Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2013, Article ID 756719, 7 pages
http://dx.doi.org/10.1155/2013/756719

Research Article

Parallel Swarms Oriented Particle Swarm Optimization

Tad Gonsalves and Akira Egashira

Department of Information and Communication Sciences, Faculty of Science & Technology, Sophia University, 7-1 Kioicho, Chiyoda-ku, Tokyo 102-8554, Japan

Correspondence should be addressed to Tad Gonsalves; tad-gonsal@sophia.jp

Received 29 August 2013; Accepted 9 September 2013

Academic Editor: Baoding Liu

The particle swarm optimization (PSO) is a recently invented evolutionary computation technique which is gaining popularity owing to its simplicity in implementation and rapid convergence. In the case of single-peak functions, PSO rapidly converges to the peak; however, in the case of multimodal functions, the PSO particles are known to get trapped in the local optima. In this paper, we propose a variation of the algorithm called parallel swarms oriented particle swarm optimization (PSO-PSO) which consists of a multistage and a single stage of evolution. In the multi-stage of evolution, individual subswarms evolve independently in parallel, and in the single stage of evolution, the sub-swarms exchange information to search for the global-best. The two interweaved stages of evolution demonstrate better performance on test functions, especially of higher dimensions. The attractive feature of the PSO-PSO version of the algorithm is that it does not introduce any new parameters to improve its convergence performance. The strategy maintains the simple and intuitive structure as well as the implemental and computational advantages of the basic PSO.

1. Introduction

Evolutionary algorithms (EAs) are increasingly being applied to solve the problems in diverse domains. These meta-heuristic algorithms are found to be successful in many domains chiefly because of their domain-independent evolutionary mechanisms. Evolutionary computation is inspired by biological processes which are at work in nature. Genetic algorithm (GA) [1] modeled on the Darwinian evolutionary paradigm is the oldest and the best known evolutionary algorithm. It mimics the natural processes of selection, crossover, and mutation to search for optimum solutions in massive search spaces.

Particle swarm optimization (PSO) is a recently developed algorithm belonging to the class of biologically inspired methods [2–9]. PSO imitates the social behavior of insects, birds, or fish swarming together to hunt for food. PSO is a population-based approach that maintains a set of candidate solutions, called particles, which move within the search space. During the exploration of the search space, each particle maintains a memory of two pieces of information: the best solution (*pbest*) that it has encountered so far and the best solution (*gbest*) encountered by the swarm as a whole. This information is used to direct the search.

Researchers have found that PSO has the following advantages over the other biologically inspired evolutionary algorithms: (1) its operational principle is very simple and intuitive; (2) it relies on very few external control parameters; (3) it can be easily implemented; and (4) it has rapid convergence. PSO has developed very fast, has obtained very good applications in wide variety of areas [10–12], and has become one of the intelligent computing study hotspots in the recent years [12–14].

In the case of single-peak functions, PSO rapidly converges to the peak; however, in the case of multi-modal functions, the PSO particles are known to get trapped in the local optima. A significant number of variations are being made to the standard PSO to avoid the particles from getting trapped in the local optima [15–25]. Other methods which restart the particles trapped at the local optima have also been proposed [18, 19, 22]. In these methods, when the velocity of a particle falls below a given threshold, it is reinitialized to a randomly selected large value to avoid stagnation. The risk of stagnation is reduced by randomly accelerating the flying particles. Some of the more recent methods utilize multiswarms to search for the global optimum while avoiding getting trapped in the local optima [26–30].

However, in all the studies mentioned above, a number of *new parameters* are introduced in the original PSO. This destroys the simplicity of the algorithm and leads to an undesirable computational overhead. In this study, we introduce another variation to the evolution of the standard PSO but without resorting to additional new parameters. The strategy maintains the simple and intuitive structure as well as the implemental and computational advantages of the basic PSO. Thus, the contribution of this study is the improvement of the performance of the basic PSO without increasing the complexity of the algorithm.

