MPI) is used to communicate between the master and slave nodes. Figure 4 plots the convergence curves by using a 10-agent swarm in the optimization for 1000 iterations. The optimal design is observed after the 500th iteration with six geometrical parameters optimized as (unit: mm)

$$\begin{aligned} L &= 54.0, & W &= 46.0, & L_s &= 47.0, \\ W_s &= 20.0, & P_s &= 12.0, & x &= 14.0. \end{aligned} \tag{9}$$

The optimal dual-band antenna is prototyped and measured, with both simulated and measured S_{11} results plotted in Figure 5. A photograph of the antenna is shown by the inset. The optimal design has S_{11} values of −18.5 dB and −19.4 dB at 1.8 GHz and 2.4 GHz, respectively. The associated S_{11} < 10 dB bandwidths are 110 MHz at 1.8 GHz (6.1%) and 270 MHz at 2.4 GHz (11.2%). The measured results agree quite well with the simulation despite a slight frequency shift. A less than −15 dB return loss is observed at both operating frequencies.

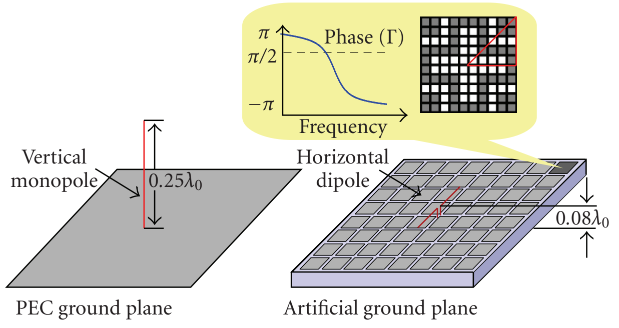


FIGURE 6: The profile of a conventional vertical dipole can be significantly reduced by an SWA. The unit cell of artificial ground plane in SWA is optimized by BPSO to match the horizontal dipole.

5. BINARY PSO: ARTIFICIAL GROUND PLANE FOR SURFACE WAVE ANTENNAS

As a comparison to the parameter refinement problem discussed above, the design of an artificial ground plane for surface wave antennas (SWAs) is presented in this section to illustrate the capability of BPSO in topology explorations. As shown in Figure 6, vertical monopole antennas with a perfect electric conductor (PEC) as the ground plane are often used in mobile communications (for instance, on the roof of a vehicle) to obtain the maximum radiation near the grazing angle. In order to reduce the profile of the antenna, which is typically a quarter free space wavelength (λ_0), a SWA is proposed in [22] by horizontally mounting a 0.28 λ_0 long dipole over an artificial ground plane.

According to the image theory in electromagnetics, it is difficult to match a horizontal dipole near the ground plane due to the radiation cancellation by the image of dipole.

However, investigations [23] have shown that by periodically loading a dielectric slab, a horizontal dipole can be well matched in a frequency band where the phase of normal reflection coefficient (Γ) of the periodic structure varies between 45° and 135°. Generally this unique reflection characteristic of periodic structure is closely related to the topology of unit cell. In this example, the unit cell with a dimension of 7.5 mm $\times$ 7.5 mm is discretized into 10×10 pixels. The BPSO algorithm is applied to determine the PEC/dielectric state of each pixel and explore the unit cell topology, in order to achieve the lowest center frequency of the matching band where phase (Γ) = 90°. For a fixed unit cell dimension, the optimal solution is inherently the miniaturized design that improves the homogeneity of the artificial ground plane. Compared to the optimization goal in the design of E-shaped antenna, it should be clarified that we are still trying to minimize the return loss of the SWA, while it is accomplished here in an indirect manner by characterizing the reflection coefficient of the artificial ground plane.

As shown in Figure 6, in order to achieve a polarization-independent design, the candidate unit cell has a fourfold symmetry, and only 15 pixels are optimized. A 15-bit binary string x_i ($i = 1, 2, \ldots, 15$) is used by each agent to map the candidate design into a discrete solution space. In particular, the ith pixel is filled by PEC, if $x_i = 1$, and by dielectric when $x_i = 0$. The fitness function is defined as

$$f = \text{freq}_{\text{phase}\,(\Gamma)=90^\circ} \tag{10}$$

to minimize the phase (Γ) = 90° frequency. For fabrication purpose, unit cell topologies with any two PEC pixels diagonally connected are prohibited and are assigned a very bad fitness value. The reflection phase of each candidate design is calculated by analyzing the unit cell using FDTD algorithm with periodic boundary condition, which takes about 3 minutes. In order to reduce the computational cost, an agent is checked at each iteration against the previous records before its fitness evaluation. If the same topology has been visited by the optimizer, the associated fitness value will be directly assigned to the agent with the repeated solution. The full-wave analysis is only executed for those agents located at positions that have never been explored. This repeated solution checking scheme is suitable for accelerating the BPSO process since the solution space is a finite set, particularly during the late stage of optimization when most agents have converged to the global optima.

Figure 7 plots the convergence curves by using a 10-agent swarm for 100 iterations. The global optimum is observed at the 27th iteration, and the average fitness indicates a quite good convergence at the 100th iteration. The optimal design is represented by a binary string of

$$\{x_i\} = \{0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,1\,1\,1\}, \tag{11}$$

and the unit cell topology is plotted in Figure 8. The simulated reflection phase of the optimal design is also plotted. The phase (Γ) = 90° frequency is observed at 5.18 GHz.

A SWA based on the optimized artificial ground plane is fabricated and measured, as shown in Figure 9. The ground plane is etched on a 3-mm-thick substrate with a relative permittivity of 2.94 ($\epsilon_r = 2.94$) and consists of 18×18 optimized unit cells. Figure 10 plots measured return loss of the SWA. The antenna has a minimum return loss of 5.26 GHz, which is only 1.5% off the optimized phase (Γ) = 90° frequency at 5.18 GHz. Also plotted in Figure 10 is the S_{11} curve of a horizontal dipole with the same size on a PEC ground plane. It is observed that the matching of antenna is significantly deteriorated due to the existence of image dipole. The measured radiation pattern of SWA is shown in Figure 11, where maximum radiations are observed at 70° and 290°. Compared to the pattern of a vertical monopole on a PEC ground plane with the same size of $2.4\lambda_0 \times 2.4\lambda_0$, the pattern

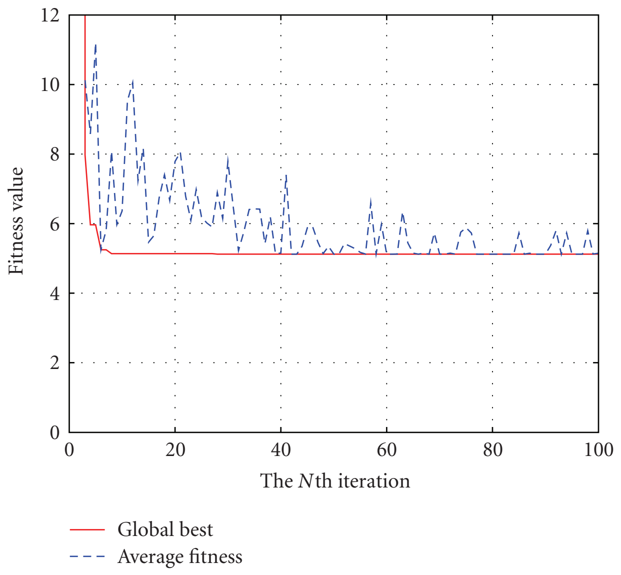


Figure 7: Convergence curves of the BPSO optimization by using 10 agents for 100 iterations. The fitness function is defined in (10).

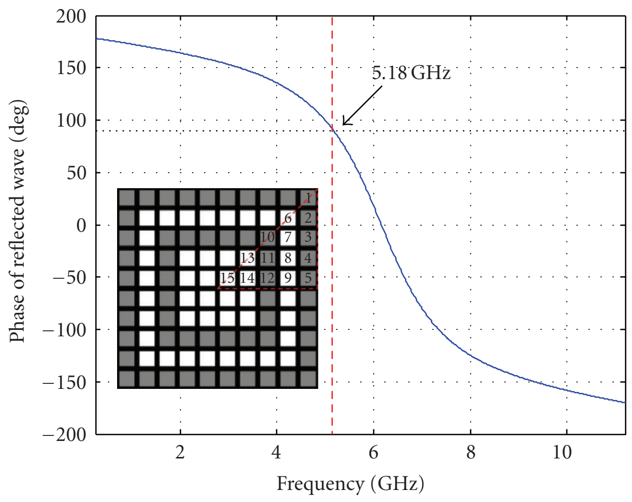


Figure 8: Simulated reflection phase of the optimal design with a phase (Γ) = 90° frequency of 5.18 GHz observed. The optimal unit cell topology is plotted by the inset.

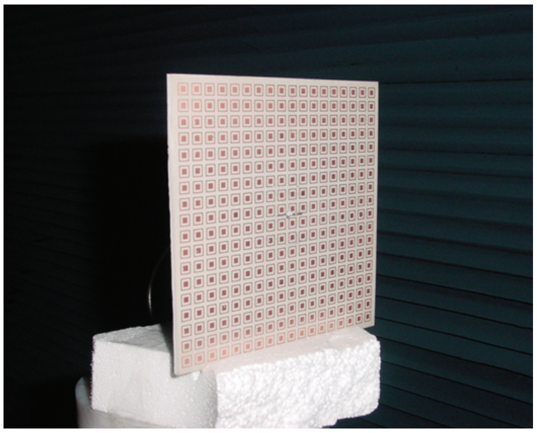


FIGURE 9: An SWA is fabricated based on the artificial ground plane which consists of 18 × 18 optimized unit cells.

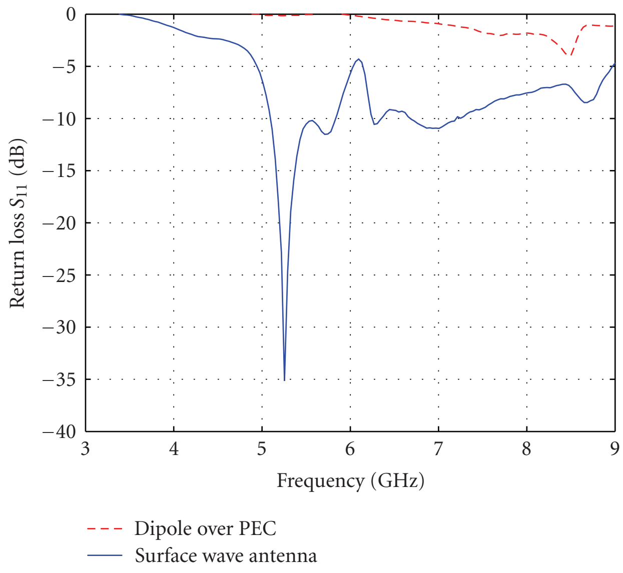


FIGURE 10: Measured S_{11} of the SWA. Compared to a horizontal dipole over a conventional PEC ground, the SWA is well matched at 5.26 GHz with a return loss of −35 dB.

of SWA shows a great similarity, while the 0.25 λ_0-profile of the vertical monopole is reduced to 0.08 λ_0 in this example.

6. MULTIOBJECTIVE PSO: LOW-SIDELOBE APERIODIC ANTENNA ARRAYS

Antenna arrays have been broadly used in communication and radar systems where a highly directive radiation pattern is desired. For conventional periodic arrays, complete design and synthesis theories have been established for several decades. However, grating lobes (sidelobes with the same level as the main beam) in the radiation patterns of periodic arrays are inevitable, when the uniform element spacing is greater than λ_0. In recent years, evolutionary algorithms have been extensively used in the design of aperiodic antenna arrays to obtain the lowest peak SLL [24–27]. In this example, MOPSO is applied to design an 8-element nonuniform

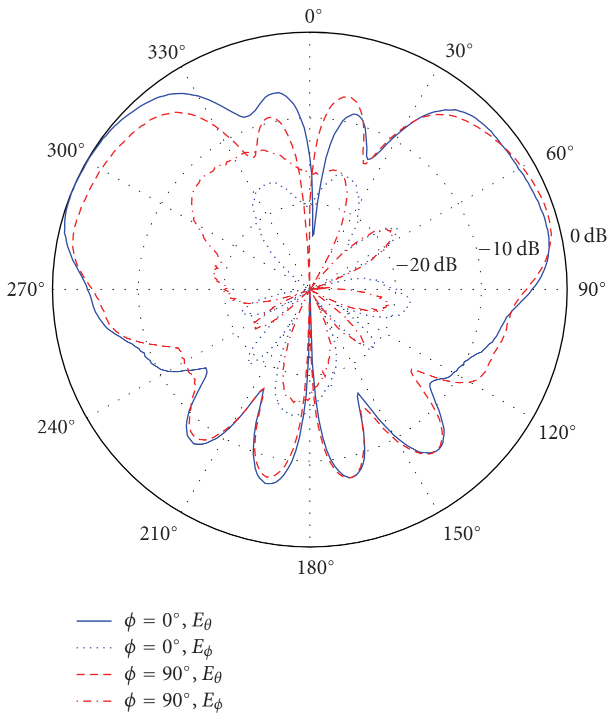


FIGURE 11: Measured radiation pattern of the SWA which resembles the pattern of a vertical monopole.

antenna array by optimizing the element positions, in order to investigate the best tradeoff between its peak SLL and beamwidth.

An 8-element linear nonuniform array is shown in Figure 12. The antenna elements are identical patch antennas etched on an 0.787 mm thick substrate with $\epsilon_r = 2.2$. The dimension of each patch antenna is $0.49\lambda_g \times 0.70\lambda_g$ ($0.33\lambda_0 \times 0.48\lambda_0$). The array is assumed to be symmetric with respect to the z-axis, and the array configuration is mapped into a 4-dimensional real-valued solution space $\{d_i\}$ ($i = 1, 2, 3, 4$) which represents the element spacings. All elements are allowed to vary within $\pm 5.0\lambda_0$, which gives a maximum element spacing of $d_{\text{avg}} = 1.43\lambda_0$. To prevent adjacent elements from getting overlapped, the element spacings are subjected to

$$d_1 \in (0.24\lambda_0, 3.56\lambda_0), \qquad d_2, d_3, d_4 \in (0.48\lambda_0, 3.8\lambda_0),$$
$$\sum_{i=1}^{4} d_i \leq 5.0\lambda_0. \tag{12}$$

The lower bound of each variable is defined according to the width of patch, and the upper bound is calculated by assuming that other three patches are connected to each other.

The peak SLL and the null-to-null beamwidth are represented by the following two fitness functions:

$$f_1 = \max\{20\log|\text{AF}(\theta) \times \text{EF}(\theta)|\}, \quad 0° < \theta < \theta_n,$$
$$f_2 = 2(90^\circ - \theta_n), \tag{13}$$

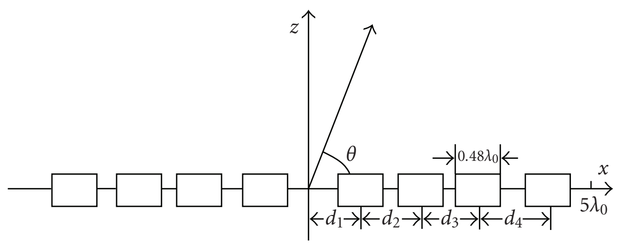


Figure 12: An 8-element, symmetric, nonuniform antenna array with a maximum aperture size of 10 λ_0. The element positions are optimized by MOPSO to investigate the best tradeoff relationship between its peak SLL and beamwidth.

where AF(θ) and EF(θ) are the array factor, and the element pattern analytically calculated by

$$\mathrm{AF}(\theta) = \sum_{i=1}^{4} \cos\left[2\pi \sum_{k=1}^{i} d_k(\cos\theta)\right], \quad (14)$$

$$\mathrm{EF}(\theta) = \tan\theta \sin(0.49\cos\theta), \quad (15)$$

respectively. θ_n denotes the position of the first null away from the broadside ($\theta = 90^\circ$). It should be noted that the radiation pattern only depends on the elevation angle θ in this linear array case. Compare to the general pattern optimization problem discussed in Section 2, the fitness function has a simpler form since the design goal is to achieve the minimum peak SLL instead of matching a desired pattern in the entire angular domain.

The optimization is executed for 4000 iterations using a 20-agent swarm. Figure 13 plots the Pareto front. Each non-dominated design located at (ξ, η) can be interpreted as, for any design that has a null-to-null beamwidth of η-degree, its peak SLL can not be lower than ξ-dB. As a comparison, the beamwidth-SLL relationship of an 8-element periodic array is plotted in Figure 13. This curve is completely dominated by the Pareto front, which indicates that an aperiodic array on the Pareto front always has a lower SLL than a periodic array with the same beamwidth. For instance, design A is arbitrarily selected from the Pareto front with optimized element spacings of

$$\{d_i\} = \{0.49\lambda_0, 2.17\lambda_0, 1.13\lambda_0, 1.21\lambda_0\}, \quad (16)$$

and an aperture size of 10 λ_0. Compared to a periodic array with the same aperture size, A', the aperiodic array has a similar beamwidth but a significantly reduced peak SLL. Their radiation patterns calculated by (14) and (15) are plotted in Figure 14. The periodic array A' has a high SLL of −4.3 dB due to the strong grating lobes in the array factor, while design A has a much lower SLL of −10.1 dB by aperiodically arranging the elements and eliminating the grating lobes. The directivities of arrays A' and A are 17.2 dB and 17.6 dB, respectively.

Figure 15 shows fabricated prototypes of both arrays, and their measured radiation patterns at the operating frequency of 15 GHz are shown in Figure 16. It is observed that the −3.9 dB grating lobes of the periodic array are reduced to −9.6 dB, and the nonuniform array exhibits an approximate equal-sidelobe feature. Furthermore, the beamwidth, the number, and the location of sidelobes of both arrays agree fairly well with the analytical simulation results shown in Figure 14. It is believed that in the fabricated antenna arrays, the mutual coupling is considerably small due to a large $d_{\mathrm{avg}} = 1.43\lambda_0$, which makes the optimization result a good prediction for designing a low-SLL array with a sparse configuration.

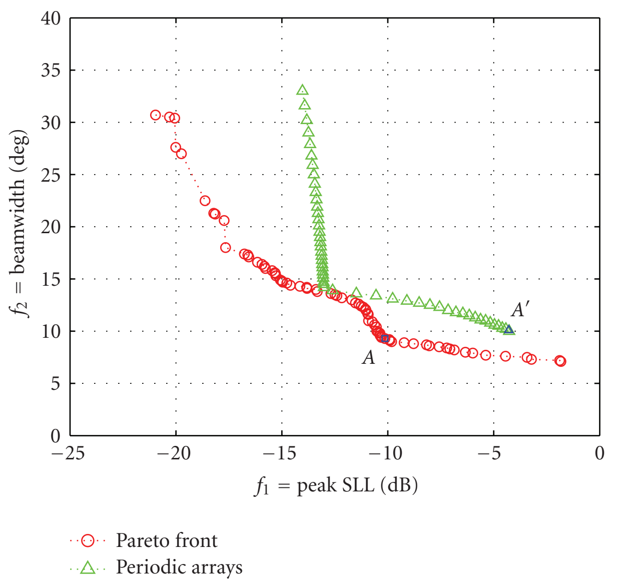


Figure 13: The Pareto front of the MOPSO optimization for trading off the peak SLL and the beamwidth of an 8-element nonuniform patch antenna array. The SLL-beamwidth relationship of periodic arrays is also plotted.

