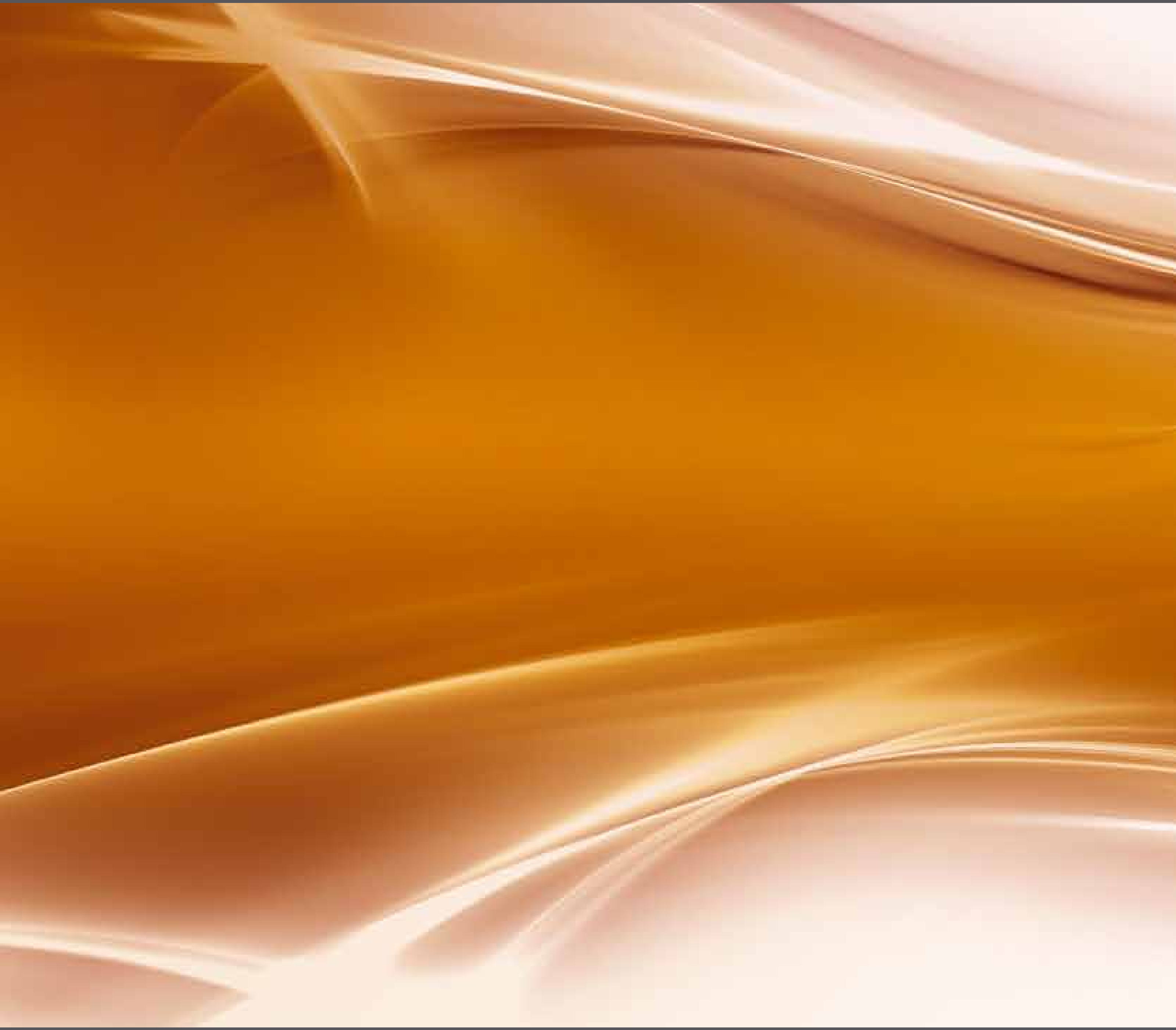
Applied Computational Intelligence and Soft Computing

Theory and Applications of Evolutionary Computation

Guest Editors: Tzung-Pei Hong, Chuan-Kang Ting, and Oliver Kramer



Theory and Applications of Evolutionary Computation

Applied Computational Intelligence and Soft Computing

Theory and Applications of Evolutionary Computation

Guest Editors: Tzung-Pei Hong, Chuan-Kang Ting, and Oliver Kramer

Contents

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 360796, 2 pages
doi:10.1155/2010/360796