This paper is organized as follows. In Section 2, we present the original PSO as proposed by Kennedy and Eberhart in 1948 [6]. In Section 3, we explain the working of the standard PSO and some of the variations found in the literature on PSO. In Section 4, we present our new parallel swarm oriented (PSO) and demonstrate the performance results in Section 5. We conclude the paper in Section 6 and propose some ideas for further research.

2. Original Particle Swarm Optimization

The population-based PSO conducts a search using a population of individuals. The individual in the population is called the particle, and the population is called the swarm. The performance of each particle is measured according to a predefined fitness function. Particles are assumed to "fly" over the search space in order to find promising regions of the landscape. In the minimization case, such regions possess lower functional values than other regions visited previously. Each particle is treated as a point in a d-dimensional space which adjusts its own "flying" according to its flying experience as well as the flying experience of the other companion particles. By making adjustments to the flying based on the local best (*pbest*) and the global best (*gbest*) found so far, the swarm as a whole converges to the optimum point, or at least to a near-optimal point, in the search space.

The notations used in PSO are as follows. The ith particle of the swarm in iteration t is represented by the d-dimensional vector, $x_i(t) = (x_{i1}, x_{i2}, \ldots, x_{id})$. Each particle also has a position change known as velocity, which for the ith particle in iteration t is $v_i(t) = (v_{i1}, v_{i2}, \ldots, v_{id})$. The best previous position (the position with the best fitness value) of the ith particle is $p_i(t-1) = (p_{i1}, p_{i2}, \ldots, p_{id})$. The best particle in the swarm, that is, the particle with the smallest function value found in all the previous iterations, is denoted by the index g. In a given iteration t, the velocity and position of each particle are updated using the following equations:

$$\begin{aligned} v_i(t) = {} & w v_i(t-1) + c_1 r_1 \left(p_i(t-1) - x_i(t-1)\right) \\ & + c_2 r_2 \left(p_g(t-1) - x_i(t-1)\right), \end{aligned} \tag{1}$$

$$x_i(t) = x_i(t-1) + v_i(t), \tag{2}$$

where $i = 1, 2, \ldots, \mathrm{NP}$; $t = 1, 2, \ldots, T$. NP is the size of the swarm, and T is the iteration limit; c_1 and c_2 are positive constants (called "social factors"); r_1 and r_2 are random numbers between 0 and 1; and w is the inertia weight that controls the impact of the previous history of the velocities on the current velocity, influencing the tradeoff between the global and local experiences. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration (fine-tuning the current search area). Equation (1) is used to compute a particle's new velocity, based on its previous velocity and the distances from its current position to its local best and to the global best positions. The new velocity is then used to compute the particle's new position (2).

3. Standard PSO and Its Variations

Any successful meta-heuristic algorithm maintains a delicate balance between exploration (diversifying the search to wider areas of the search space) and exploitation (intensifying the search in narrow promising areas). Shi and Eberhart later introduced the inertia weight ω in the original PSO to improve the PSO search [31]. A high value of the inertial weight favors exploration, while a low value favors exploitation. The inertia weight is defined as

$$\omega = (\omega_1 - \omega_2) \times \frac{(\mathrm{MaxIter} - \mathrm{Iter})}{\mathrm{MaxIter}} + \omega_2, \tag{3}$$

where ω_1 and ω_2 are, respectively, the initial and the final values of the inertia weight, Iter is the current iteration number, and MaxIter is the maximum number of iterations. Many studies have shown that the PSO performance is improved by decreasing ω linearly from 0.9 to 0.4 using the above equation.