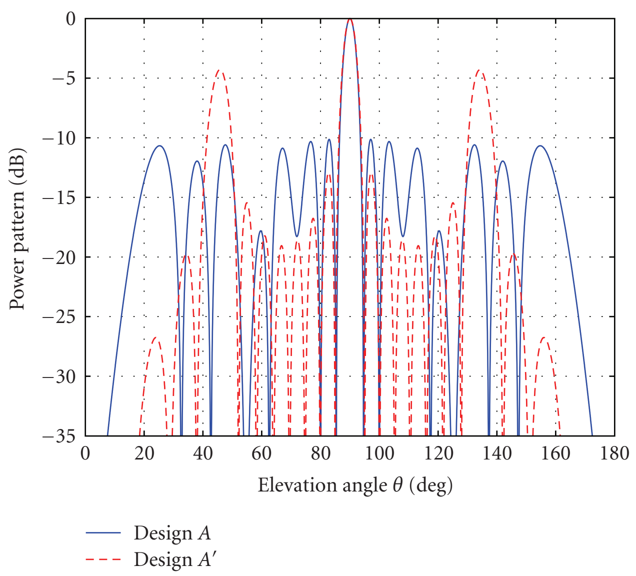


Figure 14: Comparison between the radiation patterns of a selected nonuniform array A and a periodic array A' in Figure 13.

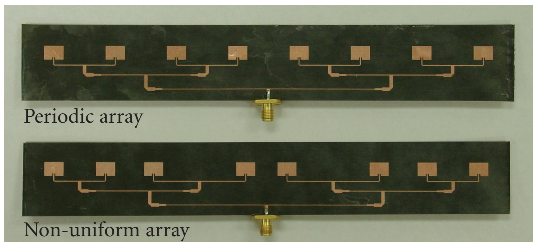


Figure 15: Fabricated array prototypes of designs A and A' in Figure 13.

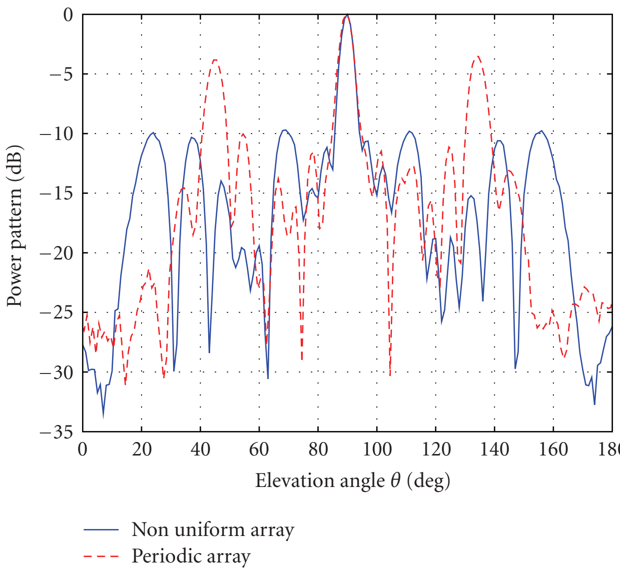


Figure 16: Measured radiation patterns of designs A and A'. Grating lobes of the periodic array are eliminated by the nonuniform array configuration.

7. CONCLUSION

The enormous interest in applying PSO technique to antenna designs is evident due to the wide range of practical problems that can be solved by this novel, nature-inspired, evolutionary algorithm. Modelled by fundamental Newtonian mechanics, the swarm intelligence is imbedded in the design process to accommodate different types of optimizations. With this powerful PSO engine available, it is relatively easy to apply it to different problems without making significant changes to the kernel of the optimizer.

In this paper, we only present three examples to illustrate the functionality of PSO in a large variety of real-world problems. These examples are categorized into the optimizations of return loss and radiation pattern, while the flexibility in defining the fitness function allows PSO to address other design requirements. It is observed that as a stochastic global optimizer, PSO is particularly suitable for antenna optimizations which are in general extremely nonlinear and multimodal. Both simulation and measurement results of PSO-optimized antennas are presented in each example, in order to validate the capability of PSO in producing a useful and practical design.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Wash, Australia, 1995 November.

[2] J. Kennedy and R. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[3] R. Poli, J. Kennedy, and T. Blackwell, "Paticle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[4] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397–407, 2004.

[5] Y. Rahmat-Samii, D. Gies, and J. Robinson, "Particle swarm optimization (PSO): a novel paradigm for antenna designs," *The Radio Science Bulletin*, vol. 305, pp. 14–22, 2003.

[6] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[7] Y. Rahmat-Samii and E. Michielssen, Eds., *Electromagnetic Optimization by Genetic Algorithms*, John Wiley & Sons, New York, NY, USA, 1999.

[8] N. Jin and Y. Rahmat-Samii, "A hybrid real-binary particle swarm optimization (HPSO) algorithm: concepts and implementations," in *Proceedings of IEEE Antennas Propagation International Symposium*, pp. 1585–1588, Honolulu, Hawaii, USA, 2007 June.

[9] N. Jin and Y. Rahmat-Samii, "Advances in particle swarm optimization for antenna designs: real-number, binary, single-objective and multiobjective implementations," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3 I, pp. 556–567, 2007.

[10] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1945–1950, Washington, DC, USA, 1999 July.

[11] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC '97)*, vol. 5, pp. 4104–4108, Orlando, FLa, USA, 1997 October.

[12] J. Fieldsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence," in *Proceedings of the U.K. Workshop on Computational Intelligence (UKCI '02)*, pp. 37–44, Birmingham, UK, 2002 September.

[13] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '00)*, vol. 1, pp. 84–88, La Jolla, Calif, USA, 2000 July.

[14] S. Xu and Y. Rahmat-Samii, "Boundary conditions in particle swarm optimization revisited," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3 I, pp. 760–765, 2007.

[15] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, no. 2-3, pp. 235–306, 2002.

[16] X. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC*

'02), vol. 2, pp. 1677–1681, Honolulu, Hawaii, USA, 2002 May.

[17] C. Coello Coello and M. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1051–1056, Honolulu, Hawaii, USA, 2002 May.

[18] Y. Jin, M. Olhofer, and B. Sendhoff, "Dynamic weighted aggregation for evolutionary multiobjective optimization: why does it work and how?" in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01)*, pp. 1042–1049, San Francisco, Calif, USA, 2001 July.

[19] D. Gies, "Particle swarm optimization: applications in electromagnetic design," M. Eng. thesis, Los Angeles, Calif, USA, 2004.

[20] N. Jin and Y. Rahmat-Samii, "Parallel particle swarm optimization and finite difference time-domain (PSO/FDTD) algorithm for multiband and wide-band patch antenna designs," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3459–3468, 2005.

[21] F. Yang, X.-X. Zhang, X. Ye, and Y. Rahmat-Samii, "Wide-band E-shaped patch antennas for wireless communications," *IEEE Transactions on Antennas and Propagation*, vol. 49, no. 7, pp. 1094–1100, 2001.

[22] F. Yang and Y. Rahmat-Samii, "Wire antennas on artificial complex ground planes: a new generation of low gain antennas," in *Proceedings of IEEE Antennas Propagation International Symposium*, vol. 1, pp. 309–312, Monterey, Calif, USA, 2004 June.

[23] F. Yang and Y. Rahmat-Samii, "Reflection phase characterizations of the EBG ground plane for low profile wire antenna applications," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 10, pp. 2691–2703, 2003.

[24] R. Haupt, "Thinned arrays using genetic algorithms," in *Electromagnetic Optimization by Genetic Algorithms*, Y. Rahmat-Samii and E. Michielssen, Eds., John Wiley & Sons, New York, NY, USA, 1999.

[25] K. Chellapilla, S. S. Rao, and A. Hoorfar, "Optimization of thinned phased arrays using evolutionary programming," in *Proceedings of the 7th International Conference on Evolutionary Programming*, vol. 1447, pp. 157–166, San Diego, Calif, USA, 1998 March.

[26] D. Gies and Y. Rahmat-Samii, "Particle swarm optimization for reconfigurable phase-differentiated array design," *Microwave and Optical Technology Letters*, vol. 38, no. 3, pp. 172–175, 2003.

[27] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771–779, 2004.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 745694, 8 pages
doi:10.1155/2008/745694

Research Article

Generating Complete Bifurcation Diagrams Using a Dynamic Environment Particle Swarm Optimization Algorithm

Julio Barrera, Juan J. Flores, and Claudio Fuerte-Esquivel

Programa de Graduados e Investigación en Ingeniería Eléctrica, Universidad Michoacana de San Nicolas de Hidalgo, Ciudad Universitaria, 58030 Morelia Michoacan, Mexico

Correspondence should be addressed to Julio Barrera, julio.barrera@gmail.com

Received 20 July 2007; Accepted 28 November 2007

Recommended by Jim Kennedy

A dynamic system is represented as a set of equations that specify how variables change over time. The equations in the system specify how to compute the new values of the state variables as a function of their current values and the values of the control parameters. If those parameters change beyond certain values, the system exhibits qualitative changes in its behavior. Those qualitative changes are called bifurcations, and the values for the parameters where those changes occur are called bifurcation points. In this contribution, we present an application of particle swarm optimization methods for dynamic environments for plotting bifurcation diagrams used in the analysis of dynamical systems. The use of particle swarm optimization methods presents various advantages over traditional methods.

1. INTRODUCTION

A dynamic system is represented as a set of equations that specify how variables change over time. The minimum set of variables that uniquely determines the state of a system is called state variables. The equations in the system specify how to compute the new values of the state variables as a function of their current values and the values of the control parameters. If we allow the control parameters to change, the system changes with them. If those parameters change beyond certain values, the system exhibits qualitative changes in its behavior. Those qualitative changes are called bifurcations, and the values for the parameters where those changes occur are called bifurcation points.

If we allow one parameter to vary and plot the norm of the vector of state variables for which we find fixed points of the system, against the changing parameter, we obtain what we call a bifurcation diagram. In a bifurcation diagram, we see that fixed points may disappear, appear, or even change their stability nature as the changing parameter varies. Those changes may occur even for infinitesimal changes in the control parameter. Bifurcation diagrams are a tool used in stability analysis of dynamic systems.

Traditionally, bifurcation diagrams are plotted starting from a fixed point and using the so-called continuation methods to determine how the fixed point moves with changes in the control parameter.

In this paper, we propose an alternative by using an intelligent optimization method, namely, particle swarm optimization (PSO); we can determine all fixed points of the system for a given value of the control parameter. Iterating through the allowed range of the control parameter, we can build the whole bifurcation diagram in one pass, without the need for continuation methods.

PSO is one of the several bioinspired techniques found in artificial intelligence. PSO algorithms are inspired in the behavior of bird flocking and fish schooling [1]. When a bird finds a region with food, the others will follow it in its direction.

To illustrate our ideas and implementation work, we use a set of examples in this paper. The first examples show how the system is able to produce bifurcation diagrams for problems belonging to each one of the classes known as normal forms. Normal forms typify dynamic systems by the kind of bifurcations they may exhibit. The results match very accurately those shown in text books and the results produced with XPPAUT [2]. The last example shows a power system and its governing differential equations. Even though the system may look like a toy problem, it is in fact a benchmark in electrical engineering, for the richness of the set of qualitative features that its behavior exhibits.

The rest of the article is organized as follows. Section 2 provides the necessary background on dynamic systems and bifurcation diagrams. Section 3 mentions the particular ideas in the implementation of the PSO optimizer used in this work. (We assume the reader is already familiar with PSO, and just mention the fine details. The novice reader may consult [3].) Section 4 proposes a novel approach to the construction of bifurcation diagrams, using a PSO-based optimizer to determine all fixed points of a dynamic system. Section 5 illustrates the validity and applicability of our proposal through several examples. Finally, Section 6 concludes the work with a comparison between our proposal and traditional methods, and states several directions of research found in our agenda.

2. BACKGROUND

Electric power systems are the large physical systems interconnecting various devices to perform generation, delivery, and consumption of electricity. From an engineering point of view, the task in the power system operation is to maintain proper frequency and voltage magnitude within appropriate tolerance so that the system may operate at a stable equilibrium point in the steady state. Power system equilibrium equations which determine the system's operation state typically depend on a very large number of parameters. Hence, numerous parameters such as generation, load, network conditions, and so forth, that can change with time and circumstances, affect the system behavior. Under normal variations of those parameters, the operating point varies smoothly so the variation can be tracked by local linear analysis. However, this behavior changes qualitatively at certain critical parameter values such that the equilibrium point becomes unstable causing operational problems. One such operation problem occurs in the area of system magnitude voltages, which gradually decrease as the system load increases. This voltage decrement could be spread uncontrollably throughout the power system causing a total (blackout) or partial disruption on the power system operation. The phenomenon of this catastrophic event is referred to as a voltage collapse. Nowadays, voltage collapse problems are considered to be the principal threat to power system stability, security, and reliability in many utilities around the world [4, 5].

Several methods have been proposed to find the critical parameters that make an equilibrium point become unstable or disappear, producing a voltage collapse [5]. However, in the last decade, bifurcation theory received considerable attention from researches to increase the understanding of the complex behavior associated to the voltage collapse phenomenon [6–13]. Bifurcation theory refers to characterizing sudden changes in the qualitative response of the system as its parameters are varied smoothly and continuously over a specified range [13, 14]. When these changes relate to qualitative changes occurring in the neighborhood of an equilibrium point or limit cycle such that an equilibrium point or limit cycle appears, disappears, or losses stability, they are referred to as local bifurcations. In bifurcation problems, it is very useful to consider a space formed by system state variables and parameters, called state-control space. In this space, locations at which bifurcations occur are called bifurcation points.

The numerical analysis to locate these points is based on the principle of continuation [14, 15]. A continuation method generates a chain of solutions from an established solution of the equations representing the system under analysis. The solution branch thus established can then be examined for bifurcation points, at which a qualitative change of the preceding solution type can be observed. The representation of this branch of solutions in the state-control space is referred to as bifurcation diagram. The continuation methods applied to obtain bifurcation diagrams are based on prediction-correction schemes [15]. Generally, most prediction-correction schemes have similar prediction steps, prediction along the tangent direction, but different correction steps [15].

Opposite to the continuation methods commonly used to compute bifurcation diagrams, this paper proposes a heuristic-based method to assess these diagrams. The proposed approach solves the set of nonlinear equations representing the system under analysis by applying particle swarm optimization.

This new method is first tested to obtain bifurcation diagrams of normal forms related to generic bifurcations. In order to validate the new proposal, it is applied to a benchmark power system proposed in [8] to analyze the bifurcation diagram involved in a voltage collapse process. The model developed in [8] has been thoroughly tested by several authors [9–11, 13] in order to show that several kinds of bifurcations coexist in the voltage collapse phenomenon at different levels of system loadings. The results obtained compared well with those obtained with the popular continuation package.

3. PARTICLE SWARM OPTIMIZATION METHODS FOR DYNAMIC ENVIRONMENTS

The problem of plotting a bifurcation diagram can be stated as the problem to find all solutions of a nonlinear equation system in a given region and trace the solutions when a change in one or more parameters is introduced. This is the case of environments which present changes in the position or number of the optima of the given fitness function in the presence of one or more parameters that change. The particle swarm optimization method, developed by Li et al. [16], presents good results in this type of environments.

3.1. *Particle swarm optimization*

Particle swarm optimization is a nature inspired algorithm that has been developed for function optimization; particle swarm algorithms are inspired in the behavior of bird flocking and fish schooling [1]. When a bird finds a region with food, other birds in the flock will follow it in its direction. In the particle swarm algorithm, a point in the search space is viewed as a particle; the fitness value of a particle is equivalent to the fitness value in genetic algorithms.

In a first state, all particles have no velocity, and the particle with optimal value of fitness is selected and all the other particles are directed towards it. A velocity is computed for each of the other particles and the position of each particle is updated. This procedure is repeated for a given number of times until an optimum or a limit is reached. Variations of those algorithms have been developed for multimodal problems [3, 16].

3.2. *Genetic algorithms and species*

The particle swarm optimization method, developed by Xiaodong Li, takes two concepts from genetic algorithms for multimodal optimization: species and clearing. Dividing a population in species is a method used in genetic algorithms to preserve diversity; it consists of selecting the best available individual; all individuals with a distance less than a parameter value, called radius, with respect to the selected individual are considered to be in the same species. The selected individual and the individuals in the species are marked as used and the procedure is repeated until all individuals belong to a species. A good example is the species conservation algorithm of J.-P. Li [17]. In this method, the best individual in a species is preserved to be reinserted in the next generation of the population.

The fitness value of the individuals that are in the same species can be modified by a function to improve the probability of the best individual in the species being selected in the matting process. In some cases, only a given number of individuals are maintained with their fitness value without change, and the value of fitness of all other individuals is set to zero or they are placed in a random position within the search space; this procedure is called clearing. An example of this method is examined in the work of Pétrowski [18] where the fitness value of all individuals in a species, except for the best individual, is set to zero. A complete review of some of the methods of genetic algorithms can be found in [19].

3.3. *Hybrid particle swarm optimization method*

In addition to the basic particle swarm optimization method, two steps are added; after the initial swarm is created, the particle with the best fitness value is selected and all particles with distance less than the value of a parameter r are selected as individuals of a species; the global optimum of the particles in the species is set to the particle initially selected. The particles in the species are marked as used, and the next particle with the best fitness value available is selected to form a new species. The procedure is repeated until all particles belong to a species (the fitness value of all particles is preserved). After all species are formed, only a given number of particles are permitted in the species. Only the n particles closest to the particle with the best fitness value are preserved; the remaining particles are reinitialized at random positions.

Once the previous steps have been made, the velocity and position of all the particles are updated according to the rules given as follows:

$$\begin{aligned} v_{\text{new}} &= \chi\left[v + C_1R_1\left(P_{\text{best}} - P\right) + C_2R_2\left(P_{\text{global}} - P\right)\right], \\ P_{\text{new}} &= P + V_{\text{new}}, \end{aligned} \tag{1}$$

where v, P are the current velocity and position, respectively, R_1 and R_2 are random numbers generated in the range of $[0, 1]$, C_1 and C_2 are the learning factors, P_{best} is the position of the best fitness of the particle at the current iteration, P_{global} is the position of the particle with the best fitness in the swarm (in this case, in the species), and v_{new}, P_{new} represent the new position and velocity of the particle. The constant χ is calculated according to

$$\begin{aligned} \phi &= C_1 + C_2, \\ \chi &= \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}. \end{aligned} \tag{2}$$

The constant χ is called constriction coefficient [3, 16, 20], and it prevents that the particles explore too far away from the search space. Thus, we do not need a $v_{\max}$ limit for the velocity of the particles.

4. GENERATING BIFURCATION DIAGRAMS

Given the set of differential equations (see (3)) that models a dynamic system with one parameter:

$$\dot{x} = f(x, \alpha), \quad x \in R^n,\ \alpha \in R^1, \tag{3}$$

the generation of the bifurcation diagram consists of finding one stable fixed point and then using a continuation method to find the next fixed point based on the condition

$$f(x, \alpha) = 0 \tag{4}$$

which defines a smooth one-dimensional curve in R^{n+1}. Our approach to this problem is to determine all points $x \in X$, for a given region $X \subset R^n$, that satisfy (4) for a given value of the parameter α.

To find all points x using particle swarm optimization, we first transform the problem of finding all solutions for (4) into a problem of finding maxima using the transformation

$$g(x) = \frac{1}{1 + |f(x)|}. \tag{5}$$

This nontrivial transformation sends all points x such that $f(x) = 0$ to 1, and maps the domain of the function $f(x)$ to $(0, 1]$, thus zeroes of f map to maxima of function g.

Starting with a value $\alpha = \alpha_0$, the hybrid particle swarm optimization method is applied to a given number of iterations to find all maxima of function g with the fixed value α_0. Once all the points x for a given region and fixed parameter value α_0 are found, they are recorded; then the parameter α is changed to $\alpha_1 = \alpha_0 + \Delta\alpha$ and a new set of fixed points x is determined. The search is based on the information accumulated in the previous swarm. A bifurcation diagram is generated by changing the parameter α and recording all solutions found for each value α_k of the parameter.