Editorial

Theory and Applications of Evolutionary Computation

Tzung-Pei Hong,[1] Chuan-Kang Ting,[2] and Oliver Kramer[3]

[1] *Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan*
[2] *Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi 621, Taiwan*
[3] *Department of Computer Science, Dortmund University of Technology, 44221 Dortmund, Germany*

Correspondence should be addressed to Tzung-Pei Hong, tphong@nuk.edu.tw

Received 25 May 2010; Accepted 25 May 2010

Evolutionary computation is a powerful problem solver inspired from natural evolution. It models the essential elements of biological evolution and explores the solution space by gene inheritance, mutation, and selection of the fittest candidate solutions. The dialects of evolutionary algorithms include genetic algorithms, evolutionary strategies, genetic programming, particle swarm optimization, ant colony optimization, artificial immune systems, estimation of distribution algorithms, differential evolution, and memetic algorithms. These evolutionary methods have proven their success on various hard and complex optimization problems.

This *special issue* on Theory and Applications of Evolutionary Computation is dedicated to latest developments in the area of evolutionary computation. Eight articles from researchers around the world contribute to further steps into the understanding and application of evolutionary computation. The special issue covers a broad bandwidth of research, from theoretical investigations to real-world applications, and comprises articles in active research areas like multiobjective and constrained optimization.

Toward improving evolutionary algorithms based on theoretical finding, the paper by Pepper investigates the selection efficiency of threshold selection, stochastic proportionate selection, and deterministic proportionate selection. The next paper, by Browne and dos Santos, introduces flexible adaptive genome representations for gene expression programming that allow parallelization and maintenance of population diversity. The ability of parallelization is the most important advantage of evolutionary approaches in comparison to many other optimization heuristics.

In the applications of evolutionary computation, the paper by Kawabe presents an evolutionary controller for the receding horizon control problem—an advanced method of process control that has been used in the process industries such as chemical plants. The simulation results prove that stochastic optimizers are strong problem solvers for complex practical problems. The paper by Shi et al. devises a twin-screw coded evolutionary algorithm for the multilevel production scheduling problem. The comparative study shows that the proposed method outperforms genetic algorithm and tabu search in solution quality. The paper by De Falco et al. proposes a distributed differential evolution algorithm to address the multisite grip mapping problem. According to the experimental results, this method can minimize the consumption of grid resources.

This special issue includes two review papers. The article by Kramer gives a survey of constraint handling techniques that have developed in the field of evolution strategies in the last years, in particular concentrating on the prevention of premature step size stagnation that can often be observed at the boundary of the infeasible solution space. In addition, the work by Gong et al. gives a comprehensive overview of evolutionary gait optimization. The authors reviewed several success stories of evolutionary methods in evolving programs for legged robots and concluded that the domain is still a fruitful field of research.

We thank all the authors and reviewers for their great contributions to this special issue. We would also like to thank Professor Hsien-Chung Wu, the editor-in-chief, for his full support.

Tzung-Pei Hong
Chuan-Kang Ting
Oliver Kramer

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 696345, 7 pages
doi:10.1155/2010/696345

Research Article

Efficient Use of Variation in Evolutionary Optimization

John W. Pepper

Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501, USA

Correspondence should be addressed to John W. Pepper, jpepper@santafe.edu

Received 19 July 2009; Accepted 6 January 2010

Academic Editor: Chuan-Kang Ting

Evolutionary algorithms face a fundamental trade-off between exploration and exploitation. Rapid performance improvement tends to be accompanied by a rapid loss of diversity from the population of potential solutions, causing premature convergence on local rather than global optima. However, the rate at which diversity is lost from a population is not simply a function of the strength of selection but also its efficiency, or rate of performance improvement relative to loss of variation. Selection efficiency can be quantified as the linear correlation between objective performance and reproduction. Commonly used selection algorithms contain several sources of inefficiency, some of which are easily avoided and others of which are not. Selection algorithms based on continuously varying generation time instead of discretely varying number of offspring can approach the theoretical limit on the efficient use of population diversity.

1. Introduction

"Premature convergence", or the loss of diversity before a satisfactory solution is found, is a persistent problem in evolutionary optimization [1]. This reflects the fundamental trade-off between exploration and exploitation, or between thoroughness and speed in evolutionary search [2]. If selection is too weak, progress is slow and many generations are required to find a solution. On the other hand, if selection is too strong, the population rapidly loses diversity and may become stranded on a local fitness peak. A wide variety of techniques have been proposed to address this problem, but it has generally been approached on an *ad hoc* empirical basis, and little theory has been available to guide the design of selection algorithms.