Eberhart and Shi have further proposed a random modification for the inertia weight to make the standard PSO applicable to dynamic problems [4]. The inertia weight ω is randomly modified according to the following equation:

$$\omega = 0.5 + \frac{\mathrm{rand}\,(0,1)}{2}. \tag{4}$$

As opposed to the above linear decrement, Jie et al. [32] have proposed a nonlinear modification of the inertia weight over time given by

$$\omega = (\omega_1 - \omega_2) \sin\left[\frac{(\mathrm{MaxIter} - \mathrm{Iter})}{\mathrm{MaxIter}} \times \frac{\pi}{2}\right] + \omega_2. \tag{5}$$

This inertia weight varies slowly in the initial stages but more rapidly in the final stages. This implies that the algorithm makes a wider global search in the early stages and narrower local search in the final stages.

Generally, the very same velocity and position update formulae are applied to each and every flying particle. Moreover, the very same inertia weight is applied to each particle in a given iteration. However, Yang et al. [33] have proposed a modified PSO algorithm with dynamic adaptation, in which a modified velocity updating formula of the particle is used, where the randomness in the course of updating the particle velocity is relatively decreased and each particle has a *different* inertia weight applied to it in a given iteration. Further, this algorithm introduces two new parameters describing the

evolving state of the algorithm, the evolution speed factor and the aggregation degree factor. In the new strategy, the inertia weight is dynamically adjusted according to the evolution speed and the aggregation degree. The evolution speed factor is given by

$$h_i^t = \left| \frac{\min\left(F\left(pbest_i^{t-1}\right), F\left(pbest_i^t\right)\right)}{\max\left(F\left(pbest_i^{t-1}\right), F\left(pbest_i^t\right)\right)} \right|, \quad (6)$$

where $F(pbest_i^t)$ is the fitness value of $pbest_i^t$. The parameter h $(0 < h \leq 1)$ reflects the evolutionary speed of each particle. The smaller the value of h is, the faster the speed is.

The aggregation degree is given by

$$s = \left| \frac{\min\left(F_{tbest}, \overline{F_t}\right)}{\max\left(F_{tbest}, \overline{F_t}\right)} \right|, \quad (7)$$

where $\overline{F_t}$ is the mean fitness of all the particles in the swarm and F_{tbest} is the optimal value found in the tth iteration.

The inertia weight is updated as

$$\omega_i^t = \omega_{\text{ini}} - \alpha\left(1 - h_i^t\right) + \beta s, \quad (8)$$

where ω_{ini} is the initial inertia weight. The choice of α and β is typically in the range $[0, 1]$.

Another variation is the introduction of a constriction coefficient to replace ω to ensure the quick convergence of PSO [34]. The velocity update is given by

$$\begin{aligned} v_i(t) &= \chi\left(v_i(t-1) + c_1 r_1\left(p_i(t-1) - x_i(t-1)\right)\right. \\ &\quad \left. + c_2 r_2\left(p_g(t-1) - x_i(t-1)\right)\right) \\ \chi &= \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \end{aligned} \quad (9)$$

where χ is the constriction coefficient, $\varphi = c_1 + c_2$, and $\varphi > 4$.

4. Parallel Swarm Oriented PSO

In all the PSO variations mentioned in the preceding sections, a number of *new parameters* are introduced in the original PSO. This destroys the simplicity of the algorithm and leads to an undesirable computational overhead. In this section, we describe our novel approach called the parallel swarms oriented PSO (PSO-PSO). This version of the PSO does not introduce any new algorithm parameters to improve its convergence performance. The strategy maintains the simple and intuitive structure as well as the implemental and computational advantages of the basic PSO.

The algorithm consists of a multievolutionary phase and a single-evolutionary phase. In the multi-evolutionary phase, initially a number of sub-swarms are randomly generated so as to uniformly cover the decision space. The multiple sub-swarms are then allowed to evolve independently, each one maintaining its own particle-best (*pbest*) and swarm-best (*sbest*). The latter is a new term that we have introduced to represent the best particle in a given swarm in its history of evolution. After a predetermined number of evolutionary cycles, the multi-evolutionary phase ends and the single-evolutionary phase begins. The sub-swarms exchange information and record the global-best (*gbest*) of the entire collection of the sub-swarms. In the single-evolutionary phase all the subswarms merge and continue to evolve using the individual sub-swarm particle-best and swarm-best and the overall global-best. The sub-swarms then return to the multi-evolutionary phase and continue the search. The algorithm flow chart of the PSO-PSO algorithm is shown in Figure 1.