5. RESULTS

We present two examples. The normal forms for systems with one parameter that exhibit bifurcations are examined; the

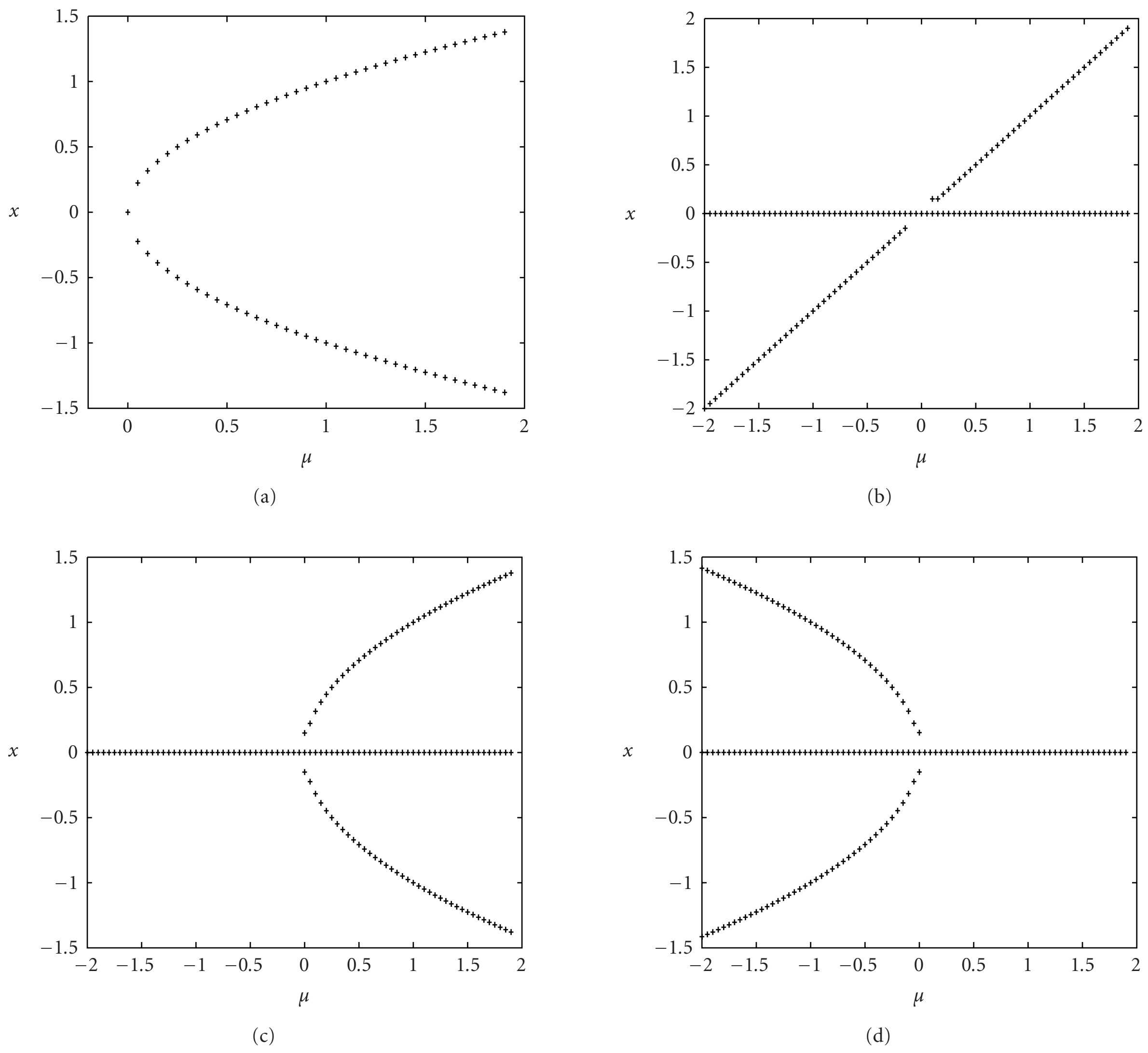


Figure 1: Bifurcation diagrams generated with the PSO for the normal forms: (a) saddle-node bifurcation (see (6)), (b) transcritical bifurcation (see (7)), (c) pitchfork supercritical bifurcation (see (8)), and (d) pitchfork subcritical bifurcation (see (9)).

bifurcation types that the systems exhibit are saddle-node, transcritical, pitchfork supercritical, and pitchfork subcritical [21, 22] (see Figure 1), and a comparative of bifurcation diagrams generated with the XPPAUT software and the particle swarm optimization method for an electrical network that illustrates the voltage collapse phenomenon arising from the load variation is presented.

5.1. Normal forms for systems with one parameter

A fixed point associated to a dynamic system where the Jacobian matrix presents all eigenvalues with nonzero real parts is called hyperbolic fixed point. If we examine the behavior of this fixed point under the variation of the parameter, the eigenvalues of the Jacobian matrix condition can only be violated in two ways: the real part of an eigenvalue approaches zero, or two eigenvalues reach the imaginary axis. When the condition is violated, the bifurcations that the dynamic system presents can be one of the bifurcations arising in the systems represented by

$$\dot{x} = \mu - x^2, \tag{6}$$

$$\dot{x} = (\mu - x)x, \tag{7}$$

$$\dot{x} = (\mu - x^2)x, \tag{8}$$

$$\dot{x} = (\mu + x^2)x. \tag{9}$$

These equations are called normal forms or topological normal forms, and they are employed to reduce a given nonlinear system to the simplest possible form preserving the dynamics in a neighborhood of the fixed point where the condition is violated. In this context, by means of a variable change, a dynamic system with one parameter, close to the hyperbolic fixed point, can be rewritten as one of the normal forms, having the same bifurcation diagram. The variable change is often nontrivial.

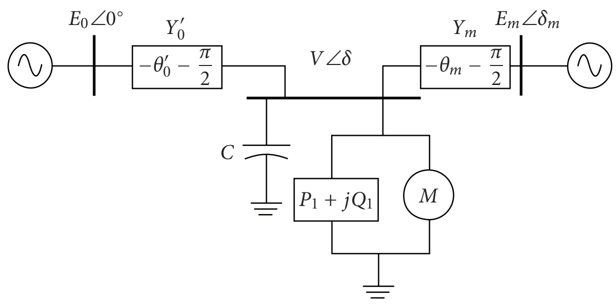


FIGURE 2: Diagram for the power system analyzed.

For the normal forms, the particle swarm optimization algorithm was set up with a swarm of 500 particles, a maximum of 10 particles per species, and $r = 0.15$. The search space for all normal forms was the interval $[-2, 2]$. The initial value $\mu = -2$ was incremented in $\Delta\mu = 0.05$ until $\mu = 2$. For each μ_k, the PSO was run for 100 cycles. The bifurcation diagrams that were found for each one of the normal forms are shown in Figure 1; these results match those in the bibliography (see [14, 21, 22]).

5.2. Bifurcation diagram for an electrical power system

The electrical network shown in Figure 2 has been used in [8, 23] and other papers to illustrate the voltage collapse phenomenon arising from the load variation. The three-node equivalent circuit is to be viewed as an equivalent circuit of a local area of interest connected to a large network. The network is modeled as an infinite bus represented by a voltage source providing constant voltage magnitude and phase, $E_0\theta_0$, regardless of power flow. The generator terminal bus is $E_m\delta_m$. The complex admittance of the transmission lines connected to the generator and infinite bus, a load, and a capacitor are also shown in Figure 2. The measured voltage at the load terminal is $V\delta$.

The system's dynamics are governed by the following four ordinary differential equations:

$$\begin{aligned}
\dot{\delta}_m &= \omega,\\
\dot{\omega} &= \frac{1}{M}\left(P_m - D_m\omega + VE_mY_m\sin(\delta-\delta_m-\theta_m) + E_m^2Y_m\sin\theta_m\right),\\
\dot{\delta} &= \frac{1}{K_{qw}}\left(-K_{qv2}V^2 - K_{qv}V + Q - Q_0 - Q_1\right),\\
\dot{V} &= \frac{1}{TK_{qw}K_{pv}}\left[-K_{qw}(P_0+P_1-P) + (K_{pw}K_{qv} - K_{qw}K_{pv})V\right.\\
&\qquad \left. + K_{pw}(Q_0 + Q_1 + Q) + K_{pw}K_{qv2}V^2\right].
\end{aligned} \tag{10}$$

In these equations, the state variables are defined as follows. δ_m is the generator voltage angle phase, ω is the rotor speed, δ is the load voltage phase angle, and V is the load voltage magnitude. The functions P and Q appearing in these equations represent, respectively, the active and reactive powers supplied to the load by the network. They are given by

$$\begin{aligned}
P &= -VE_0'Y_0'\sin(\delta+\theta_0') - VE_mY_m\sin(\delta-\delta_m+\theta_m)\\
&\quad + V^2(Y_0'\sin\theta_0' + Y_m\sin\theta_m),\\
Q &= VE_0'Y_0'\cos(\delta+\theta_0') + VE_mY_m\cos(\delta-\delta_m+\theta_m)\\
&\quad - V^2(Y_0'\cos\theta_0' + Y_m\cos\theta_m).
\end{aligned} \tag{11}$$

In the above equations, instead of including the capacitor in the circuit, a Thevenin equivalent circuit with the capacitor has been derived with values given by

$$\begin{aligned}
E_0' &= \frac{E_0}{\sqrt{1 + C^2Y_0^{-2} - 2CY_0^{-1}\cos\theta_0}},\\
Y_0' &= Y_0\sqrt{1 + C^2Y_0^{-2} - 2CY_0^{-1}\cos\theta_0},\\
\theta_0' &= -\frac{\pi}{2} + \tan^{-1}\frac{C - Y_0\cos\theta_0}{-Y_0\sin\theta_0}.
\end{aligned} \tag{12}$$

Other quantities appearing in (10)–(12) are constant parameters, relating to either the load or the network and generator. All of these parameters are fixed during the analysis, except for Q_1, the reactive power demand of the load. The load, network, and generator parameters are given in Table 1.

The bifurcation diagram for the system, relating the voltage magnitude V to the bifurcation parameter Q_1 (reactive power load), is obtained by the proposed approach. In order to show the validity of this result, the same analysis is carried out employing the continuation/bifurcation software package XPPAUT [24].

For the electrical network, the particle swarm optimization algorithm was set up with a swarm of 500 particles, 20-particle admittance for species, and $r = 0.1$. The intervals for the four variables for the electrical network are shown in Table 2. The initial value $Q_1 = 10$ was incremented in $\Delta Q_1 = 0.01$ until a Q_1 value of 11.5 is reached. For each Q_{1k}, the PSO was run for 100 cycles. The fixed points that were found for electrical network with the PSO and the bifurcation diagram generated by the XPPAUT software package are shown in Figure 3.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we report our research work on the application of intelligent optimization methods, namely, particle swarm optimization, to the production of bifurcation diagrams. Bifurcation diagrams are a tool for stability analysis of dynamic systems, with applications to electrical power systems, biology, chemistry, economics, and so forth.

The obtained results correspond to those produced by XPPAUT [24], which is the standard tool for producing bifurcation diagrams. This fact validates our results in the sense that the bifurcation diagrams produced with our method are as good approximations to the ideal ones as those produced by XPPAUT.

Table 1: Constant values for the symbols in the differential equation system.

Symbol	Value	Symbol	Value	Symbol	Value
K_{pw}	0.4	K_{pv}	0.3	K_{qw}	−0.03
K_{qv2}	2.1	T	8.5	K_{qv}	−2.8
P_0	0.6	P_1	0	Q_0	1.3
E'_0	2.5	P_m	1	E_m	1
M	0.3	Y_m	5	Y'_0	8
θ'_0	−0.2094	Q_1	10	θ_m	−0.08726
D_m	0.05	—	—	—	—

Table 2: Values of the ranges of search for variables.

Variable	Range	Variable	Range	Variable	Range	Variable	Range
δ_m	$[0, 1]$	ω	$[-1, 1]$	δ	$[0, 1]$	V	$[0, 2]$

Comparing our results with those produced by XPPAUT, the proposed method presents several advantages.

(1) XPPAUT requires 14 parameters to produce a bifurcation diagram, while ours depends only on four $(r, \Delta r, N_p, n)$.

(2) The sensitivity of XPPAUT to those parameters is very high, while our method is sensitive only to r, the radius. This dependence, though, affects only the accuracy of the results. If those parameters are not set right, XPPAUT may not produce any diagram at all.

(3) Besides those parameters, XPPAUT needs a stable fixed point to start a bifurcation diagram. If the engineer/scientist does not count with one, the system cannot produce any results. Our approach does not need initial conditions at all.

(4) In terms of time, XPPAUT is faster than our method. For a given dynamic system, XPPAUT may produce results in, let us say, 15 seconds, while ours may take 15 minutes. Nevertheless, taking into account the setting of all parameters that XPPAUT needs for operation, it may take a student/engineer about 3 days to complete an experiment. Using our system, the total time to produce results is still 15 minutes.

(5) XPPAUT uses a continuation method (that is why it needs initial conditions to start with) and produces only a segment of the total bifurcation diagram, stopping where it finds a bifurcation. Our approach produces the whole bifurcation diagram for the specified region.

(6) An additional effect of using our approach is to further the automation of the production of bifurcation diagrams, since they can be produced without any human intervention, other than the specification of the system (through a set of differential equations). Using XPPAUT, on the other hand, the user needs to specify new initial conditions to produce more segments of the diagram, or to select bifurcation points to continue plotting a given segment.

(7) Finally, XPPAUT produces only bifurcation diagrams of systems with one parameter (two-dimensional diagrams). Using our approach, we have already produced bifurcation diagrams for systems with more than one parameter, being able to plot them for 2 parameters varying at the same time (three-dimensional diagrams).

There are several directions to work on. You may have noticed from Figure 3 that the parabole in the bifurcation diagram presents a discontinuity; this is caused by the radius value needed to determine to which species a particle belongs. Since the method is based on the radius of species, it can only detect one solution per species, and since there is one species per region of radius r, their representative individuals must be at least at a distance r from each other. So, the method cannot detect solutions appearing closer than the radius r, even though regions may intersect. Therefore, we are missing solutions in areas where they become very close together. From a practical viewpoint, the closeness of the points does not have a practical implication given that the bifurcation region has been determined, as well as solution trajectories emanating from the bifurcation point. Hence, further calculations, such as stability analysis and computation of doubling period branches, can be done from those fixed points composing the solution trajectories. The mentioned problem can be fixed by using a variable-radius strategy, similar to the variable-step integration methods [25]. Another approach to solve this problem is to use the convex hull method presented by Barrera and Flores in [26].

One feature found in XPPAUT and not in our implementation is the determination of period-doubling fixed points. Those points correspond to complex roots of the characteristic equation of the differential equation. Those solutions represent limit cyclic or strange attractor solutions. The extension to determine those is, in principle, straightforward. By extending the search space to include imaginary components in the roots, the same search procedure will be able to find real and imaginary roots. We do expect to find detail that will need refinement, once this extension is implemented.

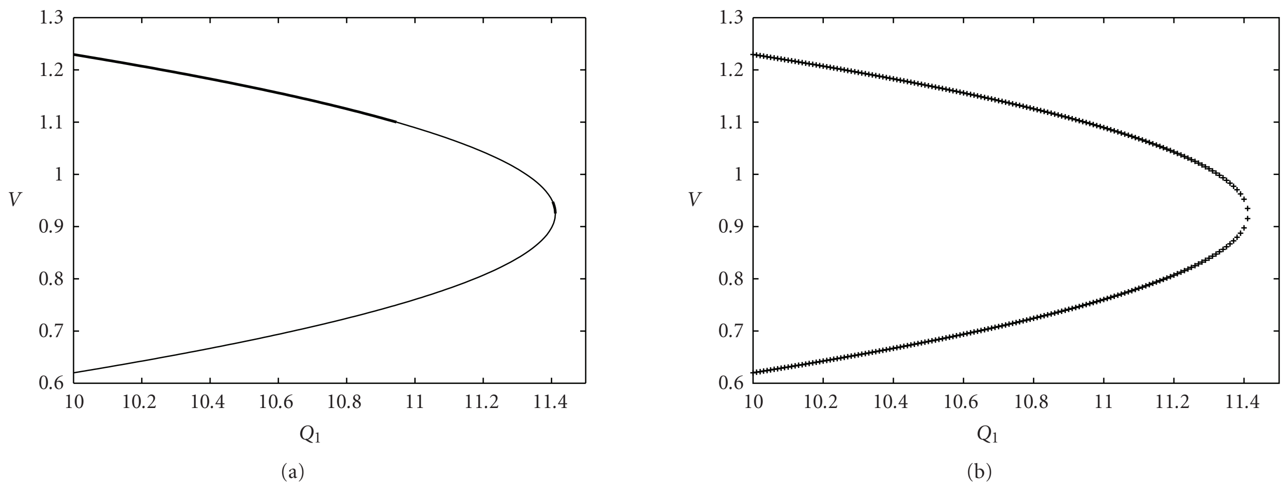


FIGURE 3: Bifurcation diagrams obtained (a) with the XPPAUT software package, and (b) hybrid particle swarm optimization.

Another feature not included in our implementation is discerning between stable and unstable fixed points. The determination of the nature of the fixed points has been in the literature (see [22]) for a while; so it will not present an obstacle.

In summary, we will continue this area of research, with the goals of producing alternative solutions to mathematical and engineering problems in the area of dynamical systems, and furthering the automation of the overall process, leaving more work to the computer and less to the scientist. As in all applications of computer science, the idea is not to replace the human being behind this process, but to free him/her from tedious work providing more time to do more productive work.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN '95)*, pp. 1942–1948, Perth, Australia, December 1995.

[2] B. Ermentrout, Xppaut5.96—the differential equations tool, 2006.

[3] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[4] IEEE Working Group, "Voltage stability of power systems: concepts, analytical tools, and industry experience," Tech. Rep. IEEE 90 TH 0358-2-PWR, IEEE Power Systems Engineering Committee, New York, NY, USA, 1990.

[5] T. V. Cutsem and C. Vournas, *Voltage Stability of Electric Power Systems*, Power Electronics and Power System Series, Springer, New York, NY, USA, 1998.

[6] E. H. Abed and P. P. Varaiya, "Nonlinear oscillations in power systems," *International Journal of Electrical Power and Energy System*, vol. 6, no. 1, pp. 37–43, 1984.

[7] H. Kwatny, A. Pasrija, and L. Bahar, "Static bifurcations in electric power networks: loss of steady-state stability and voltage collapse," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 10, pp. 981–991, 1986.

[8] J. S. T. Thorp, I. Dobson, and H. D. Chiang, "A model of voltage collapse in electric power systems," in *Proceedings of the 27th IEEE Conference on Decision and Control (CDC '88)*, vol. 3, pp. 2104–2109, Austin, Tex, USA, December 1988.

[9] V. Ajjarapu and B. Lee, "Bifurcation theory and its application to nonlinear dynamical phenomena in an electrical power system," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 424–431, 1992.

[10] E. H. Abed, J. C. Alexander, H. Wang, A. M. A. Hamdan, and H. C. Lee, "Dynamic bifurcations in a power system model exhibiting voltage collapse," Tech. Research Report, Systems Research Center, Maryland University, College Park, Md, USA, February 1992.

[11] H. D. Chiang, T. P. Connen, and A. J. lueck, "Bifurcation and chaos in electric power systems," *Journal of the Franklin Institute*, vol. 33, no. 6, pp. 1001–1036, 1994.

[12] C. A. Canizares, "On bifurcations, voltage collapse and load modeling," *IEEE Transactions on Power Systems*, vol. 10, no. 1, pp. 512–522, 1995.

[13] D. J. Hill, "Special issue on nonlinear phenomena in power systems: theory and practical impli," *Proceedings of the IEEE*, vol. 83, no. 11, p. 1439, 1995.

[14] R. Seydel, *Practical Bifurcation and Stability Analysis From Equilibrium to Chaos*, Springer, New York, NY, USA, 2nd edition, 1994.

[15] S. Richter and R. DeCarlo, "Continuationmethods: theory and applications," *IEEE Transactions on Circuits and Systems*, vol. 30, no. 6, pp. 347–352, 1983.