While the trade-off between improving performance and preserving diversity cannot be avoided, it can be ameliorated through the efficient use of variation. Diversity within a population acts as the fuel of the selection process: it is required for selection to act, but is itself consumed in the process. However, selection algorithms differ not only in speed, but also in "fuel efficiency", or rate of improvement relative to loss of variation. In the following sections, I develop a method for quantifying the efficiency of fitness functions, defined here as mappings from objective performance to reproduction. (Such mappings are sometimes referred to as "selection methods".) The approach is based on the powerful formalism from evolutionary biology known as the "Price equation", which is increasingly used in evolutionary genetics [3]. I next compare several widely-used selection methods to characterize their sources of inefficiency, and to illustrate the advantages of more efficient selection. I also consider whether less efficient algorithms have any offsetting advantages that justify their use. Finally, I discuss the design of fast and efficient fitness functions, and propose a new kind of algorithm, based on varying generation time instead of number of offspring, which can approach perfect efficiency in the use of genetic variation.

2. Quantifying Selection Efficiency

The ultimate goal of evolutionary optimization is to maximize some objective measure of performance on a given task. Here I measure progress toward optimization in terms of the mean performance level of the population (In evolutionary computation applications, the ultimate interest may be in the highest performance level in a population of candidate solutions, rather than the mean. However, mathematical theory is only available to quantify change in population mean through selection rather than change in population maximum. As a practical matter, maximizing mean performance will also maximize best performance, all else being equal). The goal of improving performance conflicts partially with a subsidiary goal: maintaining the diverse population of candidate solutions or "individuals" needed to thoroughly explore search spaces and find the best possible solutions. The conflict arises because the unequal reproduction that drives improvement in average performance also reduce population diversity. Unequal contributions to the next generation's gene pool by different individuals always reduces diversity except in the special case of negative frequency-dependent selection (which increases diversity). If selection is frequency-independent, unequal reproduction reduces diversity, in direct proportion to the reproductive variance among individuals (see the appendix).

Although selection cannot improve a population's average performance in the next generation without unequal reproduction, the converse is not true. Unequal reproduction and resulting loss of diversity need not improve average performance. Variance in reproduction that is uncorrelated with performance can reduce genetic diversity (though genetic drift) just as quickly as can effective selection, but without increasing mean performance. Because correlation between performance and reproduction is what makes selection effective at optimization, I focus on the strength of this correlation to quantify the efficiency of fitness functions.

In addition to selection, genetic operators such as mutation and recombination can also change a population's mean performance (although in an unpredictable direction). Here I focus exclusively on the effects of selection, or differential reproduction, because this is the source of premature convergence in evolutionary optimization. Let each individual in the population (indexed by i) have a measured performance level p_i. The average population performance before selection is $\overline{p} = \sum p_i/N$, where N = population size. After one generation of selection, average population performance will be the average of the parent performances weighted by the contribution of each parent to the next generation: $\overline{p}' = \sum p_i w_i / \sum w_i$, where w_i = the number of offspring produced by the i'th individual. (Note that this assumes perfect heritability of performance from parent to offspring.) To simplify the notation, it is convenient to replace absolute reproduction w_i with relative reproduction, $\widetilde{w}_i = w_i/\overline{w}$, so that mean performance in the offspring generation is $\overline{p}' = \text{ave}(p_i\widetilde{w}_i)$. The change in average performance caused by one round of selection is then $\Delta\overline{p} = \overline{p}' - \overline{p}$, or

$$\Delta\overline{p} = \text{ave}(p\widetilde{w}) - \text{ave}(p). \tag{1}$$

As a result of selection, performance improvement across one generation is exactly $\Delta\overline{p}$ above. We can rewrite (1) in a useful form by using two identities: firstly, $\text{ave}(p\widetilde{w}) = \text{ave}(p) \cdot \text{ave}(\widetilde{w}) + \text{cov}(p\widetilde{w})$, where "cov" represents covariance. Secondly, $\text{ave}(\widetilde{w}) = 1$ by definition. With these substitutions, the improvement in performance from parent to offspring generation is