Multievolutionary Phase. K number of independent swarms evolve in parallel.

Step 1. Randomly generate K number of independent swarm populations so as to be uniformly distributed over the entire decision space.

Step 2. Evaluate the fitness of the particles in each individual swarm. In the minimization problems, the fitness of a particle is inversely proportional to the value of the function.

Step 3. Determine the particle-best (*pbest*) and the swarm-best (*sbest*) of each individual swarm.

Step 4. Update the velocity and position of each particle in each swarm according to (1) and (2).

Step 5. Allow the individual swarms to evolve independently through N iterations (i.e., repeat Steps 2 through 4).

Single-Evolutionary Phase. The individual swarms exchange information.

Step 6. Determine the global-best (*gbest*) by comparing the swarm-best (*sbest*) of all the swarms. For minimization problems, the *gbest* in a given iteration is given by the following equation:

$$gbest = \min\left(sbest_1, sbest_2, sbest_3, \ldots, sbest_k\right). \quad (10)$$

Step 7. The individual swarms start interacting by using the *gbest* as reference. Update the velocities of all the particles according to the following equation:

$$\begin{aligned} v_{ij}(t) &= w v_{ij}(t-1) + c_1 r_1\left(p_{ij}(t-1) - x_{ij}(t-1)\right) \\ &\quad + c_2 r_2\left(p_{sj}(t-1) - x_{ij}(t-1)\right) \\ &\quad + c_3 r_3\left(p_g(t-1) - x_{ij}(t-1)\right), \end{aligned} \quad (11)$$

where x_{ij} is the position of the ith particle in the jth swarm, v_{ij} is the velocity of the ith particle in the jth swarm, p_{ij} is the *pbest* of the ith particle in the jth swarm, p_{sj} is the *sbest* of the jth swarm, and p_g is the global-best of the entire information-exchanging collection of sub-swarms. c_1, c_2, c_3 are the acceleration parameters, and r_1, r_2, r_3 are uniform random numbers.

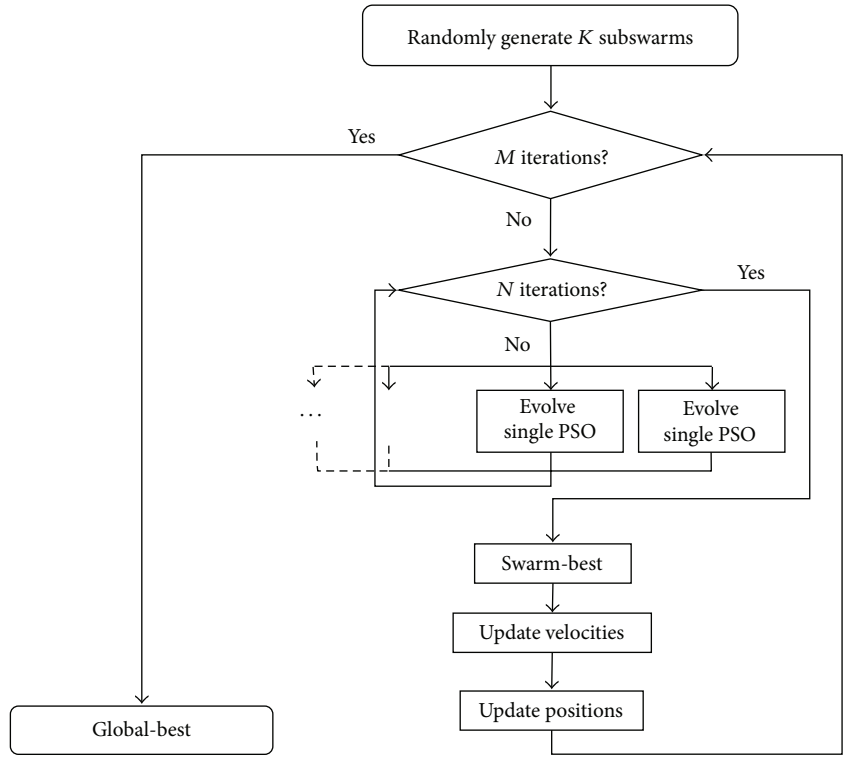

Figure 1: Parallel swarms oriented PSO algorithm flowchart.