[16] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Proceedings of the 8th Annual Conference Genetic and Evolutionary Computation (GECCO '06)*, vol. 1, pp. 51–58, New York, NY, USA, July 2006.

[17] J. -P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, 2002.

[18] A. Petrowski, "A new selection operator dedicated to speciation," in *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA '97)*, T. Bäck, Ed., pp. 144–151, Morgan Kaufmann, San Francisco, Calif, USA, July 1997.

[19] S. W. Mahfoud, *Niching methods for genetic algorithms*, Ph.D. thesis, Genetic Algorithms Laboratory, University of Illinois at Uraban-Champaign, Urbana, Ill, USA, 1995.

[20] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[21] S. H. Strogatz and S. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*, Perseus Books, New York, NY, USA, 2000.

[22] Y. A. Kuznetsov, *Elements of Applied Bifurcation Theory*, Springer, New York, NY, USA, 1998.

[23] K. Walve, “Modeling of power system components at severe disturbances,” in *Proceedings of the 15th International Conference on Large High Voltage Electric Systems*, pp. 38–18, Ljubljana, Slowenia, August 1986.

[24] E. Doedel, A. Champneys, T. Fairgrieve, Y. Kuznetsov, B. Sandstede, and X. Wang, “Auto97: continuation and bifurcation software for ordinary differential equations,” User’s Guide, Concordia University, Montreal, Canada, 1997.

[25] W. J. F. Govaerts, *Numerical Methods for Bifurcations of Dynamic Equilibria*, SIAM, Philadelphia, Pa, USA, 2000.

[26] J. Barrera and J. J. Flores, “Multimodal optimization by decomposition of the search space in regions,” in *Proceedings of the 7th International Conference on Intelligent System Design and Applications (ISDA ’07)*, pp. 863–868, Rio de Janeiro, Brazil, October 2007.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 861512, 9 pages
doi:10.1155/2008/861512

Research Article

A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem

Ali R. Guner[1] and Mehmet Sevkli[2]

[1] *Department of Industrial and Manufacturing Engineering, College of Engineering, Wayne State University, Detroit, MI 48202, USA*
[2] *Department of Industrial Engineering, Faculty of Engineering, Fatih University, 34500 Büyükçekmece, Istanbul, Turkey*

Correspondence should be addressed to Mehmet Sevkli, msevkli@fatih.edu.tr

Received 5 July 2007; Revised 27 December 2007; Accepted 12 March 2008

Recommended by T. Blackwell

A discrete version of particle swarm optimization (*DPSO*) is employed to solve uncapacitated facility location (*UFL*) problem which is one of the most widely studied in combinatorial optimization. In addition, a hybrid version with a local search is defined to get more efficient results. The results are compared with a continuous particle swarm optimization (*CPSO*) algorithm and two other metaheuristics studies, namely, genetic algorithm (*GA*) and evolutionary simulated annealing (*ESA*). To make a reasonable comparison, we applied to same benchmark suites that are collected from *OR*-library. In conclusion, the results showed that *DPSO* algorithm is slightly better than *CPSO* algorithm and competitive with *GA* and *ESA*.

1. INTRODUCTION

Efficient supply chain management has led to increased profit, increased market share, reduced operating cost, and improved customer satisfaction for many businesses. One strategic decision in supply chain management is facility location [1]. Location problems are classified into categories with some assumptions such as limiting the capacity and open number of sites. The uncapacitated facility location (*UFL*) problem assumes the cost of satisfying the client requirements has two components: a fixed cost of setting up a facility in a given site, and a transportation cost of satisfying the customer requirements from a facility. The capacities of all the facilities are assumed to be infinite [2].

1.1. Literature review

There are many different titles for the *UFL* problem in the literature: the problem of a nonrecoverable tools optimal system [3], the standardization and unification problem [4], the location of bank accounts problem [5], warehouse location problem [6], uncapacitated facility location problem [7], and so on. The academic interest to investigate this mathematical model reasoned different interpretations. *UFL* problem is one of the most widely studied problems in combinatorial optimization problems thus there is a very rich literature in operations research (*OR*) for this kind of problem [8]. All important approaches relevant to *UFL* problems can be classified into two main categories: exact algorithms and metaheuristics-based methods.

There is a variety of exact algorithms to solve the *UFL* problem, such as branch and bound [6, 9], linear programming [10], Lagrangean relaxation algorithms [11], dual approach (*DUALLOC*) of Erlenkotter [12], and the primal-dual approaches of Körkel [13]. Although Erlenkotter [12] developed this dual approach as an exact algorithm, it can also be used as a heuristic to find good solutions. It is obvious that since the *UFL* problem is NP-hard [14], exact algorithms may not be able to solve large practical problems efficiently. There are several studies to solve *UFL* problem with heuristics and metaheuristics methods. Alves and Almeida [15] proposed a simulated annealing algorithms and reported they produce high-quality solutions, but quite expensive in computation times. A new version of evolutionary simulated annealing algorithm (*ESA*) called distributed *ESA* presented by Aydin and Fogarty [16]. They stated that with implementing it they get good quality of solutions within short times. Another popular metaheuristic, tabu search algorithm, is applied by Al-Sultan and Al-Fawzan in [17]. Their application produces good solutions, but takes significant computing time and limits the applicability of the algorithm. Michel and Van Hentenryck [18] also applied tabu search and their proposed algorithm generates more robust solutions. Sun [19] examined tabu search procedure against the Lagrangean method and heuristic procedures reported by Ghosh [2]. Genetic algorithms (*GA*) are also applied by Kratica and Jaramillo [20, 21]. Finally, there are also artificial neural network approaches to solve *UFL* problems in Gen et al. [22] and Vaithyanathan et al. [23].

The particle swarm optimization (*PSO*) is one of the recent metaheuristics invented by Eberhart and Kennedy [24] based on the metaphor of social interaction and communication such as bird flocking and fish schooling. On one hand, it can be counted as an evolutionary method with its way of exploration via neighborhood of solutions (particles) across a population (swarm) and exploiting the generational information gained. On the other hand, it is different from other evolutionary methods in such a way that it has no evolutionary operators such as crossover and mutation. Another advantage is its ease of use with fewer parameters to adjust. In *PSO*, the potential solutions, the so-called particles, move around in a multidimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. *PSO* has been successfully applied to a wide range of applications such as function optimization, neural network training [25], task assignment [26], and scheduling problem [27, 28].

Since *PSO* is developed for continuous optimization problem initially, most existing *PSO* applications are resorting to continuous function value optimization [29–32]. Recently, a few researches applied *PSO* for discrete combinatorial optimization problems [26–28, 33–37].

1.2. UFL problem definition

In a *UFL* problem, there are a number of customers, m, to be satisfied by a number of facilities, n. Each facility has a fixed cost, fc_j. A transport cost, c_{ij}, is accrued for serving customer, i, from facility, j. There is no limit of capacity for any candidate facility and the whole demand of each customer has to be assigned to one of the facilities. We are asked to find the number of facilities to be established and specify those facilities such that the total cost will be minimized (1). The mathematical formulation of the problem can be stated as follows [14]:

$$Z = \min\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\cdot x_{ij} + \sum_{j=1}^{n} fc_j \cdot y_j\right), \qquad (1)$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \text{ in } m, \qquad (2)$$

$$0 \le x_{ij} \le y_j, \quad y_j \in \{0;1\}, \qquad (3)$$

where $i = 1,\dots,m$; $j = 1,\dots,n$; x_{ij} represents the quantity supplied from facility i to customer j; y_j indicates whether facility j is established ($y_j = 1$) or not ($y_j = 0$).

Constraint (2) makes sure that all customers demands have been met by an open facility and (3) is to keep integrity. Since it is assumed that there is no capacity limit for any facility, the demand size of each customer is ignored, and therefore (2) established without considering demand variable.

It is obvious that since the main decision in *UFL* is opening or closing facilities, *UFL* problems are classified in discrete problems. On the other hand, *PSO* is mainly designed for continuous problem thus it has some drawbacks when applying *PSO* for a discrete problem. This tradeoff increased our curiosity to apply *PSO* algorithm for solving *UFL* problem.

The organization of the paper is as follows: in Section 2, the implementation of both continuous and discrete *PSO* algorithms for *UFL* problem is given with the details of how a local search procedure is embedded. Section 3 reports the experimental settings and results. There are three sets of comparisons: the first is between *CPSO* and *DPSO* algorithms; the second is between *CPSO* with local search ($CPSO_{LS}$) and *DPSO* with local search ($DPSO_{LS}$) algorithms; and the third is among $DPSO_{LS}$ with two other algorithms from the literature. Finally, Section 4 provides with the conclusion.

2. PSO ALGORITHMS FOR UFL PROBLEM

As mentioned Section 1, *PSO* is one of the population-based optimization technique inspired by nature. It is a simulation of social behaviour of a swarm, that is, bird flocking, fish schooling. Suppose the following scenario: a flock of bird is randomly searching for food in an area, where there is only one piece of food available and none of them knows where it is, but they can estimate how far it would be at each iteration. The problem here is "what is the best strategy to find and get the food?". Obviously, the simplest strategy is to follow the bird known as the nearest one to the food. *PSO* inventers were inspired of such natural process-based scenarios to solve the optimization problems. In *PSO*, each single solution, called a particle, is considered as a bird, the group becomes a swarm (population) and the search space is the area to explore. Each particle has a fitness value calculated by a fitness function, and a velocity of flying towards the optimum, the food. All particles fly across the problem space following the particle nearest to the optimum. *PSO* starts with initial population of solutions, which is updated iteration-by-iteration. Therefore, *PSO* can be counted as an evolutionary algorithm besides being a metaheuristics method, which allows exploiting the searching experience of a single particle as well as the best of the whole swarm.

2.1. Continuous PSO algorithm for UFL problem

The continuous particle swarm optimization (*CPSO*) algorithm used here is proposed by the authors, Sevkli and Guner [33]. The *CPSO* considers each particle has three key vectors: position (X_i), velocity (V_i), and open facility (Y_i). X_i denotes the ith position vector in the swarm,

Table 1: An illustration of deriving open facility vector from position vector for a 6-customer 5-facility problem.

ith particle vectors	Particle dimension (k)				
	1	2	3	4	5
Position vector (X_i)	1.8	3.01	−0.99	0.72	−5.45
Velocity vector (V_i)	−0.52	2.06	3.56	2.45	−1.44
Open facility vector (Y_i)	1	1	0	0	1

$X_i = [x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}]$, where x_{ik} is the position value of the ith particle with respect to the kth dimension ($k = 1, 2, 3, \ldots n$). V_i denotes the ith velocity vector in the swarm, $V_i = [v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in}]$, where v_{ik} is the velocity value of the ith particle with respect to the kth dimension. Y_i represents the opening or closing facilities based on the position vector (X_i), $Y_i = [y_{i1}, y_{i2}, y_{i3}, \ldots, y_{in}]$, where y_{ik} represents opening or closing kth facility of the ith particle. For an *n-facility* problem, each particle contains n number of dimensions.

Initially, the position and the velocity vectors are generated as continuous uniform random variables, using the following rules:

$$x_{ij} = x_{\min} + (x_{\max} - x_{\min}) \times r_1, \qquad v_{ij} = v_{\min} + (v_{\max} - v_{\min}) \times r_2, \tag{4}$$

where $x_{\min} = -10.0$, $x_{\max} = 10.0$, $v_{\min} = -4.0$, $v_{\max} = 4.0$ which are consistent with the literature [38], and r_1 and r_2 are uniform random numbers in [0, 1] for each dimension and particle. The position vector $X_i = [x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}]$ corresponds to the continuous position values for the n facilities, but it does not represent a candidate solution to calculate a total cost (fitness value). In order to create a candidate solution, a particle, the position vector is converted to a binary variable, $Y_i \leftarrow X_i$, which is also a key element of a particle. In other words, a continuous set is converted to a discrete set for the purpose of creating a candidate solution, particle. The fitness of the ith particle is calculated by using open facility vector (Y_i). For simplicity, let $f_i(Y_i \leftarrow X_i)$ be denoted as f_i.

In order to ascertain how to derive an open facility vector from position vector, an instance of 5-facility problem is illustrated in Table 1. Position values are converted to binary variables using following formula:

$$y_i = \lfloor |x_i| (\text{mod } 2) \rfloor. \tag{5}$$

In (5), the absolute value of a position value is first divided by 2 and then the remainder is floored to nearest integer number. Then it is assigned to corresponding element of the open facility vector. For example, fifth element of the open facility vector, y_5, can be calculated as follows:

$$\lfloor |-5.45| (\text{mod } 2) \rfloor = \lfloor 5.45 (\text{mod } 2) \rfloor = \lfloor 1.45 \rfloor = 1. \tag{6}$$

Table 2: An example of 5-facility to 6-customer.

Facility locations		1	2	3	4	5
Fixed cost		12	5	3	7	9
Customers	1	2	3	6	7	1
	2	0	5	8	4	12
	3	11	6	14	5	8
	4	19	18	21	16	13
	5	3	9	8	7	10
	6	4	7	9	6	0

Considering the 5-facility to 6-customer example shown in Table 2, the total cost of *open facility vector* (Y_i) can be calculated as follows:

$$\begin{aligned} &\text{total cost} \\ &\quad = \{\text{open facilities fixes costs } (fc_j) \\ &\qquad + \min (\text{cost of supply from open} \\ &\qquad\qquad \text{facilities to customer } i[ci_j])\} \\ &\quad \cdot \{(12+5+9) + \min(2, 3, 1) \\ &\qquad + \min(0, 5, 12) + \min(11, 6, 8) \\ &\qquad + \min(19, 18, 13) + \min(3, 9, 10) + \min(4, 7, 0)\} \\ &\quad = \{(26) + (1+0+6+13+3+0)\} \\ &\quad = \{26 + 23\} = \{49\}. \end{aligned} \tag{7}$$

For each particle in the swarm, let define $P_i = [p_{i1}, p_{i2}, \ldots, p_{in}]$, as the personal best, where p_{ik} denotes the position value of the ith personal best with respect to the kth dimension. The personal bests are determined just after generating Y_i vectors and their corresponding fitness values. In every generation, the personal best of each particle is updated based on its position vector and fitness value. Regarding the objective function, $f_i(Y_i \leftarrow X_i)$, the fitness values for the personal best of the ith particle, P_i, is denoted by $f_i^{pb} = f(Y_i \leftarrow P_i)$. At the beginning, the personal best values are equal to position values ($P_i = X_i$), explicitly $P_i = [p_{i1} = x_{i1}, p_{i2} = x_{i2}, p_{i3} = x_{i3}, \ldots, p_{in} = x_{in}]$ and the fitness values of the personal bests are equal to the fitness of positions, $f_i^{pb} = f_i$.

Then the best particle in the whole swarm is selected as the global best. $G = [g_1, g_2, g_3, \ldots, g_n]$ denotes the best position of the globally best particle, $f_g = f(Y \leftarrow G)$, achieved so far in the whole swarm. At the beginning, global best fitness value is determined as the best of personal bests over the whole swarm, $f_g = \min\{f_i^{pb}\}$, with its corresponding position vector X_g, which is to be used for $G = X_g$, where $G = [g_1 = x_{g1}, g_2 = x_{g2}, g_3 = x_{g3}, \ldots, g_n = x_{gn}]$ and corresponding $Y_g = [y_{g1}, y_{g2}, y_{g3}, \ldots, y_{gn}]$ denotes the open facility vector of the global best found.

```
Begin
  Initialize particles (population) randomly
  For each particle
   Calculate open facility vectors (1)
   Calculate fitness value using open facility vector
   Set to position vector and fitness value
   as personal best ($P_i^t$)
   Select the best particle and its position
   vector as global($G^t$)
  End
  Do {
   Update inertia weight
   For each particle
     Update velocity (8)
     Update position(9)
     Find open facility vectors
     Calculate fitness value using open
     facility vector (1)
     Update personal best($P_i^t$)
     Update the global best ($G^t$) value with
     position vector
    End
   Apply local search (for $CPSO_{LS}$) to global best
  } While (Maximum Iteration is not reached)
End
```

Algorithm 1: Pseudocode of *CPSO* algorithm for *UFL* problem.

The velocity of each particle is updated based on its personal best and the global best in the following way:

$$v_{ik}(t+1) = w \cdot v_{ik}(t) + c_1 r_1 (p_{ik}(t) - x_{ik}(t)) + c_2 r_2 (g_k(t) - x_{ik}(t)), \tag{8}$$

where w is the inertia weight used to control the impact of the previous velocities on the current one, t is generation index, r_1 and r_2 are different random numbers for each dimension and particle in [0, 1], and c_1 and c_2 are the learning factors which are also called social and cognitive parameters. The next step is to update the positions.

$$x_{ik}(t+1) = x_{ik}(t) + v_{ik}(t+1). \tag{9}$$

After updating position values for all particles, the corresponding open facility vectors can be determined by their fitness values in order to start a new iteration if the predetermined stopping criterion has not met yet. In this study, we employed the *gbest* model of Kennedy et al. [38] for *CPSO*, which is elaborated in the pseudocode given below.

2.2. Discrete PSO algorithm for UFL problem

The discrete *PSO* (*DPSO*) algorithm used here is first proposed by Pan et al. [37] for the no-wait flowshop scheduling problem. We employed the DPSO algorithm for UFL problem. In DPSO, each particle is based on only the open facility vector $Y_i = [y_{i1}, y_{i2}, y_{i3}, \ldots, y_{in}]$, where y_{ik} represents opening or closing kth facility of the ith particle. For an *n-facility* problem, each particle contains n number of dimensions. The dimensions of Y_i are binary random numbers. The fitness of the ith particle is calculated by $f_i(Y_i)$.

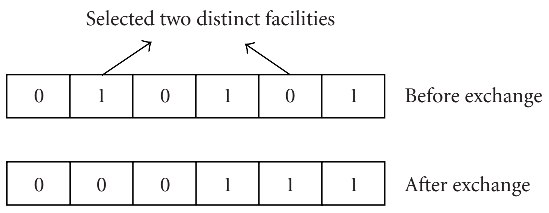


Figure 1: Exchange operator.

The open facility vector (Y_i) of the particle i at iteration t can be updated as follows [37]:

$$Y_i^t = c_2 \oplus F_3(c_1 \oplus F_2(w \oplus F_1(Y_i^{t-1}), P_i^{t-1}), G^{t-1}), \tag{10}$$

$$\lambda_i^t = w \oplus F_1(Y_i^{t-1}). \tag{11}$$

Equation (10) consists of three components: the first component (11) is the *velocity* of the particle. F_1 represents the exchange operator (Figure 1) which is selecting two distinct facilities from the open facility vector, Y_i^{t-1}, of particle and swapping randomly with the probability of w. In other words, a uniform random number, r, is generated between 0 and 1. If r is less than w then the exchange operator is applied to generate a perturbed Y_i vector of the particle by $\lambda_i^t = F_1(Y_i^{t-1})$, otherwise current Y_i is kept as $\lambda_i^t = Y_i^{t-1}$.

$$\delta_i^t = c_1 \oplus F_2(\lambda_i^t, P_i^{t-1}). \tag{12}$$

The second component (12) is the *cognition* part of the particle representing particle's own experience. F_2 represents the one-cut crossover (Figure 2) with the probability of c_1. Note that λ_i^t and P_i^{t-1} will be the first and second parents for the crossover operator, respectively. It is resulted either in $\delta_i^t = F_2(\lambda_i^t, P_i^{t-1})$ or in $\delta_i^t = \lambda_i^t$ depending on the choice of a uniform random number

$$X_i^t = c_2 \oplus F_3(\delta_i^t, G^t). \tag{13}$$

The third component (13) is the *social* part of the particle representing experience of whole swarm. F_3 represents the two-crossover (Figure 3) operator with the probability of c_2. Note that δ_i^t and G^{t-1} will be the first and second parents for the crossover operator, respectively. It is resulted either in $Y_i^t = F_3(\delta_i^t, G^{t-1})$ or in $Y_i^t = \delta_i^t$ depending on the choice of a uniform random number. In addition, one-cut and two-cut crossovers produce two children. In this study, we selected one of the children randomly.