$$\Delta\overline{p} = \text{cov}(p, \widetilde{w}) \tag{2}$$

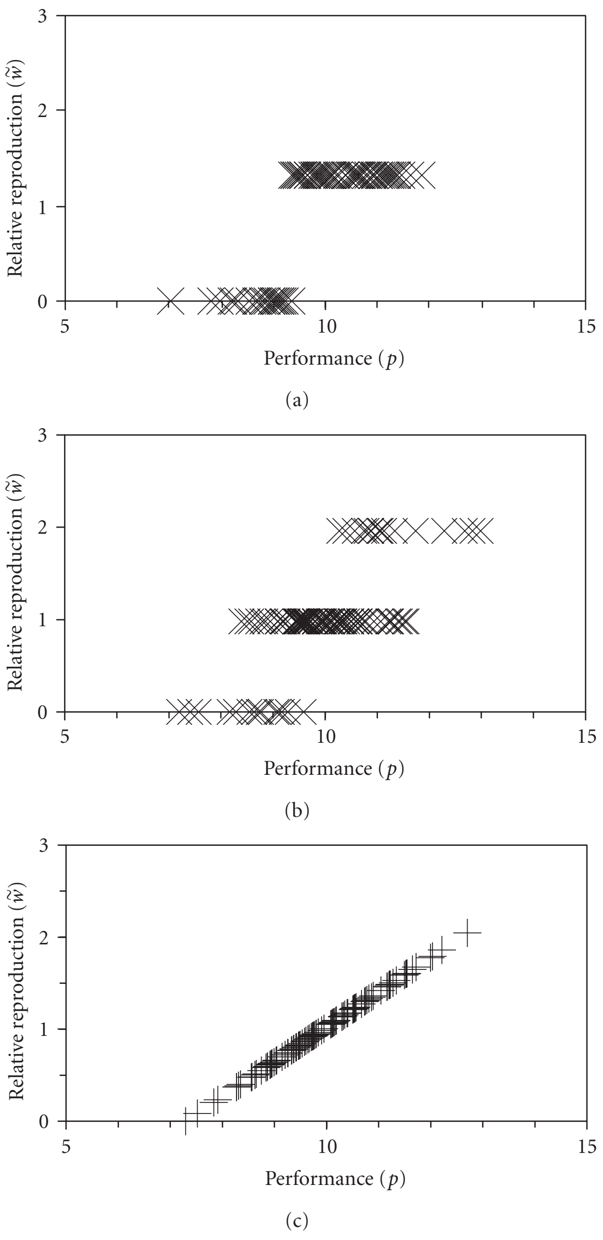


Figure 1: Three fitness functions illustrated using the same set of 100 simulated individuals with performance values drawn from a normal distribution with mean = 10 and standard deviation = 1. (a) threshold selection (b) stochastic proportionate selection (SPS), (c) deterministic proportionate selection (using (8)). Each mark represents one individual.

(see [4]). To highlight the factors affecting optimization rate, it is useful to use another identity to rewrite this covariance as a product of its three factors:

$$\Delta\overline{p} = \sigma_p \cdot \sigma_{\widetilde{w}} \cdot \rho_{p\widetilde{w}}, \tag{3}$$

where σ is a standard deviation among individuals in performance (p) or relative reproduction ($\widetilde{w}$), and $\rho_{p\widetilde{w}}$ is the linear correlation coefficient between the two [4].

Equation (3) provides insight into how to maximize selection efficiency, or the ratio of performance improvement

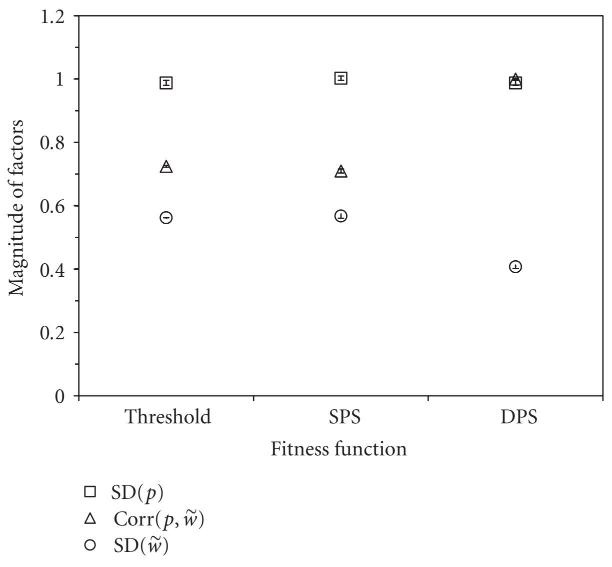


Figure 2: The three factors contributing to performance improvement compared over 1 round of selection across three fitness functions using numerical simulations: threshold selection, stochastic proportionate selection (SPS), and deterministic proportionate selection (DPS). Each sample consisted of 100 simulated individuals with performance values drawn from a normal distribution with mean = 100, SD = 1. Markers show means, and bars show ± standard error over 100 samples. (Note that error bars are too small to extend beyond marker symbols.)