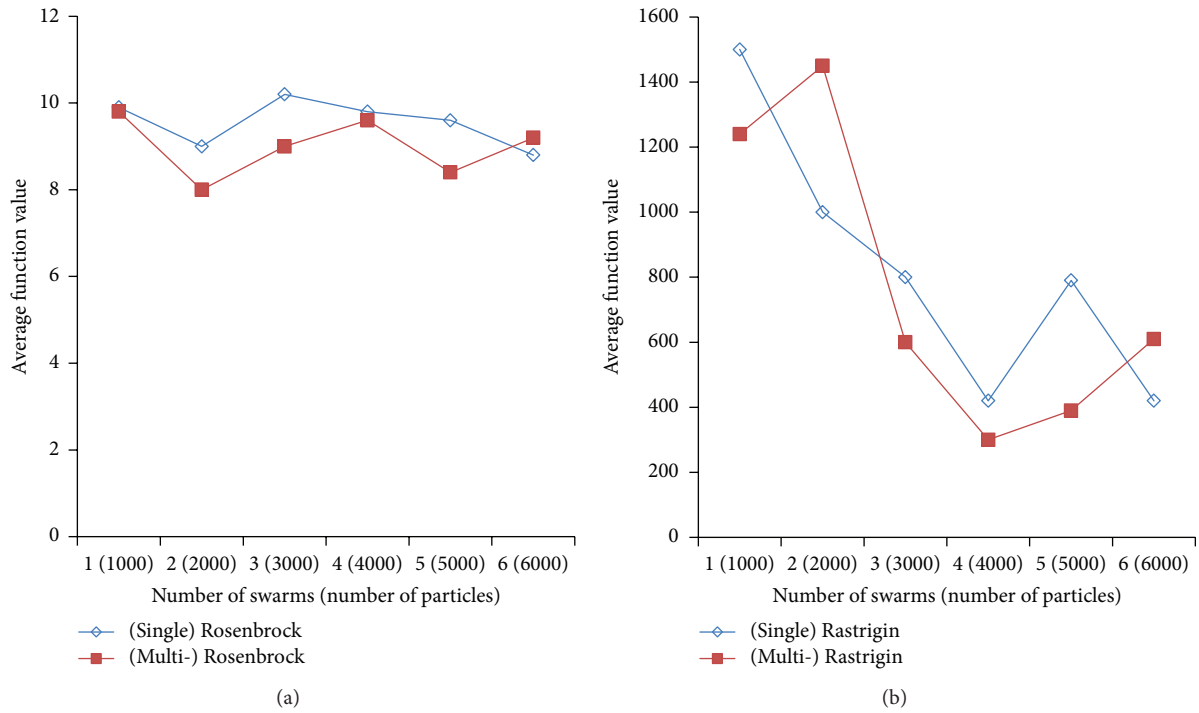

Figure 2: (a) 10D Rosenberg and (b) 10D Rastrigin.