The corresponding Y_i vectors are determined with their fitness values so as to start a new iteration if the predetermined stopping criterion has not met yet. We apply the *gbest* model of Kennedy and Eberhart for DPSO. The pseudocode of DPSO is given in Algorithm 2.

2.3. Local search for CPSO and DPSO algorithm

Apparently, *CPSO* and *DPSO* conduct such a rough search that they produce premature results, which do not offer satisfactory solutions. For this reason, it is inevitable to embed

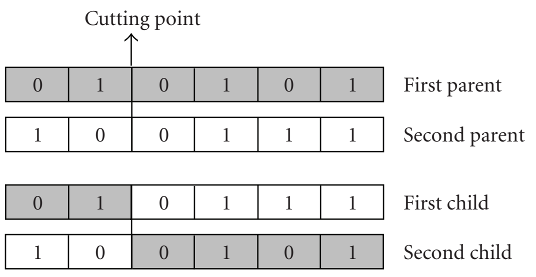


Figure 2: One-cut crossover operator.

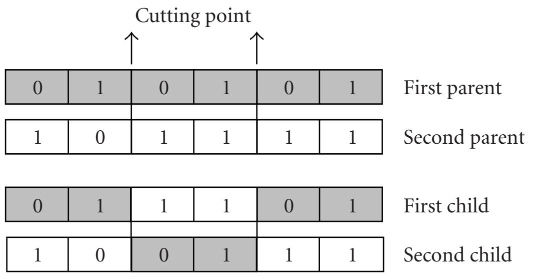


Figure 3: Two-cut crossover operator.

Table 3: Benchmarks tackled with the sizes and the optimum values.

Benchmarks		
Problems	Size ($m \times n$)	Optimum
Cap71	16×50	932615.75
Cap72	16×50	977799.40
Cap73	16×50	1010641.45
Cap74	16×50	1034976.98
Cap101	25×50	796648.44
Cap102	25×50	854704.20
Cap103	25×50	893782.11
Cap104	25×50	928941.75
Cap131	50×50	793439.56
Cap132	50×50	851495.33
Cap133	50×50	893076.71
Cap134	50×50	928941.75
CapA	100×1000	17156454.48
CapB	100×1000	12979071.58
CapC	100×1000	11505594.33

a local search algorithm to *CPSO* and *DPSO* so as to produce more satisfactory solutions. In this study, we have applied a simple local search method to neighbours of the global best particle in every generation. In *CPSO* global best has three vectors, so local search is applied to the position vector (X_i). Since *DPSO* has one vector. Local search is applied only this vector (Y_i).

The neighbourhood structure with which neighbour solutions are determined to move is one of the key elements in metaheuristics. The performance of the hybrid algorithm depends on the efficiency of the neighbourhood structure. For both algorithms, flip operator is employed as a neighbourhood structure. It is defined as: picking randomly one position value (i) of the global best, then changing its value with using formula (14) for $CPSO_{LS}$ and formula (15) for $DPSO_{LS}$. Since binary and continuous values are stored in Y_i and X_i vectors, respectively, the equations are slightly different

```
Begin
  Initialize open facility vector
  Calculate fitness value using $Y_i$(1)
  Do
    Find personal best ($P_i^t$)
    Find global best ($G^t$)
    For Each Particle
      Apply velocity component (11)
      Apply cognition component (12)
      Apply social component (13)
      Calculate fitness value using $Y_i$(1)
    Apply local search (for $DPSO_{LS}$)
    to global best
  While (Maximum Iteration is not reached)
End
```

Algorithm 2: Pseudocode of *DPSO* algorithm for *UFL* problem.

$$g_i \longleftarrow g_i + 1, \tag{14}$$

$$g_i \longleftarrow 1 - g_i. \tag{15}$$

The local search algorithm applied in this study is sketched in Algorithm 3. The global best found at the end of each iteration of *CPSO* and *DPSO* is adopted as the initial solution by the local search algorithm. In order not to lose the best found and to diversify the solution, the global best is modified with two facilities (η and κ) which are randomly chosen. Then flip operator is applied to as long as it gets a better solution. The final produced solution, s **,** is replaced with the old global best if its fitness value is better than the initial one.

3. EXPERIMENTAL RESULTS

The experimental study has been completed in three stages; first, we compared the *CPSO* and *DPSO* algorithms without local search, then we compared these algorithms with local search ($CPSO_{LS}$ and $DPSO_{LS}$) with respect to their solution quality; finally, $DPSO_{LS}$ results are compared with other two metaheuristics, namely, genetic algorithm (*GA*) and evolutionary simulated annealing algorithm (*ESA*). Experimental results provided in this section are carried out over 15 benchmark problems well-known by the researchers of *UFL problem*. The benchmarks are undertaken from the *OR library* [39], a collection of benchmarks for operations research (*OR*) studies. There are currently 15 *UFL* test problems in the *OR*-library. Among these test problems, 12 are relatively small in size ranging from $m \times n = 50 \times 16$ to $m \times n = 50 \times 50$. The other three are relatively large with $m \times n = 1000 \times 100$. The benchmarks are introduced in Table 3 with their optimum values. Although the optimum

Table 4: Experimental results gained for *CPSO* and *DPSO* without local search.

	CPSO			DPSO		
Problem	ARPE	HR	ACPU	ARPE	HR	ACPU
Cap71	0.026	0.83	0.1218	0.000	1.00	0.0641
Cap72	0.050	0.83	0.1318	0.000	1.00	0.0651
Cap73	0.034	0.73	0.1865	0.000	1.00	0.0708
Cap74	0.095	0.00	0.1781	0.000	1.00	0.0693
Cap101	0.183	0.00	0.8818	0.000	1.00	0.3130
Cap102	0.135	0.33	0.7667	0.000	1.00	0.3062
Cap103	0.145	0.00	0.9938	0.000	1.00	0.3625
Cap104	0.286	0.60	0.6026	0.002	0.93	0.2021
Cap131	0.911	0.00	3.6156	0.173	0.13	2.5464
Cap132	0.756	0.00	3.5599	0.090	0.17	2.6328
Cap133	0.496	0.00	3.7792	0.042	0.43	2.5292
Cap134	0.691	0.23	3.3333	0.000	1.00	1.7167
CapA	21.242	0.00	29.5739	8.654	0.00	24.8972
CapB	10.135	0.00	27.1318	4.918	0.00	22.0652
CapC	8.162	0.00	27.6149	4.545	0.00	23.1340

Table 5: Experimental results of $CPSO_{LS}$ and $DPSO_{LS}$.

	$CPSO_{LS}$			$DPSO_{LS}$		
Problem	ARPE	HR	ACPU	ARPE	HR	ACPU
Cap71	0.000	1.00	0.0146	0.000	1.00	0.0130
Cap72	0.000	1.00	0.0172	0.000	1.00	0.0078
Cap73	0.000	1.00	0.0281	0.000	1.00	0.0203
Cap74	0.000	1.00	0.0182	0.000	1.00	0.0109
Cap101	0.000	1.00	0.1880	0.000	1.00	0.1505
Cap102	0.000	1.00	0.0906	0.000	1.00	0.0557
Cap103	0.000	1.00	0.2151	0.000	1.00	0.1693
Cap104	0.000	1.00	0.0370	0.000	1.00	0.0344
Cap131	0.000	1.00	1.4281	0.000	1.00	0.4922
Cap132	0.000	1.00	1.0245	0.000	1.00	0.2745
Cap133	0.000	1.00	1.3651	0.000	1.00	0.4516
Cap134	0.000	1.00	0.3635	0.000	1.00	0.0594
CapA	0.037	1.00	16.3920	0.051	0.53	14.5881
CapB	0.327	0.63	19.6541	0.085	0.40	17.6359
CapC	0.091	0.00	17.4234	0.036	0.13	15.7685

values are known, it is really hard to hit the optima in every attempt of optimization. Since the main idea is to test the performance of *CPSO* and *DPSO* algorithm with *UFL* benchmark, the results of both algorithms are provided in Table 4 as the solution quality: *average relative percent error (ARPE)*, *hit to optimum rate,* (HR) and *average computational processing time (ACPU)* in seconds. *ARPE* is the percentage of difference from the optimum and defined as following:

$$\text{ARPE} = \sum_{i=1}^{R}\left(\frac{H_i - U}{U}\right) \times \frac{100}{R}, \tag{16}$$

where H_i denotes the ith replication solution value, U is the optimal value provided in the literature, and R is the number of replications. HR provides the ratio between the number of runs yielded the optimum and the total numbers of experimental trials.

Obviously, the higher the HR the better quality of solution, while the lower the *ARPE* the better quality. The computational time spent for *CPSO* [33] and *DPSO* cases are obtained as time to get best value over 1000 iterations, while for $CPSO_{LS}$ and $DPSO_{LS}$ cases are obtained as time to get best value over 250 iterations. All algorithms and other related software were coded with Borland C++ Builder 6 and run on an Intel Centrino 1.7 GHz PC with 512 MB memory.

There are fewer parameters used for the *DPSO* and $DPSO_{LS}$ algorithms and they are as follows: the size of the population (swarm) is the number of facilities, the social and

Table 6: Summary of results gained from different algorithms for comparison.

	Deviation from optimum [33]			Average CPU		
Problem	GA [21]	ESA [16]	DPSO$_{LS}$	GA [21]	ESA [16]	DPSO$_{LS}$
Cap71	0.00	0.00	0.000	0.287	0.041	0.0130
Cap72	0.00	0.00	0.000	0.322	0.028	0.0078
Cap73	0.00033	0.00	0.000	0.773	0.031	0.0203
Cap74	0.00	0.00	0.000	0.200	0.018	0.0109
Cap101	0.00020	0.00	0.000	0.801	0.256	0.1505
Cap102	0.00	0.00	0.000	0.896	0.098	0.0557
Cap103	0.00015	0.00	0.000	1.371	0.119	0.1693
Cap104	0.00	0.00	0.000	0.514	0.026	0.0344
Cap131	0.00065	0.00008	0.000	6.663	2.506	0.4922
Cap132	0.00	0.00	0.000	5.274	0.446	0.2745
Cap133	0.00037	0.00002	0.000	7.189	0.443	0.4516
Cap134	0.00	0.00	0.000	2.573	0.079	0.0594
CapA	0.00	0.00	0.051	184.422	17.930	14.5881
CapB	0.00172	0.00070	0.085	510.445	91.937	17.6359
CapC	0.00131	0.00119	0.036	591.516	131.345	15.7685

```
Begin
Set globalbest open facility vector ($\mathbf{Y_g}$)
to $\mathbf{s_0}$ (for D$PSO_{LS}$)
Set globalbest position vector ($\mathbf{X_g}$)
to $\mathbf{s_0}$ (for C$PSO_{LS}$)
Modify $\mathbf{s_0}$ based on $\boldsymbol{\eta}$, $\boldsymbol{\kappa}$ and set to $\mathbf{s}$
Set 0 to loop
Repeat:
    Apply Flip to $\mathbf{s}$ and get $\mathbf{s_1}$
    if ($\mathbf{f}(\mathbf{s_1}) \le \mathbf{f}(\mathbf{s_0})$)
    Replace $\mathbf{s}$ with $\mathbf{s_1}$
    else
    loop = loop + 1
Until loop < n is false.
if ($\mathbf{f}(\mathbf{s}) \le \mathbf{f}(\mathbf{s_0})$) Replace $\mathbf{Y_g}$ with $\mathbf{s}$ (for D$PSO_{LS}$)
if ($\mathbf{f}(\mathbf{s}) \le \mathbf{f}(\mathbf{s_0})$) Replace $\mathbf{X_g}$ with $\mathbf{s}$ (for C$PSO_{LS}$)
End.
```

Algorithm 3: Pseudocode for local search.

cognitive probabilities, c_1 and c_2, are set as $c_1 = c_2 = 0.5$, and inertia weight, w, is set to 0, 9. Each problem solution run is conducted for 30 replications. There were two termination criteria that have been applied for every run: first, one is getting the optimum solution, the other is reaching the maximum iteration number that is chosen for obtaining the result in a reasonable *CPU* time.

The performance of *CPSO* algorithm looks not very impressive as the results produced within the range of time over 1000 iterations. The *CPSO* search found 6 optimal solutions, whereas the *DPSO* algorithm found 12 among 15 benchmark problems. The *ARPE* index which is expected lower for good solution quality is very high for *CPSO* when applied CapA, CapB, and CapC benchmarks and none of the attempts for these benchmarks hit the optimum value. As come to the *ARPE* index of *DPSO*, it is better than the *ARPE* index of *CPSO,* but not satisfactory as expected. In term of *CPU*, *DPSO* is better than *CPSO* as well. When the results are investigated statistically with using the t-test with 99% levels of confidence, the *DPSO* produced significantly better fitness results than *CPSO* when CapA, CapB, and CapC fitness results are excluded. It may be possible to improve the solutions quality by carrying on with algorithms for a further number of iterations, but, then the main idea and useful motivation of employing the heuristics, that is, getting a better quality within shorter time, will be lost. This fact imposed that it is essential to empower *CPSO* and *DPSO* with hybridizing with a local search algorithm. Thus a simple local search algorithm is employed in this case for that purpose, as mentioned before.

The results of *CPSO$_{LS}$* and *DPSO$_{LS}$* are shown in Table 5. The performance of *CPSO$_{LS}$* and *DPSO$_{LS}$* looks very impressive compared to the *CPSO* and the *DPSO* algorithms with respect of all three indexes. *HR* is 1.00 which means 100% of the runs yield with optimum for all benchmark except CapB and CapC for *CPSO$_{LS}$* and except CapA, CapB, and CapC for *DPSO$_{LS}$*. On the other hand, it should be mentioned that *DPSO$_{LS}$* found optimum solutions of all instances, while *CPSO$_{LS}$* found optimums except for CapC. The *ARPE* index results of *CPSO$_{LS}$* and *DPSO$_{LS}$* are very small for both algorithms and very similar to each other thus there is no meaningful difference. When the results are compared statistically with using the t-test with 99% levels of confidence, the *CPSO$_{LS}$* and *DPSO$_{LS}$* can be considered as equal. In term of *CPU*, *CPSO$_{LS}$* consumed 18% more time than *DPSO$_{LS}$* thus we can say that the results of *DPSO$_{LS}$* are slightly better than *CPSO$_{LS}$*.

The experimental study is also carried out as a comparative work in Table 6. A genetic algorithm (*GA*) introduced by Jaramillo et al. [21] and an evolutionary simulated annealing algorithm (*ESA*) proposed by Aydin and Fogarty [16] are taken for the comparison. These algorithms were coded and run under the same condition with the $DPSO_{LS}$ algorithm. The performance of $DPSO_{LS}$ algorithm looks slightly better than *GA* in both two indexes. Especially, in respect of *CPU* time $DPSO_{LS}$ much more better than *GA*. Comparing with *ESA,* both algorithms deviations from optimum are very similar. However, especially for CapA, CapB, and CapC; *GA* and *ESA* consume more *CPU* time than $DPSO_{LS}$ algorithm.

4. CONCLUSION

In this paper, one of the recent metaheuristics algorithms called *DPSO* is applied to solve *UFL* benchmark problems. The algorithm has been tested on several benchmark problem instances and optimal results are obtained in a reasonable computing time. The results of *DPSO* with local search ($DPSO_{LS}$) are compared with results of a *CPSO* [33] with local search ($CPSO_{LS}$) and two other metaheuristics approaches, namely, *GA* [21] and *ESA* [16]. It is concluded that the $DPSO_{LS}$ is slightly better than the $CPSO_{LS}$ and competitive with GA and ESA.

The main purpose of this paper is testing performance of *CPSO* and *DPSO* algorithms under the same condition. When compared *CPSO*, *DPSO* proves to be a better algorithm for *UFL* problems. It also should be noted that, since *CPSO* considers each particle based on three key vectors; position (X_i), velocity (V_i), and open facility (Y_i). So, *CPSO* allocates more memory than *DPSO* for each particle. In addition, to the best our knowledge, this is the first application of discrete *PSO* algorithm applied to the *UFL* problem.

REFERENCES

[1] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi, *Designing and Managing the Supply Chain, Concepts, Strategies and Case Studies*, McGraw-Hill, Boston, Mass, USA, 2000.

[2] D. Ghosh, "Neighborhood search heuristics for the uncapacitated facility location problem," *European Journal of Operational Research*, vol. 150, no. 1, pp. 150–162, 2003.

[3] V. L. Beresnev, E. H. Gimady, and V. T. Dementev, *Extremal Problems of Standardization*, Nauka, Novosibirsk, Russia, 1978.

[4] E. H. Gimady, *The Choice of Optimal Scales in One Problem Class of Location or Standartization*, Manage Systems, Nauka, Novosibirsk, Russia, 6th edition, 1970.

[5] G. Corneuéjols, M. L. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms," *Management Science*, vol. 23, no. 8, pp. 789–810, 1977.

[6] B. M. Khumawala, "An efficient branch and bound algorithm for the warehouse location problem," *Management Science*, vol. 18, no. 12, pp. B718–B733, 1972.

[7] J. Krarup and P. M. Pruzan, "The simple plant location problem: survey and synthesis," *European Journal of Operation Research*, vol. 12, no. 1, pp. 36–81, 1983.

[8] P. B. Mirchandani and R. L. Francis, Eds., *Discrete Location Theory*, Wiley-Interscience, New York, NY, USA, 1990.

[9] A. Klose, "A branch and bound algorithm for an *UFLP* with a side constraint," *International Transactions in Operational Research*, vol. 5, no. 2, pp. 155–168, 1998.

[10] T. J. Van Roy, "A cross decomposition algorithm for capacitated facility location," *Operations Research*, vol. 34, no. 1, pp. 145–163, 1986.

[11] J. Barcelo, Å. Hallefjord, E. Fernandez, and K. Jörnsten, "Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing," *OR Spektrum*, vol. 12, no. 2, pp. 79–88, 1990.

[12] D. Erlenkotter, "A dual-based procedure for uncapacitated facility location," *Operations Research*, vol. 26, no. 6, pp. 992–1009, 1978.

[13] M. Köerkel, "On the exact solution of large-scale simple plant location problems," *European Journal of Operational Research*, vol. 39, no. 2, pp. 157–173, 1989.

[14] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, "The uncapacitated facility location problem," in *Discrete Location Theory*, pp. 119–171, John Wiley & Sons, New York, NY, USA, 1990.

[15] M. L. Alves and M. T. Almeida, "Simulated annealing algorithm for the simple plant location problem: a computational study," *Revista Investigaćão Operacional*, vol. 12, 1992.

[16] M. E. Aydin and T. C. Fogarty, "A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems," *Journal of Heuristics*, vol. 10, no. 3, pp. 269–292, 2004.

[17] K. S. Al-Sultan and M. A. Al-Fawzan, "A tabu search approach to the uncapacitated facility location problem," *Annals of Operations Research*, vol. 86, pp. 91–103, 1999.

[18] L. Michel and P. Van Hentenryck, "A simple tabu search for warehouse location," *European Journal of Operational Research*, vol. 157, no. 3, pp. 576–591, 2004.

[19] M. Sun, "Solving the uncapacitated facility location problem using tabu search," *Computers & Operations Research*, vol. 33, no. 9, pp. 2563–2589, 2006.

[20] J. Kratica, D. Tošic, V. Filipović, and I. Ljubić, "Solving the simple plant location problem by genetic algorithm," *RAIRO Operations Research*, vol. 35, no. 1, pp. 127–142, 2001.

[21] J. H. Jaramillo, J. Bhadury, and R. Batta, "On the use of genetic algorithms to solve location problems," *Computers & Operations Research*, vol. 29, no. 6, pp. 761–779, 2002.

[22] M. Gen, Y. Tsujimura, and S. Ishizaki, "Optimal design of a star-LAN using neural networks," *Computers & Industrial Engineering*, vol. 31, no. 3-4, pp. 855–859, 1996.