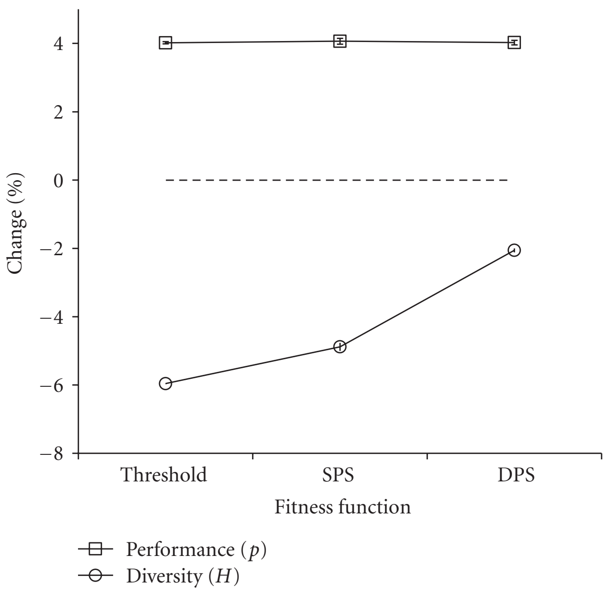


Figure 3: The same three fitness functions shown in Figure 2 compared for the one-generation change produced in mean performance, and in population diversity. Bars show standard errors. (Note that error bars are too small to extend beyond marker symbols).

to loss of diversity. Deviation in individual performance (σ_p) is fixed for a given population, but $\sigma_{\tilde{w}}$ and $\rho_{p\tilde{w}}$ depend on the selection method. Deviation in reproduction ($\sigma_{\tilde{w}}$) varies with the strength of selection. Increasing $\sigma_{\tilde{w}}$ can increase performance improvement, but at the cost of faster loss of diversity. The linear correlation between performance and reproduction ($\rho_{p\tilde{w}}$) corresponds to the efficiency of selection, in the sense that increasing this term increases performance improvement *without* increasing loss of diversity and performance variation. When $\rho_{p\tilde{w}} = 0$, selection is completely inefficient: it consumes variation without improving average performance. In the language of evolutionary theory, this is termed “drift” instead of “selection”. At the other extreme of $\rho_{p\tilde{w}} = 1$, the ratio of performance increase to variance reduction is maximized. Thus the rate at which variation is lost from a population is not simply a function of selection strength ($\sigma_{\tilde{w}}$), as is sometimes assumed, but also of selection efficiency ($\rho_{p\tilde{w}}$).

3. Sources of Inefficiency in Fitness Functions

The perfectly linear fitness function ($\rho_{p\tilde{w}} = 1$) is an ideal of efficiency that is not realized by any algorithm in general use. All standard fitness functions depart from linear correlation either through deterministic nonlinearities, fluctuating stochastic nonlinearities, or both. An example of a deterministically nonlinear fitness function is threshold selection, in which reproduction is an all-or-nothing step function of performance (Figure 1(a)). Any such highly nonlinear fitness function will necessarily have a linear correlation well below 1. Fitness functions without any deterministic nonlinearity are termed “fitness-proportionate selection” because expected reproduction is directly proportional to performance [1]. However, these functions introduce fluctuating stochastic nonlinearity in converting expected to actual reproduction, so that expected reproduction has perfect linear correlation with performance, but actual reproduction does not. This is hard to avoid because unlike the expected number of offspring, the actual number of offspring is constrained to whole numbers and so must vary stochastically around the expected number. For example, the commonly used “stochastic universal sampling” algorithm [5] works as follows: an expected reproduction of ω is partitioned into a fractional portion (ω%1) and a whole-number portion [$\omega-(\omega$%1)], where % is the modulo operator. The algorithm produces the whole number of offspring, plus one additional offspring with a fractional probability of (ω%1). Despite its lack of deterministic nonlinearity, the correlation between performance and actual number of offspring is less than 1 because of stochastic fluctuations (e.g., Figure 1(b), where $\omega = 1$ for each individual, but w varies stochastically). I will refer to this algorithm as “stochastic proportionate selection” (SPS).