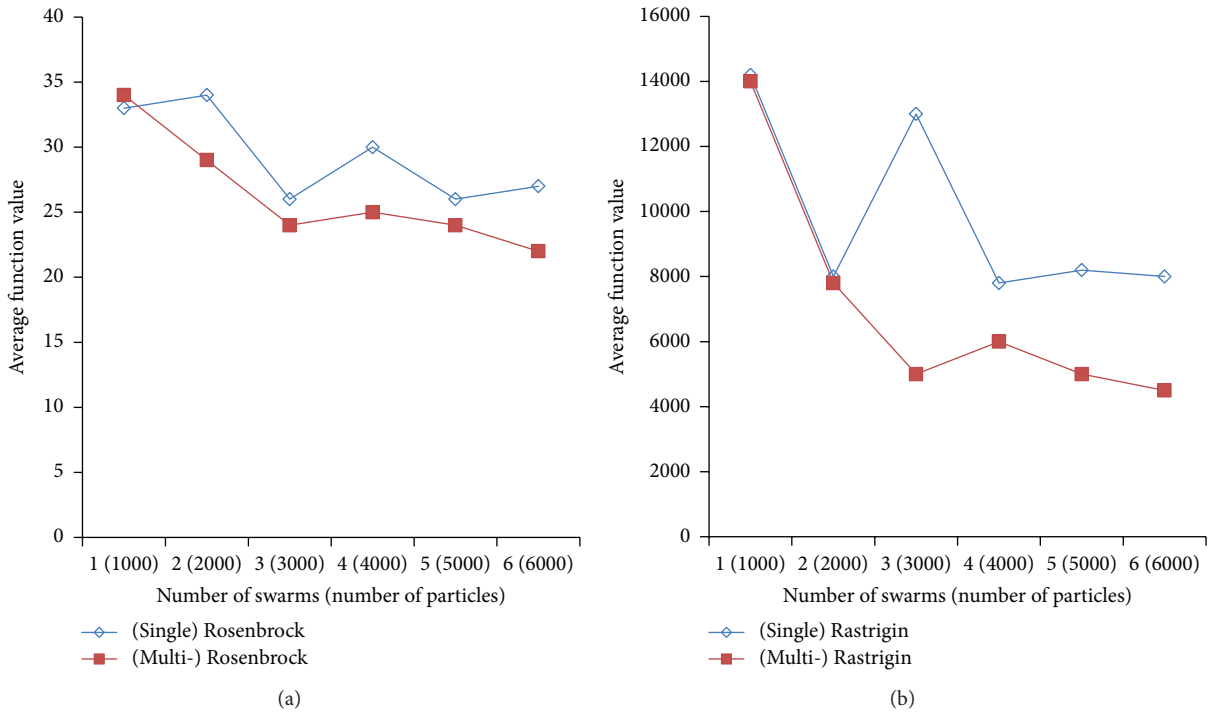

Figure 3: (a) 20D Rosenbrock and (b) 20D Rastrigin.