[23] S. Vaithyanathan, L. I. Burke, and M. A. Magent, "Massively parallel analog tabu search using neural networks applied to simple plant location problems," *European Journal of Operational Research*, vol. 93, no. 2, pp. 317–330, 1996.

[24] R. C. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[25] F. Van den Bergh and A. P. Engelbecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Computer Journal*, vol. 26, pp. 84–90, 2000.

[26] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[27] A. Allahverdi and F. S. Al-Anzi, "A *PSO* and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application," *Computers & Operations Research*, vol. 33, no. 4, pp. 1056–1080, 2006.

[28] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop

sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.

[29] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 611–616, Springer, Berlin, Germany, 1998.

[30] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Australia, November-December 1995.

[31] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 60–68, 2003.

[32] H. Yoshida, K. Kawata, Y. Fukuyama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage stability," in *Proceedings of the International Conference on Intelligent System Application to Power System (ISAP '99)*, pp. 117–121, Rio de Janeiro, Brazil, April 1999.

[33] M. Sevkli and A. R. Guner, "A continuous particle swarm optimization algorithm for uncapacitated facility location problem," in *Ant Colony Optimization and Swarm Intelligence*, vol. 4150 of *Lecture Notes in Computer Science*, pp. 316–323, Springer, Berlin, Germany, 2006.

[34] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC '97)*, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.

[35] P.-Y. Yin, "A discrete particle swarm algorithm for optimal polygonal approximation of digital curves," *Journal of Visual Communication and Image Representation*, vol. 15, no. 2, pp. 241–260, 2004.

[36] C.-J. Liao, C.-T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Computers & Operations Research*, vol. 34, no. 10, pp. 3099–3111, 2007.

[37] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Computers & Operations Research*, vol. 35, no. 9, pp. 2807–2839, 2008.

[38] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[39] J. E. Beasley, "*OR*-Library," 2005, http://people.brunel.ac.uk/mastjjb/jeb/info.html.

Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 876746, 12 pages
doi:10.1155/2008/876746

Research Article

Particle Swarm for Attribute Selection in Bayesian Classification: An Application to Protein Function Prediction

Elon S. Correa, Alex A. Freitas, and Colin G. Johnson

Computing Laboratory and Center for Biomedical Informatics, University of Kent, Canterbury CT2 7NF, UK

Correspondence should be addressed to Elon S. Correa, elonsc@yahoo.com

Received 29 July 2007; Revised 26 November 2007; Accepted 10 January 2008

Recommended by Jim Kennedy

The discrete particle swarm optimization (DPSO) algorithm is an optimization technique which belongs to the fertile paradigm of Swarm Intelligence. Designed for the task of attribute selection, the DPSO deals with discrete variables in a straightforward manner. This work empowers the DPSO algorithm by extending it in two ways. First, it enables the DPSO to select attributes for a Bayesian network algorithm, which is more sophisticated than the Naive Bayes classifier previously used by the original DPSO algorithm. Second, it applies the DPSO to a set of challenging protein functional classification data, involving a large number of classes to be predicted. The work then compares the performance of the DPSO algorithm against the performance of a standard Binary PSO algorithm on the task of selecting attributes on those data sets. The criteria used for this comparison are (1) maximizing predictive accuracy and (2) finding the smallest subset of attributes.

1. INTRODUCTION

Most of the particle swarm algorithms present in the literature deal only with continuous variables [1–3]. This is a significant limitation because many optimization problems are set in a search space featuring discrete variables. Typical examples include problems which require the ordering or arranging of discrete variables, such as scheduling or routing problems [4]. Therefore, the design of particle swarm algorithms that deal directly with discrete variables is pertinent to this field of study.

The work in [5] proposed a discrete particle swarm optimization (PSO) algorithm for attribute selection in Data Mining. Hereafter, this algorithm will be refereed to as the discrete particle swarm optimization (DPSO) algorithm. The DPSO deals directly with discrete variables, and its population of candidate solutions contains particles of different sizes—the DPSO forces each particle to carry a constant number of attributes across iterations. The DPSO algorithm interprets the concept of velocity, used in traditional PSO, as "probability;" renders velocity as a proportional likelihood; and uses this information to sample new particle positions. The motivation behind the DPSO algorithm is indeed to introduce a probability-like approach to particle swarm.

Although specifically designed for the task of attribute selection, the DPSO is not limited to this kind of application. By performing a few modifications, one can apply this algorithm to many other discrete optimization problems, such as facility location problems [6].

Many data mining applications involve the task of building a model for predictive classification. The goal of such a model is to classify examples—records or data instances—into classes or categories of the same type. Noise or unnecessary attributes may reduce the accuracy and reliability of a classification or prediction model. Unnecessary attributes also increase the costs of building and running a model—particularly on large data sets. Before performing classification, it is therefore important to select an appropriate subset of "good" attributes. Attribute selection tries to simplify a data set by reducing its dimensionality and identifying relevant underlying attributes without sacrificing predictive accuracy. As a result, it reduces redundancy in the information provided by the attributes used for prediction. For a more detailed review of the attribute selection task using genetic algorithms, see [7].

The main difference between the DPSO and other traditional PSO algorithms is that the particles in the DPSO do not represent points inside an n-dimensional Euclidean space (continuous case) or lattice (binary case) as in the standard PSO algorithms [8]. Instead, they represent a combination of selected attributes. In previous work, the DPSO was used to select attributes for a Naive Bayes (NB) classifier. The resulting NB classifier was then used to predict postsynaptic function in proteins.

The study presented here extends previous work reported in [5, 9] in two ways. First, it enables the DPSO to select attributes for a Bayesian network algorithm, which is more sophisticated than the Naive Bayes algorithm previously used. Second, it increases the number of data sets used to evaluate the PSO from 1 to 6. All the 6 functional classification data sets used have a much greater number of classes to be predicted—in contrast with the postsynaptic data set used in [5] which had just two classes to be predicted.

The work is organized as follows. Section 2 briefly addresses Bayesian networks and Naive Bayes classifier. Section 3 shortly discusses PSO algorithms. Section 4 describes the standard binary PSO algorithm and Section 5 the DPSO algorithm. Section 6 describes the G-protein-coupled receptors (GPCRs) and Enzyme data sets used in the computational experiments. Section 7 reports computational experiments—it also includes a discussion of the results obtained. Section 8 presents conclusions and points out future research directions.

2. BAYESIAN NETWORKS AND NAIVE BAYES

The Naive Bayes (NB) classifier uses a probabilistic approach to assign each record of the data set to a possible class. In this work, the NB classifier assigns a protein of a data set of proteins to a possible class. A Naive Bayes classifier assumes that all attributes are conditionally independent of one another given the class [10].

A Bayesian network (BN), by contrast, detects probabilistic relationships among these attributes and uses this information to aid the attribute selection process.

Bayesian networks are graphical representations of a probability distribution over a set of variables of a given problem domain [11, 12]. This graphical representation is a directed acyclic graph in which nodes represent the variables of the problem and arcs represent conditional probabilistic independencies among the nodes. A directed acyclic graph G is an ordered pair $G = (V, E)$, where V is a set whose elements are called vertices or nodes and E is a set whose elements are called directed edges, arcs, or arrows. The graph G contains no directed cycles—for any vertex $v \in V$, there is no directed path that starts and ends on v.

An example of a Bayesian network is as follows. (This is a modified version of the so-called "Asia" problem [13].) Suppose that a doctor is treating a patient who has been suffering from shortness of breath—called dyspnoea. The doctor knows that diseases such as tuberculosis, bronchitis, and lung cancer are possible causes for that. The doctor also knows that other relevant information includes whether the patient is a smoker—increasing the chances of lung cancer and bronchitis—and what sort of air pollution the patient has been exposed to. A positive x-ray would indicate either tuberculosis or lung cancer. The set of variables for this problem and their possible values are shown in Table 1.

Table 1: Bayesian network: nodes and values for the lung cancer problem. L = low, H = high, T = true, F = false, Pos = positive, and Neg = negative.

Node name	Values
Pollution	{L, H}
Smoker	{T, F}
Cancer	{T, F}
Dyspnoea	{T, F}
X-ray	{Pos, Neg}

Figure 1 shows a Bayesian network representing this problem. For applications of Bayesian networks on evolutionary algorithms and optimization problems, see [14, 15].

More formally, let $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ be a multivariate random variable whose components X_i are also random variables. A corresponding lower-case letter x_i denotes an assignment of state or value to the random variable X_i. Parents (X_i) represent the set of nodes—variables or attributes in this work—that have a directed edge pointing to X_i. Let us consider a BN containing n nodes, X_1 to X_n, taken in that order. A particular value of $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ in the joint probability distribution is represented by

$$p(\mathbf{X}) = p(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n), \tag{1}$$

or more compactly, $p(x_1, x_2, \dots, x_n)$. The chain rule of probability theory allows the factorization of joint probabilities, therefore

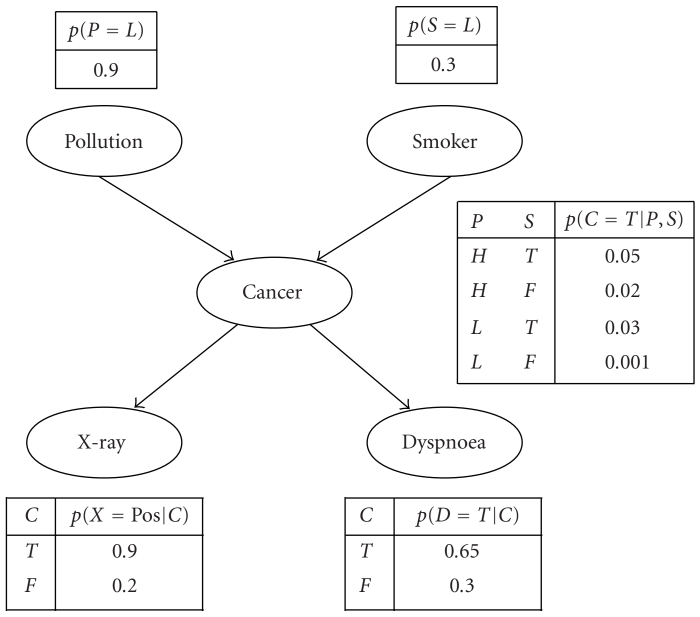
$$\begin{aligned} p(\mathbf{X}) &= p(x_1)\,p(x_2 \mid x_1)\cdots p(x_n \mid x_1, \dots, x_{n-1}) \\ &= \prod_i p(x_i \mid x_1, \dots, x_{i-1}). \end{aligned} \tag{2}$$

As the structure of a BN implies that the value of a particular node is conditional only on the values of its parent nodes, (2) may be reduced to

$$p(\mathbf{X}) = \prod_i p(X_i \mid \text{Parents}\,(X_i)). \tag{3}$$

Learning the structure of a BN is an NP-hard problem [16, 17]. Many algorithms that were developed to this end use a scoring metric and a search procedure. The scoring metric evaluates the goodness-of-fit of a structure to the data. The search procedure generates alternative structures and selects the best one based on the scoring metric. To reduce the search space of networks, only candidate networks in which each node has at most k-inward arcs (parents) are considered —k is a parameter determined by the user. In the present work, k is set to 20 ($k = 20$) to avoid overly complex models.

A greedy search algorithm is used to generate alternative structures for the BN starting with an empty network, the greedy search algorithm adds into the network the edge that most increases the score of the resulting network. The search stops when no other edge addition improves the score of the network. Algorithm 1 shows the pseudocode of this generic greedy search algorithm.



P	S	$p(C = T \mid P, S)$
H	T	0.05
H	F	0.02
L	T	0.03
L	F	0.001

C	$p(X = \text{Pos} \mid C)$
T	0.9
F	0.2

C	$p(D = T \mid C)$
T	0.65
F	0.3

Figure 1: A Bayesian network representing the lung cancer problem.

```
Require: Initialize an empty Bayesian network $G$ containing $n$ nodes (i.e., a BN with $n$ nodes but no edges)
1: Evaluate the score of $G$: $Score(G)$
2: $BEST = Score(G)$
3: repeat
4:   $FROM = 0$
5:   $TO = 0$
6:   for $i = 1$ to $n$ do
7:     for $j = 1$ to $n$ do
8:       $G' = G$
9:       if $i \neq j$ then
10:        if there is no edge between the nodes $i$ and $j$ in $G'$ then
11:          Modify $G'$: add an edge between the nodes $i$ and $j$ in $G'$ such that $i$ is a parent of $j$: $(i \rightarrow j)$
12:          if the resulting $G'$ is a DAG then
13:            if $(Score(G') > BEST)$ then
14:              $BEST = Score(G')$
15:              $FROM = i$
16:              $TO = j$
17:            end if
18:          end if
19:        end if
20:      end if
21:    end for
22:  end for
23:  if $FROM > 0$ then
24:    Modify $G$: add an edge between the nodes $FROM$ and $TO$ in $G$ such that $FROM$ is a parent of $TO$: (FROM $\rightarrow$ TO)
25:  end if
26: until $FROM = 0$
27: return $G$ as the structure of the BN
```

Algorithm 1: Pseudocode for a generic greedy search algorithm.

To evaluate the "goodness-of-fit" (score) of a network structure to the data, an unconventional scoring metric—specific for the target classification task—is adopted. The entire data set is divided into mutually exclusive training and test sets—the standard methodology for evaluating classifiers, see Section 7.1. The training set is further divided into two mutually exclusive parts. The first part is used to compute the probabilities for the Bayesian network. The second part is used as the validation set. During the search for the best possible network structure, only the validation set is used to compute predictive accuracy. The score of a candidate network is given by the classification accuracy in the validation set. The graphical model of the network that shows the highest predictive accuracy on the validation set—during the entire PSO run—is then used to compute the predictive accuracy on the test set.

Once the best network structure is selected, at the end of the PSO run, the validation set and the other part of the training set are merged and this merged data—that is, the entire original training set—is used to compute the probabilities for the selected Bayesian network. The predicted accuracy—reported as the final result—is then computed on the previously untouched test set. This process is discussed again, in more details, in Section 7.1. A similar process is adopted for the computation of the predictive accuracy using the Naive Bayes classifier.

3. A BRIEF INTRODUCTION TO PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) comprises a set of search techniques, inspired by the behavior of natural swarms, for solving optimization problems [8]. In PSO, a potential solution to a problem is represented by a particle, $\mathbf{Y}(i) = (Y_{(i,1)}, Y_{(i,2)}, \ldots, Y_{(i,n)})$ in an n-dimensional search space. $\mathbf{Y}(i)$ represents the ith particle in the population and n represents the number of variables of the problem. The coordinates $Y_{(i,d)}$ of these particles have a rate of change (velocity) $V_{(i,d)}$, $d = 1, 2, \ldots, n$. Note that the use of the double subscript notation (i, d) like in $Y_{(i,d)}$ represents the dth component of the ith particle in the swarm $\mathbf{Y}(i)$—the same rationale is used for $V_{(i,d)}$, and so forth.

Every particle keeps a record of the best position that it has ever visited. Such a record is called the particle's previous best position and denoted by $\mathbf{B}(i)$. The global best position attained by any particle so far is also recorded and stored in a particle denoted by $\mathbf{G}$. An iteration comprises evaluation of each particle, then stochastic adjustment of $V_{(i,d)}$ in the direction of particle $\mathbf{Y}(i)$'s previous best position and the previous best position of any particle in the neighborhood [18]. There is much variety in the neighborhood topology used in PSO, but quite often *gbest* or *lbest* topologies are used. In the *gbest* topology, the neighborhood of a particle consists of all the other particles in the swarm, and therefore all the particles will have the same global best neighbor—which is the best particle in the entire population. In the *lbest* topology, each particle has just a "local" set of neighbors, typically much fewer than the number of particles in the swarm, and so different particles can have different best local neighbors. For a review of the neighborhood topologies used in PSO the reader is referred to [8, 19].

As a whole, the set of rules that govern PSO are evaluate, compare, and imitate. The evaluation phase measures how well each particle (candidate solution) solves the problem at hand. The comparison phase identifies the best particles. The imitation phase produces new particle positions based on some of the best particles previously found. These three phases are repeated until a given stopping criterion is met. The objective is to find the particle that best solves the target problem.

Important concepts in PSO are velocity and neighborhood topology. Each particle, $\mathbf{Y}(i)$, is associated with a velocity vector. This velocity vector is updated at every generation. The updated velocity vector is then used to generate a new particle position $\mathbf{Y}(i)$. The neighborhood topology defines how other particles in the swarm, such as $\mathbf{B}(i)$ and $\mathbf{G}$, interact with $\mathbf{Y}(i)$ to modify its respective velocity vector and, consequently, its position as well.

4. THE STANDARD BINARY PSO ALGORITHM

Potential solutions to the target problem are encoded as fixed size binary strings; that is, $\mathbf{Y}(i) = (Y_{(i,1)}, Y_{(i,2)}, \ldots, Y_{(i,n)})$, where $Y_{(i,j)} \in \{0, 1\}$, $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, n$ [8]. Given a list of attributes $A = (A_1, A_2, \ldots, A_n)$, the first element of $\mathbf{Y}(i)$, from the left to the right hand side, corresponds to the first attribute "A_1," the second to the second attribute "A_2," and so forth. A value of 0 on the site associated to an attribute indicates that the respective attribute is not selected. A value of 1 indicates that it is selected.

4.1. *The initial population for the standard binary PSO algorithm*

For the initial population, N binary strings of size n are randomly generated. Each particle $\mathbf{Y}(i)$ is generated independently. For every position $Y_{(i,d)}$ in $\mathbf{Y}(i)$, a uniform random number φ is drawn on the interval (0, 1). If $\varphi < 0.5$, then $Y_{(i,d)} = 1$, otherwise $Y_{(i,d)} = 0$.

4.2. *Updating the records for the standard binary PSO algorithm*

At the beginning, the previous best position of $\mathbf{Y}(i)$, denoted by $\mathbf{B}(i)$, is empty. Therefore, once the initial particle $\mathbf{Y}(i)$ is generated, $\mathbf{B}(i)$ is set to $\mathbf{B}(i) = \mathbf{Y}(i)$. After that, every time that $\mathbf{Y}(i)$ is updated, $\mathbf{B}(i)$ is also updated if $f(\mathbf{Y}(i))$ is better than $f(\mathbf{B}(i))$. Otherwise, $\mathbf{B}(i)$ remains as it is. Note that $f(\cdot)$ represents the fitness function used to measure the quality of the candidate solutions. A similar process is used to update the global best position $\mathbf{G}$. Once that all the $\mathbf{B}(i)$ have been determined, $\mathbf{G}$ is set to the fittest $\mathbf{B}(i)$ previously computed. After that, $\mathbf{G}$ is updated if the fittest $\mathbf{B}(i)$ in the swarm is better than $\mathbf{G}$. And, in that case, $f(\mathbf{G})$ is set to $f(\mathbf{G}) = f(\text{fittest } \mathbf{B}(i))$. Otherwise, $\mathbf{G}$ remains as it is.

4.3. Updating the velocities for the standard binary PSO algorithm

Every particle $\mathbf{Y}(i)$ is associated to a unique vector of velocities $\mathbf{V}(i) = (V_{(i,1)}, V_{(i,2)}, \dots, V_{(i,n)})$. Note that, for simplicity, this work uses row vectors rather than column vectors. The elements $V_{(i,d)}$ in $\mathbf{V}(i)$ determine the rate of change of each respective coordinate $Y_{(i,d)}$ in $\mathbf{Y}(i)$, $d = 1, 2, \dots, n$. Each element $V_{(i,d)} \in \mathbf{V}(i)$ is updated according to the equation

$$V_{(i,d)} = wV_{(i,d)} + \varphi_1 (B_{(i,d)} - Y_{(i,d)}) + \varphi_2 (G_{(d)} - Y_{(i,d)}), \tag{4}$$

where w $(0 < w < 1)$, called the inertia weight, is a constant value chosen by the user and $d = 1, 2, \dots, n$. Equation (4) is a standard equation used in PSO algorithms to update the velocities [20, 21]. The factors φ_1 and φ_2 are uniform random numbers independently generated in the interval (0, 1).