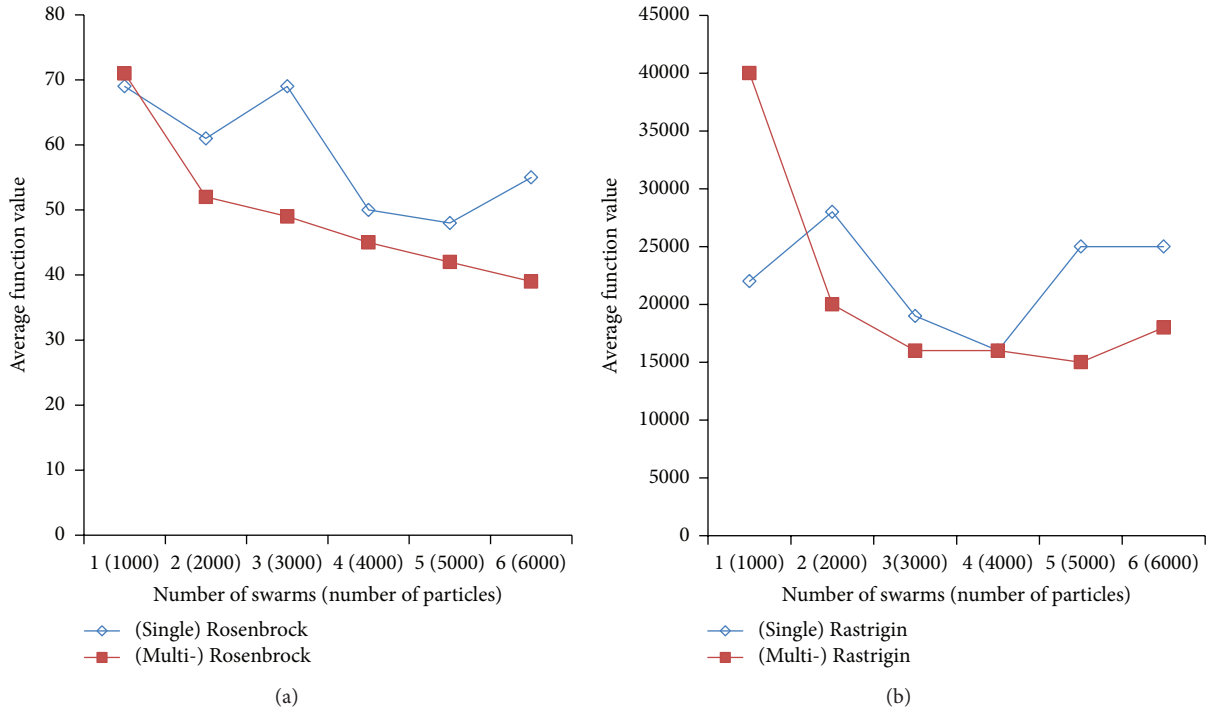

Figure 4: (a) 30D Rosenbrock and (b) 30D Rastrigin.

Step 8. Update the positions of all the particles according to the following equation:

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t). \quad (12)$$

Step 9. Repeat Steps 2–8 through M iterations and output the global-best.

5. Performance Comparison of Single and Multiswarm PSO

The PSO-PSO algorithm is implemented in MATLAB on a 24-core server with 256 GB RAM. Each CPU core is dedicated to the evolution of a single sub-swarm in the multi-stage computational phase of the algorithm. The results are presented in Figures 2, 3, and 4. For smaller dimensions, there is no appreciable difference between the performance of the ordinary (single) PSO and the multi-swarm PSO-PSO. This can be seen in the optimization results of the 10-dimensional Rosenbrock and Rastrigin functions (Figures 2(a) and 2(b)). However, the superior performance of the multi-swarm PSO-PSO approach is evident in higher dimensions. This can be seen in the optimization of the 20- (Figures 3(a) and 3(b)) and 30-dimensional Rosenbrock and Rastrigin functions (Figures 4(a) and 4(b)).

6. Conclusion

The particle swarm optimization (PSO) is increasingly becoming widespread in a variety of applications as a reliable and robust optimization algorithm. The attractive features of this evolutionary algorithm is that it has very few control parameters, is simple to program, and is rapidly converging. The only drawback reported so far is that at times it gets trapped in local optima. Many researchers have addressed this issue but by introducing a number of new parameters in the original PSO. This destroys the simplicity of the algorithm and leads to an undesirable computational overhead. In this study, we have proposed a variation of the algorithm called parallel swarms oriented PSO (PSO-PSO) which consists of a multi-stage and a single stage of evolution. The two interweaved stages of evolution demonstrate better performance on test functions, especially of higher dimensions. The attractive feature of PSO-PSO version of the algorithm is that it does not introduce any new parameters to improve its convergence performance. The PSO-PSO strategy maintains the simple and intuitive structure as well as the implemental and computational advantages of the basic PSO. Thus, the contribution of this study is the improvement of the performance of the basic PSO without increasing the complexity of the algorithm.

References

[1] J. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass, USA, 1992.

[2] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[3] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, pp. 84–88, July 2000.

[4] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the Congress on Evolutionary Computation (CEC '01)*, pp. 94–100, San Francisco, Cailf, USA, May 2001.

[5] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation (CEC '01)*, pp. 81–86, May 2001.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1–6, pp. 1942–1948, December 1995.

[7] J. Kennedy and R. Eberhart, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[8] J. Kennedy and R. C. Eberhart, "Discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4104–4108, October 1997.