4.4. Sampling new particle positions for the standard binary PSO algorithm

For each particle $\mathbf{Y}(i)$ and each dimension d, the value of the new coordinate $Y_{(i,d)} \in \mathbf{Y}(i)$ can be either 0 or 1. The decision of whether $Y_{(i,d)}$ will be 0 or 1 is based on its respective velocity $V_{(i,d)} \in \mathbf{V}(i)$ and is given by the equation

$$Y_{(i,d)} = \begin{cases} 1, & \text{if } (\text{rand} < S(V_{(i,d)})), \\ 0, & \text{otherwise}; \end{cases} \tag{5}$$

where $0 \le \text{rand} \le 1$ is a uniform random number, and

$$S(V_{(i,d)}) = \frac{1}{1 + \exp(-V_{(i,d)})} \tag{6}$$

is the sigmoid function. Equation (5) is a standard equation used to sample new particle positions in the binary PSO algorithm [8]. Note that the lower the value of $V_{(i,d)}$ is, the more likely the value of $Y_{(i,d)}$ will be 0. By contrast, the higher the value of $V_{(i,d)}$ is, the more likely the value of $Y_{(i,d)}$ will be 1. The motivation to use the sigmoid function is to map the interval $[-V_{(i,d)}, V_{(i,d)}]$ for all i, d into the interval (0, 1) which is equivalent to the interval of a probability function.

5. THE DISCRETE PSO (DPSO) ALGORITHM

The DPSO algorithm deals directly with discrete variables (attributes) and, unlike the binary PSO algorithm, its population of candidate solutions contains particles of different sizes. Potential solutions to the optimization problem at hand are represented by a swarm of particles. There are N particles in a swarm. The size of each particle may vary from 1 to n, where n is the number of variables—attributes in this work—of the problem. In this context, the size of a particle refers to the number of different attribute indices that the particle is able to represent at a single time.

For example, given $i, j \in \{1, 2, \dots, N\}$, in DPSO it may occur that a particle $\mathbf{Z}(i)$ in the population has size 6 ($\mathbf{Z}(i) = \{*, *, *, *, *, *\}$), whereas another particle $\mathbf{Z}(j)$ in the same population has size 2 ($\mathbf{Z}(i) = \{*, *\}$), and so forth, or any other sizes between 1 and n.

Each particle $\mathbf{Z}(i)$ keeps a record of the best position it has ever attained. This information is stored in a separate vector labeled as $\mathbf{B}(i)$. The swarm also keeps a record of the global best position ever attained by any particle in the swarm. This information is also stored in a separate vector labeled $\mathbf{G}$. Note that $\mathbf{G}$ is equal to the best $\mathbf{B}(i)$ present in the swarm.

5.1. Encoding of the particles for the DPSO algorithm

Each attribute is represented by a unique positive integer number, or index. These numbers, indices, vary from 1 to n. A particle is a subset of nonordered indices without repetition, for example, $\mathbf{Z}(k) = \{2, 4, 18, 1\}$, $k \in \{1, 2, \dots, N\}$.

5.2. The initial population for the DPSO algorithm

The original work on DPSO [5] used a randomly generated initial population for the standard PSO algorithm and a new randomly generated initial population for the DPSO algorithm, when comparing these algorithms' performances in a given data set. However, the way in which those populations were initialized generated a doubt about a possible advantage of one initial population over the other—which would bias the performance of one algorithm over the other. In this work, to eliminate this possible bias, the initial population used by the DPSO is always identical to the initial population used by the binary PSO. They differ only in the way in which solutions are represented. The conversion of every particle in the initial population of solutions of the binary PSO to the Discrete PSO initial population is as follows.

The index of every attribute that has value 1 is copied to the new solution (particle) of the DPSO initial population. For instance, an initial candidate solution for the binary PSO algorithm equal to $\mathbf{Y}(k) = (1, 0, 1, 1, 0)$ is converted into $\mathbf{Z}(k) = \{1, 3, 4\}$ for the DPSO algorithm—because attributes A_1, A_3, and A_4 are set to 1 (are present) in $\mathbf{Y}(k)$, $k \in \{1, 2, \dots, N\}$. Note that the same initial population of solutions is used to both algorithms, binary PSO and DPSO, to make the comparison between the performances of these algorithms as free from initialization bias as possible.

Initializing the particles $\mathbf{Z}(i)$ in this way causes different particles, in DPSO, to have different sizes. For instance, an initial candidate solution $\mathbf{Y}(j) = (1, 1, 0, 0, 0)$ (from the binary PSO algorithm) is converted into the initial candidate solution $\mathbf{Z}(j) = \{1, 2\}$ (to the DPSO algorithm) which has size 2, whereas another initial candidate solution $\mathbf{Y}(k) = (0, 1, 1, 1, 1)$ (binary PSO) is converted into the initial candidate solution $\mathbf{Z}(k) = \{2, 3, 4, 5\}$ (DPSO) which has size 4, $j, k \in \{1, 2, \dots, N\}$ and $n = 5$.

In the DPSO algorithm, for simplicity, once the size of a particle is determined at the initialization, the particle will keep that same size during the entire execution of the algorithm. For example, particle $\mathbf{Z}(k) = \{2, 3, 4, 5\}$ above, which has been initialized with 4 indices, will always carry exactly 4 indices, $\mathbf{Z}(k) = \{*, *, *, *\}$. The values of those 4 indices, however, are likely to change every time that the particle is updated.

5.3. Velocities = proportional likelihoods

The DPSO algorithm does not use a vector of velocities as the standard PSO algorithm does. It works with proportional likelihoods instead. Arguably, the notion of proportional likelihood used in the DPSO algorithm and the notion of velocity used in the standard PSO are somewhat similar. DPSO uses $\mathbf{M}(i)$ to represent an array of proportional likelihoods and $M(i,d)$ to represent one of $\mathbf{M}(i)$'s components.

Every particle in DPSO is associated with a 2-by-n array of proportional likelihoods, where 2 is the number of rows in this array and n is the number of columns—note that the number of columns in $\mathbf{M}(i)$ is equal to the number of variables of the problem n.

This is an example of a generic proportional likelihood array

$$\mathbf{M}(i) = \begin{pmatrix} \text{proportional-likelihood-row} \\ \text{attribute-index-row} \end{pmatrix}. \qquad (7)$$

Each of the n elements in the first row of $\mathbf{M}(i)$ represents the proportional likelihood that an attribute be selected. The second row of $\mathbf{M}(i)$ shows the indices of the attributes associated with the respective proportional likelihoods.

There is a one-to-one correspondence between the columns of this array and the attributes of the problem domain. At the beginning, all elements in the first row of $\mathbf{M}(i)$ are set to 1, for example,

$$\mathbf{M}(i) = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \qquad (8)$$

After the initial population of particles is generated, this array is always updated before a new configuration for the particle associated to it is generated. The updating of the likelihoods $M_{(i,d)}$ is based on $\mathbf{Z}(i)$, $\mathbf{B}(i)$, $\mathbf{G}$ and three constant updating factors, namely, α, β, and γ. The updating factors (α, β, and γ) determine the strength of the contribution of $\mathbf{Z}(i)$, $\mathbf{B}(i)$, and $\mathbf{G}$ to the adjustment of every coordinate $M_{(i,d)} \in \mathbf{M}(i)$.

Note that α, β, and γ are parameters chosen by the user. The contribution of these parameters to the updating of $M_{(i,d)}$ is as follows. All indices present in $\mathbf{Z}(i)$ have their correspondent proportional likelihood increased by α. In addition to that, all indices present in $\mathbf{B}(i)$ have their correspondent proportional likelihood increased by β. The same for $\mathbf{G}$ for which the proportional likelihoods are increased by γ.

For instance, given $n = 5$, $\alpha = 0.10$, $\beta = 0.12$, $\gamma = 0.14$, $\mathbf{Z}(i) = \{2,3,4\}$, $\mathbf{B}(i) = \{3,5,2\}$, $\mathbf{G} = \{5,2\}$, and also

$$\mathbf{M}(i) = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \qquad (9)$$

the updated $\mathbf{M}(i)$ would be

$$\mathbf{M}(i) = \begin{pmatrix} (1.0) & (1.0+\alpha+\beta+\gamma) & (1.0+\alpha+\beta) & (1.0+\alpha) & (1.0+\beta+\gamma) \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \qquad (10)$$

Note that index 1 is absent in $\mathbf{Z}(i)$, $\mathbf{B}(i)$, and $\mathbf{G}$. Therefore, the proportional likelihood of attribute 1 in $\mathbf{M}(i)$ remains as it is. In this work, the values used for α, β, and γ were $\alpha = 0.10$, $\beta = 0.12$, and $\gamma = 0.14$. These values were empirically determined in preliminary experiments; but this work makes no claim that these are optimal values. Parameter optimization is a topic for future research. As a whole, these values make the contribution of $\mathbf{B}(i)$ ($\beta = 0.12$) to the updating of the $\mathbf{V}(i)$ a bit stronger than the contribution of $\mathbf{Z}(i)$ ($\alpha = 0.10$); and the contribution of $\mathbf{G}$ ($\gamma = 0.14$) even stronger.

The new updated array $\mathbf{M}(i)$ replaces the old one and will be used to generate a new configuration to the particle associated to it as follows.

5.4. Sampling new particle positions for the DPSO algorithm

The proportional likelihood array $\mathbf{M}(i)$ is then used to sample a new instance of particle $\mathbf{Z}(i)$—the particle associated to $\mathbf{M}(i)$. For this sampling process, a series of operations is performed on the array. To start with, every element of the first row of the array $\mathbf{M}(i)$ is multiplied by a uniform random number between 0 and 1. A new random number is drawn for every single multiplication performed.

To illustrate, suppose that

$$\mathbf{M}(i) = \begin{pmatrix} 1.00 & 1.36 & 1.22 & 1.10 & 1.26 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \qquad (11)$$

The multiplied proportional likelihood array would be

$$\mathbf{M}(i) = \begin{pmatrix} (1.00 \cdot \varphi_1) & (1.36 \cdot \varphi_2) & (1.22 \cdot \varphi_3) & (1.10 \cdot \varphi_4) & (1.26 \cdot \varphi_5) \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \qquad (12)$$

where $\varphi_1, \dots, \varphi_5$ are uniform random numbers independently drawn on the interval (0, 1).

Suppose that this is the resulting array $\mathbf{M}(i)$ after the multiplication

$$\mathbf{M}(i) = \begin{pmatrix} 0.11 & 0.86 & 0.57 & 0.62 & 1.09 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \qquad (13)$$

A new particle position is then defined by ranking the columns in $\mathbf{M}(i)$ by the values in its first row. That is, the elements in the first row of the array are ranked in a decreasing order of value; and the indices of the attributes—in the second row of $\mathbf{M}(i)$—follow their respective proportional likelihoods. For example, ranking the array $\mathbf{M}(i)$ (shown immediately above) would generate

$$\mathbf{M}(i) = \begin{pmatrix} 1.09 & 0.86 & 0.62 & 0.57 & 0.11 \\ 5 & 2 & 4 & 3 & 1 \end{pmatrix}. \qquad (14)$$

The next operation now is to select the indices that will compose the new particle position. After ranking the array $\mathbf{M}(i)$, the first s_i indices (in the second row of $\mathbf{M}(i)$), from left to right, are selected to compose the new particle position. Note that s_i represents the size of the particle $\mathbf{Z}(i)$—the particle associated to the ranked array $\mathbf{M}(i)$.

Suppose that the particle $\mathbf{Z}(i)$ associated to

$$\mathbf{M}(i) = \begin{pmatrix} 1.09 & 0.86 & 0.62 & 0.57 & 0.11 \\ 5 & 2 & 4 & 3 & 1 \end{pmatrix} \tag{15}$$

has size 3 ($\mathbf{Z}(i) = \{*,*,*\}$). That makes $s_i = 3$—note that $\mathbf{Z}(j)$, for instance, may have a different size and consequently a different s_j value. For the $\mathbf{Z}(i)$ above, however, as $s_i = 3$ the first 3 indices from the second row of $\mathbf{M}(i)$ would be selected to compose the new particle position. Based on the array $\mathbf{M}(i)$ given above and $s_i = 3$, the indices (attributes) 5, 2, and 4 would be selected to compose the new particle position, that is, $\mathbf{Z}(i) = \{5, 2, 4\}$. Note that indices that have a higher proportional likelihood are, on average, more likely to be selected.

The updating of $\mathbf{Z}(i)$, $\mathbf{B}(i)$, and $\mathbf{G}$ follows what is described in Section 4.2.

Once the algorithms have been explained, the next section briefly introduces the particular data sets (case studies) used to test the algorithms.

6. CASE STUDY: THE GPCR AND ENZYME DATA SETS USED IN THE COMPUTATIONAL EXPERIMENTS

The experiments involved 6 data sets comprising two kinds of proteins, namely, G-protein-coupled receptors (GPCRs) and Enzymes.

The G-protein-coupled receptors (GPCRs) are a protein superfamily of transmembrane receptors. Their function is to transduce signals that induce a cellular response to the environment. GPCRs are involved in many types of stimulus-response pathways, from intercellular communication to physiological senses. GPCRs are of much interest to the pharmaceutical industry because these proteins are involved in many pathological conditions—it is estimated that GPCRs are the target of 40% to 50% of modern medical drugs [22]

Enzymes are proteins that accelerate chemical reactions—they participate in many processes in a biological cell. Some enzymes are used in the chemical industry and other industrial applications where extremely specific catalysts are required. In Enzyme Nomenclature, enzymes are assigned and identified by an Enzyme Commission (EC) number. For instance, EC 2.3.4 is an enzyme with class value 2 in the first hierarchical class level, class value 3 in the second class level, and so forth. This work uses the GPCRs and EC data sets described in Table 2.

These data sets were derived from the data sets used in [23, 24]. Note that both the GPCR and the Enzyme data sets have hierarchical classes. Each protein in these data sets is assigned one class at the first (top) hierarchical level, corresponding to a broad function, another class at the second level, corresponding to a more specialized function, and another class at the third level, corresponding to an even more specialized function, and so forth. This work copes with these hierarchical classes in a simple way by predicting classes one level at a time, as explained in more detail later.

The data sets used in the experiments involved four kinds of protein signatures (biological "motifs"), namely, PROSITE patterns, PRINTS fingerprints, InterPro entries, and Pfam signatures.

Table 2: GPCR and EC data sets. "Cases" represents the number of proteins in the data set, "Attributes" represents the total number of attributes that describe the proteins in the data set, and "L1", . . . , "L4" represent the number of classes at hierarchical class levels 1, . . . , 4, respectively.

			# Classes at			
Data set	Cases	Attributes	L1	L2	L3	L4
GPCR-PRINTS	330	281	8	36	52	44
GPCR-PROSITE	190	127	8	32	32	—
GPCR-InterPro	580	448	12	43	67	46
EC-PRINTS	500	380	6	43	83	—
EC-PROSITE	570	583	6	42	84	—
EC-Pfam	730	706	6	41	92	—

Table 3: Predictive accuracy: binary PSO versus DPSO. Paired two-tailed t test for the predictive accuracy—significance level 0.05.

Class level	Naive Bayes	Bayesian network
1	$t(9) = 0.467$, $p = 0.651$	$t(9) = 3.407$, $p = 0.007$
2	$t(9) = 2.221$, $p = 0.053$	$t(9) = 3.200$, $p = 0.010$
3	$t(9) = 3.307$, $p = 0.009$	$t(9) = 3.556$, $p = 0.006$

PROSITE is a database of protein families and domains. It is based on the observation that, while there is a huge number of different proteins, most of them can be grouped, on the basis of similarities in their sequences, into a limited number of families (a protein consists of a sequence of amino acids). PROSITE patterns are essentially regular expressions describing small regions of a protein sequence which present a high sequence similarity when compared to other proteins in the same functional family.

In the data sets, the absence of a given PROSITE pattern is indicated by a value of 0 for the attribute corresponding to that PROSITE pattern. The presence of it is indicated by a value of 1 for that same attribute.

PRINTS is a compendium of protein fingerprints. A fingerprint is a group of conserved motifs used to characterize a protein family. In the PRINTS data sets, a fingerprint corresponds to an attribute. The presence of a fingerprint is indicated by a value of 1 for that same attribute; the absence by a 0.

Pfam signatures are produced by hidden Markov models, and InterPro integrates a number of protein signature databases into a single database. In this work, Pfam and InterPro entries also correspond to binary attributes indicating whether or not a protein matches those entries, using the same codification described for the PROSITE patterns and Fingerprints.

The objective of the binary PSO and DPSO algorithms is to classify each protein into its most suitable functional class level. The classification of the proteins is performed in each class level individually. For instance, given protein Υ, at first, a conventional "flat" classification algorithm assigns a class to Υ at the first class level only. Once Υ has been classified at the first class level, the conventional flat classification algorithm

Table 4: Number of selected attributes: binary PSO versus DPSO. Paired two-tailed t test for the number of attributes selected—significance level 0.05.

Class level	Naive Bayes	Bayesian network
1	$t(9) = 7.248$, $p = 4.8\text{E-}5$	$t(9) = 8.2770$, $p = 1.6\text{E-}5$
2	$t(9) = 9.052$, $p = 8.1\text{E-}6$	$t(9) = 14.890$, $p = 1.2\text{E-}7$
3	$t(9) = 6.887$, $p = 7.1\text{E-}5$	$t(9) = 9.1730$, $p = 7.3\text{E-}6$

is again applied to assign a class to Υ at the second level—no information about Υ's class at the previous level is used. The same process is used to assign a class to protein Υ at the third class level, and so forth.

7. EXPERIMENTS

The quality of a candidate solution (fitness) is evaluated in three different ways: (1) by a baseline algorithm—using all possible attributes; (2) by the binary PSO—using only the attributes selected by this algorithm; and (3) by the discrete PSO (DPSO) algorithm—using only the attributes selected by this algorithm. Each of these algorithms computes the fitness of every given solution using two distinct techniques: (a) using a Naive Bayes classifier; and (b) using a Bayesian network.

7.1. Experimental methodology

Note that the computation of the fitness function $f(\cdot)$ for the particles $\mathbf{Y}(i)$ (binary PSO algorithm) and $\mathbf{Z}(i)$ (DPSO algorithm) follows the description given below. For simplicity, only the process using $\mathbf{Y}(i)$ is described—but the same is applicable to $\mathbf{Z}(i)$. $f(\mathbf{Y}(i))$ is equal to the predictive accuracy achieved by the Naive Bayes classifier—and the Bayesian network—on each data set and using only the attributes selected in $\mathbf{Y}(i)$.

The measurement of $f(\mathbf{Y}(i))$ follows a wrapper approach. The wrapper approach searches for an optimal attribute subset tailored to a particular algorithm, such as the Naive Bayes classifier or Bayesian network. For more information on wrapper and other attribute selection approaches, see [25].

The computational experiments involved a 10-fold crossvalidation method [25]. First, the data set being considered is divided into 10 equally sized folds. The folds are randomly generated but under the following criterion. The proportion of classes in every single fold must be similar to the proportion of classes found in the original data set containing all records. This is known as stratified crossvalidation.

Each of the 10 folds is used once as a test set and the remaining of the data is used as training set. Out of the 9 folds in the training set one is reserved to be used as a validation set. The Naive Bayes classifier and the Bayesian network use the remaining 8 folds to compute the probabilities required to classify new examples. Once those probabilities have been computed, the Naive Bayes (NB) classifier and the Bayesian network (BN) classify the examples in the validation set.

The accuracy of this classification on the validation set is the value of the fitness functions $f_{\mathrm{NB}}(\mathbf{Y}(i))$ and $f_{\mathrm{BN}}(\mathbf{Y}(i))$—the same for $f_{\mathrm{NB}}(\mathbf{Z}(i))$ and $f_{\mathrm{BN}}(\mathbf{Z}(i))$. When the run of the PSO algorithm is completed, the 9 folds are merged into a full training set. The Naive Bayes classifier and the Bayesian network are then trained again on this full-training set (9 merged folds), and the probabilities computed in this final, full-training set are used to classify examples in the test set (the 10th fold), which was never accessed during the run of the algorithms.

The reasons for having separate validation and test sets are as follows. In the classification task of data mining, by definition, the goal is to measure predictive accuracy—generalization ability—on a test set unseen during training. Hence, the test set cannot be accessed by the PSO, and is reserved just to compute the predictive accuracy associated with the Bayesian classifier constructed with the best set of attributes selected at the end of the PSO run.

Concerning the validation set, which is used to compute the fitness of particles during the PSO run, this is a part of the original training set which is different from the part of the training set used to build the Bayesian classifier, and the reason for having these two separate parts of the training set is to avoid overfitting of the classifier to the training data; for overfitting in the context of classification, see [7, pages 17, 18]. In other words, if the same training set that was used to build a Bayesian classifier was also used to measure the fitness (accuracy) of the corresponding particle, there would be no pressure to build classifiers with a good generalization ability on data unseen during training, and a classifier could obtain a high accuracy by simply being overfitted to idiosyncrasies of the training set which are unlikely to generalize well to unseen data. By measuring fitness on a validation set separated from the data used to build the classifier, this is avoided, and a pressure to build classifiers with good generalization ability is introduced in the fitness function.

In each of the 10 iterations of the crossvalidation procedure, the predictive accuracy of the classification is assessed by 3 different methods, as follows.

(1) *Using all possible original attributes:* all possible attributes are used by the Naive Bayes classifier and the Bayesian network—there is no attribute selection.
(2) *Standard binary PSO algorithm:* only the attributes selected by the best particle found by the binary PSO algorithm are used by the Naive Bayes classifier and the Bayesian network.
(3) *DPSO algorithm:* only the attributes selected by the best particle found by the DPSO algorithm are used by the Naive Bayes classifier and the Bayesian network.

Since the Naive Bayes and Bayesian network classifiers used in this work are deterministic, only one run—for each of these algorithms—is performed for the classification using all possible attributes.

For the binary PSO and the DPSO algorithms, 30 independent runs are performed for each iteration of the crossvalidation procedure. The results reported are averaged over these 30 independent runs and over the 10 iterations of the crossvalidation procedure.

The population size used for both algorithms (binary PSO and DPSO) is 200 and the search stops after 20 000 fitness evaluations—or 100 iterations.

The binary PSO algorithm uses an inertia weight value of 0.8 (i.e., $w = 0.8$). The choice of the value of this parameter was based on the work presented in [26].

Other choices of parameter values for the DPSO were $\alpha = 0.10$, $\beta = 0.12$, and $\gamma = 0.14$, chosen based on empirical experiments but probably not optimal values.

The measurement of the predictive accuracy rate of a model should be a reliable estimate of how well that model classifies the test examples—unseen during the training phase—on the target problem.

In Data Mining, typically, the equation

$$\text{standard accuracy rate} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \qquad (16)$$

is used to assess the accuracy rate of a classifier—where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively [25].

However, if the class distribution is highly unbalanced, (16) is an ineffective way of measuring the accuracy rate of a model. For instance, in many problems, it is easy to achieve a high value for (16) by simply predicting always the majority class. Therefore, on the experiments reported on this work, a more demanding measurement for the accuracy rate of a classification model is used.

This measurement has been used before in [27]. It is given by the equation

$$\text{predictive accuracy rate} = \text{TPR} \cdot \text{TNR}, \qquad (17)$$

where, $\text{TPR} = \text{TP}/(\text{TP}+\text{FN})$ and $\text{TNR} = \text{TN}/(\text{TN}+\text{FP})$—TPR stands for true positive rate and TNR stands for true negative rate.

Note that if any of the quantities TPR or TNR is zero, the value returned by (17) is also zero.

7.2. Discussion

Computational results are reported in Tables 5 and 6. Let us focus the discussion on the results obtained by the 3 algorithms (binary PSO, DPSO, and baseline algorithm) for attribute selection on the GPCR-PROSITE data set, see Table 5. The results obtained for the other 5 data sets are similar. To start with, the results obtained using the Naive Bayes classifier are presented.

Results obtained using the Naive Bayes classifier approach

To assess the performance of the algorithms, two criteria were considered: (1) maximizing predictive accuracy; and (2) finding the smallest subset of attributes.

The results for the first criterion, accuracy, show that both versions of the PSO algorithm did better—in all class levels—than the baseline algorithm using all attributes.

Furthermore, the DPSO algorithm did slightly better than the binary PSO algorithm also in all class levels. Nevertheless, the difference in the predictive accuracy performance between these algorithms is, in some cases, statistically insignificant.

Table 3 shows the results of a paired two-tailed t-test for the predictive accuracy of the binary PSO versus the predictive accuracy of the DPSO at a significance level of 0.05.

Table 3 shows that, using Naive Bayes as classifier, the only statistically significant difference in performance—in terms of predictive accuracy—between the algorithms binary PSO and DPSO is at the third class level. By contrast, using Bayesian networks as classifier, the difference in performance is statistically significant at all class levels.

Nevertheless, the discriminating factor between the performance of these algorithms is on the second comparison criterion—finding the smallest subset of attributes.

The DPSO not only outperformed the binary PSO in predictive accuracy, but also did so using a smaller subset of attributes in all class levels. Moreover, when it comes to effectively pruning the set of attributes, the difference in performance between the binary PSO and the DPSO is always statistically significant, as Table 4 shows.

Results obtained using the the Bayesian network approach

Again, the predictive accuracy attained by both versions of the PSO algorithm surpassed the predictive accuracy obtained by the baseline algorithm in all class levels.

DPSO obtained the best predictive accuracy of all algorithms in all class levels. Regarding the second comparison criterion, finding the smallest subset of attributes, again DPSO always selected the smallest subset of attributes in all hierarchical levels.

The results on the performance of the classifiers—Naive Bayes versus Bayesian networks—show that Bayesian networks did a much better job. For all class levels, the predictive accuracy obtained by the 3 approaches (baseline, binary PSO and DPSO) using Bayesian networks was significantly better than the predictive accuracy obtained using Naive Bayes classifier. The Bayesian networks approach also enabled the two PSO algorithms to do the job using fewer selected attributes—compared to the Naive Bayes approach.

The results emphasize the importance of taking relationships among attributes into account—as Bayesian networks do—when performing attribute selection. If these relationships are ignored, predictive accuracy is adversely affected.

The results also show that for all 6 data sets tested, the DPSO algorithm not only selected the smallest subset of attributes, but also obtained the highest predictive accuracy in every single class level.

8. CONCLUSIONS

Computational results show that the use of unnecessary attributes tends to derail classifiers and hurt classification accuracy. Using only a small subset of selected attributes, the binary PSO and DPSO algorithms obtained better predictive accuracy than the baseline algorithm using all attributes. Previous work had already shown that the DPSO algorithm outperforms the binary PSO in the task of attribute selection [5], but that work involves only one data set. This current work

Table 5: Results for the GPCRs data sets. For the binary PSO and DPSO algorithms, 30 independent runs are performed. The results reported are averaged over these 30 independent runs. The best result on each line for each performance criterion is marked with an asterisk (*).

GPCR-PRINTS (281 attributes)						
		Average predictive accuracy			Average number of selected attributes	
Method	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	72.36 ± 2.64	73.10 ± 2.71	*73.98 ± 3.13	97.40 ± 1.26	*73.30 ± 4.35
	2	35.56 ± 2.56	37.10 ± 3.10	*40.74 ± 5.36	130.30 ± 1.34	*117.30 ± 6.04
	3	27.00 ± 1.82	29.05 ± 2.71	*31.55 ± 4.54	171.10 ± 3.93	*145.70 ± 3.80
	4	24.26 ± 1.75	26.97 ± 2.24	*30.14 ± 3.78	165.00 ± 4.11	*141.30 ± 5.21
Bayesian network	1	88.67 ± 1.72	89.46 ± 1.73	*89.97 ± 2.11	89.30 ± 3.77	*63.80 ± 3.99
	2	53.46 ± 1.40	56.75 ± 2.47	*58.91 ± 3.95	123.70 ± 3.89	*103.00 ± 4.27
	3	38.93 ± 1.92	43.08 ± 3.03	*50.33 ± 6.45	158.20 ± 4.21	*134.50 ± 4.60
	4	28.47 ± 1.90	30.56 ± 2.63	*39.52 ± 5.32	152.60 ± 3.53	*126.80 ± 4.59
GPCR-PROSITE (127 attributes)						
		Average predictive accuracy			Average number of selected attributes	
Method	Class level	Using all attributes	Binary PSO	Discrete PSO	Discrete PSO	Binary PSO
Naive Bayes	1	71.27 ± 2.08	72.88 ± 2.40	*73.05 ± 2.31	85.60 ± 2.84	*74.90 ± 3.48
	2	30.00 ± 2.10	31.34 ± 2.47	*32.60 ± 2.31	101.50 ± 3.14	*83.80 ± 4.64
	3	20.47 ± 0.96	21.47 ± 1.16	*23.25 ± 1.08	102.30 ± 3.77	*87.50 ± 4.25
Bayesian network	1	78.05 ± 2.33	79.03 ± 2.57	*80.54 ± 2.46	78.50 ± 3.50	*65.50 ± 3.41
	2	39.08 ± 2.67	40.31 ± 2.85	*43.24 ± 4.67	94.10 ± 3.70	*73.30 ± 2.67
	3	24.70 ± 1.83	26.14 ± 2.11	*28.97 ± 2.77	94.90 ± 3.90	*77.60 ± 4.35
GPCR-INTERPRO (448 attributes)						
		Average predictive accuracy			Average number of selected attributes	
Method	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	54.17 ± 2.26	55.33 ± 2.36	*56.55 ± 2.61	136.40 ± 1.17	*120.70 ± 5.01
	2	25.19 ± 1.50	26.08 ± 1.62	*27.27 ± 1.87	158.60 ± 1.07	*136.20 ± 5.53
	3	20.03 ± 0.65	21.19 ± 1.03	*22.03 ± 1.63	203.60 ± 1.26	*162.40 ± 6.62
	4	27.97 ± 1.13	29.95 ± 2.18	*30.43 ± 2.56	168.00 ± 0.94	*150.10 ± 7.31
Bayesian network	1	86.68 ± 2.99	89.20 ± 3.06	*89.49 ± 4.28	122.60 ± 4.03	*107.70 ± 5.12
	2	61.85 ± 1.71	64.57 ± 1.43	*68.66 ± 3.82	146.80 ± 3.12	*128.40 ± 5.02
	3	40.77 ± 2.13	44.11 ± 2.48	*46.51 ± 3.41	184.60 ± 2.41	*148.10 ± 3.98
	4	34.05 ± 1.64	36.89 ± 2.56	*39.03 ± 3.63	149.70 ± 2.41	*131.50 ± 4.12

shows much stronger evidence for the effectiveness of DPSO in 6 data sets. In addition, the 6 data sets mined in this work are much more challenging than the two-class data set mined in [5], because the former have several hierarchical class levels per data set, leading to a much larger number of classes to be predicted for each data set.

Even when the difference in predictive accuracy is insignificant, by selecting fewer attributes than the binary PSO, the DPSO certainly enhances computational efficiency of the classifier and is therefore preferable.

The original work on DPSO [5] questioned whether the difference in performance between these two algorithms was attributable to variations in the initial population of solutions. To overcome this possible advantage/disadvantage for one algorithm or the other, the present work used the same initial population for both algorithms.

The results demonstrate that, even using an identical initial population of particles, the DPSO is still outperforming the binary PSO in both predictive accuracy and number of selected attributes. The DPSO is arguably not too different from traditional PSO but still the algorithm has features that enable it to improve over binary PSO on the task of attribute selection.

Another result—although expected—from the experiments is the clear difference in performance between Naive Bayes and Bayesian networks used as classifiers. The Bayesian networks approach outperformed the Naive Bayes approach in all experiments and in all hierarchical class levels.

In this work, the hierarchical classification problem was dealt with in a simple way by "flattening" the hierarchy, that is, by predicting classes for one class level at a time, which permitted the use of flat classification algorithms. The algorithms made no use of the information of the class assigned to a protein in one level to help predict the class at the next hierarchical level. Future work intends to look at an algorithm that makes use of this information.

Table 6: Results for the EC data sets. For the binary PSO and the DPSO algorithms, 30 independent runs are performed. The results reported are averaged over these 30 independent runs. The best result on each line for each performance criterion is marked with an asterisk (*).

EC-PRINTS (380 attributes)						
		Average predictive accuracy			Average number of selected attributes	
METHOD	Class level	Using all attributesS	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	72.35 ± 3.33	73.78 ± 3.78	*74.81 ± 3.79	102.80 ± 1.23	*64.20 ± 4.37
	2	31.19 ± 2.26	32.07 ± 2.48	*34.06 ± 2.91	149.00 ± 1.25	*112.30 ± 3.06
	3	23.37 ± 1.73	24.64 ± 2.01	*26.97 ± 2.49	211.10 ± 1.37	*150.60 ± 5.58
Bayesian network	1	88.30 ± 1.94	89.51 ± 2.51	*90.73 ± 2.59	92.80 ± 4.57	*48.90 ± 4.68
	2	53.15 ± 1.49	55.14 ± 1.87	*56.92 ± 2.93	129.70 ± 4.11	*102.00 ± 5.52
	3	36.24 ± 1.62	38.26 ± 2.65	*40.95 ± 4.16	190.40 ± 4.55	*135.10 ± 4.28
EC-PROSITE (583 attributes)						
		Average predictive accuracy			Average number of selected attributes	
METHOD	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	69.52 ± 5.02	70.37 ± 5.15	*72.31 ± 5.44	118.80 ± 1.14	*98.90 ± 1.85
	2	35.70 ± 1.73	37.73 ± 2.04	*38.83 ± 2.66	154.50 ± 0.85	*134.90 ± 4.93
	3	21.91 ± 1.13	22.86 ± 1.30	*24.36 ± 1.66	197.70 ± 1.16	*154.50 ± 5.34
Bayesian network	1	82.80 ± 1.09	84.83 ± 1.46	*85.95 ± 2.31	105.00 ± 3.62	*92.70 ± 4.00
	2	45.30 ± 2.41	47.82 ± 2.80	*49.50 ± 3.33	135.20 ± 3.65	*119.00 ± 3.89
	3	28.44 ± 2.37	29.40 ± 2.64	*32.52 ± 3.71	172.00 ± 2.94	*146.50 ± 4.40
EC-PFAM (706 attributes)						
		Average predictive accuracy			Average number of selected attributes	
METHOD	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	71.61 ± 3.52	72.87 ± 4.02	*74.62 ± 3.77	131.60 ± 5.50	*102.20 ± 3.85
	2	46.70 ± 1.21	48.24 ± 1.39	*49.02 ± 1.17	212.60 ± 5.10	*153.90 ± 5.26
	3	31.00 ± 1.08	32.20 ± 1.53	*33.24 ± 1.76	244.40 ± 4.53	*177.70 ± 2.58
Bayesian network	1	85.94 ± 1.80	87.94 ± 1.80	*89.64 ± 3.27	116.60 ± 4.22	*91.80 ± 4.52
	2	55.34 ± 1.30	56.84 ± 1.49	*58.02 ± 2.02	198.00 ± 4.40	*141.90 ± 4.63
	3	36.56 ± 1.56	37.61 ± 1.44	*39.44 ± 3.07	221.70 ± 4.64	*168.60 ± 4.43

ACKNOWLEDGMENTS

The authors would like to thank Nick Holden for kindly providing them with the biological data sets used in this work. The authors would also like to thank EPSRC (grant *Extended Particle Swarms* GR/T11265/01) for financial support.

REFERENCES

[1] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 489–500, Springer, New York, NY, USA, 2004.

[2] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," in *Proceedings of the 1st European Workshop on Evolutionary Algorithms in Stochastic and Dynamic Environments (EvoCOP '04)*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 513–524, Springer, Coimbra, Portugal, April 2004.

[3] M. Løvbjerg and T. Krink, "Extending particle swarm optimisers with self-organized criticality," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, et al., Eds., vol. 2, pp. 1588–1593, IEEE Press, Honolulu, Hawaii, USA, May 2002.

[4] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[5] E. S. Correa, A. A. Freitas, and C. G. Johnson, "A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set," in *Proceedings of the 8th Annual Conference Genetic and Evolutionary Computation (GECCO '06)*, M. Keijzer, M. Cattolico, D. Arnold, et al., Eds., pp. 35–42, ACM Press, Seattle, Wash, USA, July 2006.

[6] E. S. Correa, M. T. A. Steiner, A. A. Freitas, and C. Carnieri, "A genetic algorithm for solving a capacity p-median problem," *Numerical Algorithms*, vol. 35, no. 2–4, pp. 373–388, 2004.

[7] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer, Berlin, Germany, 2002.

[8] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[9] E. S. Correa, A. A. Freitas, and C. G. Johnson, "Particle swarm and Bayesian networks applied to attribute selection for protein functional classification," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 2651–2658, London, UK, July 2007.

[10] T. M. Mitchell, *Machine Learning*, McGraw-Hill, London, UK, 1997.

[11] F. V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, New York, NY, USA, 1st edition, 2001.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, Calif, USA, 1st edition, 1988.

[13] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistics Society*, vol. 50, no. 2, pp. 157–224, 1988.

[14] P. Larrañaga, R. Etxeberria, J. A. Lozano, B. Sierra, I. Naki Inza, and J. M. Peña, "A review of the cooperation between evolutionary computation and probabilistic models," in *Proceedings of the 2nd International Symposium on Artificial Intelligence and Adaptive Systems (CIMAF '99)*, pp. 314–324, La Havana, Cuba, March 1999.

[15] J. M. Peña, J. A. Lozano, and P. Larrañaga, "Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks," *Evolutionary Computation*, vol. 13, no. 1, pp. 43–66, 2005.

[16] R. R. Bouckaert, "Properties of Bayesian belief network learning algorithms," in *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI '94)*, I. R. L. de Mantaras and E. D. Poole, Eds., pp. 102–109, Morgan Kaufmann, Seattle, Wash, USA, July 1994.

[17] D. M. Chickering, D. Geiger, and D. Heckerman, "Learning Bayesian networks is NP-hard," Tech. Rep. MSR-TR-94-17, Microsoft Research, Redmond, Wash, USA, November 1994.

[18] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '97)*, vol. 5, pp. 4104–4109, IEEE, Orlando, Fla, USA, October 1997.

[19] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of the Congress of Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 3, pp. 1931–1938, IEEE Press, Washington, DC, USA, July 1999.

[20] G. Kendall and Y. Su, "A particle swarm optimisation approach in the construction of optimal risky portfolios," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, part of the 23rd Multi-Conference on Applied Informatics*, pp. 140–145, Innsbruck, Austria, February 2005.

[21] R. Poli, C. D. Chio, and W. B. Langdon, "Exploring extended particle swarms: a genetic programming approach," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 169–176, ACM Press, Washington, DC, USA, June 2005.

[22] D. Filmore, "It's a GPCR world," *Modern Drug Discovery*, vol. 11, no. 7, pp. 24–28, 2004.

[23] N. Holden and A. A. Freitas, "Hierarchical classification of G-protein-coupled receptors with a PSO/ACO algorithm," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '06)*, pp. 77–84, IEEE Press, Indianapolis, Ind, USA, May 2006.

[24] N. Holden and A. A. Freitas, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 100–107, IEEE Press, Pasadena, Calif, USA, June 2005.

[25] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2nd edition, 2005.

[26] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 591–600, Springer, San Diego, Calif, USA, March 1998.

[27] G. L. Pappa, A. J. Baines, and A. A. Freitas, "Predicting post-synaptic activity in proteins with data mining," *Bioinformatics*, vol. 21, supplement 2, pp. ii19–ii25, 2005.