[9] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[10] M. R. Alrashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in power system operations," *Electric Power Components and Systems*, vol. 34, no. 12, pp. 1349–1357, 2006.

[11] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: a brief survey," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 41, no. 2, pp. 262–267, 2011.

[12] L. Wang, J. Shen, and J. Yong, "A survey on bio-inspired algorithms for web service composition," in *Proceedings of the IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD '12)*, pp. 569–574, 2012.

[13] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[14] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[15] G. M. Chen, J. Y. Jia, and Q. Han, "Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm," *Journal of Xi'an Jiaotong University*, vol. 40, pp. 1039–1042, 2006.

[16] Z.-S. Lu and Z.-R. Hou, "Particle swarm optimization with adaptive mutation," *Acta Electronica Sinica*, vol. 32, no. 3, pp. 416–420, 2004.

[17] F. Pan, X. Tu, J. Chen, and J. Fu, "A harmonious particle swarm optimizer—HPSO," *Computer Engineering*, vol. 31, no. 1, pp. 169–171, 2005.

[18] S. Pasupuleti and R. Battiti, "The gregarious particle swarm optimizer—G-PSO," in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference (CEC '06)*, pp. 67–74, July 2006.

[19] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[20] J. F. Schutte and A. A. Groenwold, "A study of global optimization using particle swarms," *Journal of Global Optimization*, vol. 31, no. 1, pp. 93–108, 2005.

[21] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 101–106, Seoul, Republic of Korea, May 2001.

[22] K. Tatsumi, T. Yukami, and T. Tanino, "Restarting multi-type particle swarm optimization using an adaptive selection of particle type," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '09)*, pp. 923–928, October 2009.

[23] Y.-L. Zheng, L.-H. Ma, L.-Y. Zhang, and J.-X. Qian, "On the convergence analysis and parameter selection in particle swarm optimization," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 1802–1807, Zhejiang University, Hangzhou, China, November 2003.

[24] L. P. Zhang, H. J. Yu, D. Z. Chen, and S. X. Hu, "Analysis and improvement of particle swarm optimization algorithm," *Information and Control*, vol. 33, pp. 513–517, 2004.

[25] X. Zhang, Y. Du, G. Qin, and Z. Qin, "Adaptive particle swarm algorithm with dynamically changing inertia weight," *Journal of Xi'an Jiaotong University*, vol. 39, no. 10, pp. 1039–1042, 2005.

[26] T. M. Blackwell and J. Branke, "Multi-Swarm optimization in dynamic environment," in *Lecture Notes in Computer Science*, vol. 3005, pp. 489–500, Springer, Berlin, Germany, 2004.

[27] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[28] K. Chen, T. Li, and T. Cao, "Tribe-PSO: a novel global optimization algorithm and its application in molecular docking," *Chemometrics and Intelligent Laboratory Systems*, vol. 82, no. 1-2, pp. 248–259, 2006.

[29] B. Niu, Y. Zhu, X. He, and H. Wu, "MCPSO: a multi-swarm cooperative particle swarm optimizer," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1050–1062, 2007.

[30] L.-Y. Wu, H. Sun, and M. Bai, "Particle swarm optimization algorithm of two sub-swarms exchange based on different evolvement model," *Journal of Nanchang Institute of Technology*, vol. 4, pp. 1–4, 2008.

[31] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 1945–1950, IEEE Service Center, Piscataway, NJ, USA, 1999.

[32] J. Jie, W. Wang, C. Liu, and B. Hou, "Multi-swarm particle swarm optimization based on mixed search behavior," in *Proceedings of the 5th IEEE Conference on Industrial Electronics and Applications (ICIEA '10)*, pp. 605–610, June 2010.

[33] X. Yang, J. Yuan, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation," *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1205–1213, 2007.

[34] M. Clerc, *Particle Swarm Optimization*, ISTE Publishing Company, London, UK, 2006.