Such stochastic fluctuations in actual reproduction are larger in other implementations of fitness-proportionate selection, such as “roulette wheel” sampling [2]. Still other algorithms, such as tournament selection [1], include both deterministic and stochastic sources of nonlinearity. Here the selection of a pair of individuals to compare is stochastic, while the choice of which of the two reproduces depends on their relative performance rank, which is a deterministic nonlinear function of performance. Both deterministic and stochastic nonlinearities in fitness functions reduce the correlation between performance and actual reproduction, and thereby reduce selection efficiency.

To examine the effect of selection efficiency on diversity, I used a numerical simulation consisting of a population of 100 individuals (candidate solutions) with performance values drawn from a normal distribution with mean = 10 and standard deviation = 1. I compared the effects of a single round of selection using threshold selection (Figure 1(a)), stochastic proportionate selection (SPS) (Figure 1(b)), and deterministic proportionate selection (DPS) ((8), Figure 1(c)). The numerical simulation allowed fractional offspring, but the problem of how deterministic proportionate selection can be implemented with whole numbers of individuals is deferred to Section 6 below. To tune the threshold fitness function to give the same performance improvement as the other two functions, I allowed reproduction only by the best-performing 76% of the population. Deterministic proportionate selection generated less variance in reproduction than the other two, but reproduction was more highly correlated with performance (Figure 2). These two differences resulted in an equal performance increase in the offspring generation for all three fitness functions (Figure 3). Thus the deterministic proportionate selection function consumed less performance variation while producing the same performance improvement. I next investigated whether DPS also preserved more genotype diversity while producing the same performance improvement.

To quantify diversity, I used the Shannon-Weiner diversity index from evolutionary biology, which is equivalent to the entropy of the genotypes in the population:

$$H = -\sum_g f_g \log_2 f_g, \tag{4}$$

where g indexes the genotypes in the population, and f_g is the population frequency of genotype g. Entropy is maximized when each individual is unique, and minimized when all individuals share the same genotype. To simplify calculations I assumed that each individual in the population was unique prior to selection, but violating this assumption would not change the outcome qualitatively. Selection reduced diversity several-fold less under the deterministic proportionate function than under either the stochastic proportionate or threshold functions, while improving performance at the same rate (Figure 3).

4. Is Inefficient Selection Ever Useful?

I have focused here on the advantages of linear fitness functions for conserving genetic diversity. However, both deterministic nonlinearities and stochastic effects have some potential advantages. Might these justify the use of nonlinear fitness functions despite their lower efficiency?

Deterministically nonlinear fitness functions permit stronger selection (higher $\sigma_{\tilde{w}}$) than linear functions. At the extreme, reproduction by only the individual(s) with the highest performance increases average performance by $\Delta\overline{p} = p_{\max} - \overline{p}$. More generally, larger one-generation improvements are possible with nonlinear than with linear fitness functions. However, this rapid short-term improvement comes at the cost of the variation required for longer-term improvement. Genetic variation could be created anew in each generation, but this is computationally expensive and reduces evolutionary search algorithms to inefficient hill-climbers. For this reason, deterministic nonlinearity in fitness functions is unlikely to be helpful in most applications.

Stochastic fitness functions offer a different potential advantage by helping populations escape from local performance peaks. Slightly deleterious mutations can persist or spread under stochastic selection, making it possible for populations to cross low-performance fitness valleys requiring multiple mutations. Stochastic effects also allow the population to drift among different genotypes with equal performance. This may facilitate the exploration of "neutral networks" in genotype space, leading to the discovery of higher performance peaks [6]. However, stochastic effects on reproduction also have drawbacks. They can push populations away from global as well as local peaks. In some algorithms, they may also slow the discovery of higher-performance peaks by allowing beneficial new mutations to be lost. It remains an open question how often stochastic fitness functions improve evolutionary optimization, and how much stochasticity is desirable. To investigate these questions, it will help to have algorithms in which stochastic effects can be directly controlled by the experimenter rather than being a by-product of the particular algorithm used. This is easily achieved by adding a stochastic term to a deterministic linear fitness function. This approach has the additional advantage that stochastic effects can be reduced to any desired magnitude without incurring a computational cost. In contrast, intrinsically stochastic algorithms require very large population sizes to drive stochastic effects to low levels.

5. Fast and Efficient Fitness Functions

How can a fitness function be designed to maximize the rate of performance increase while also optimizing efficiency? Efficiency defined as the linear correlation $\rho_{p\tilde{w}}$ is maximized when reproduction is a linear function of performance. It is convenient to represent such fitness functions in the standard linear form:

$$w_i = a(p_i + b), \tag{5}$$

where p_i and w_i are individual performance and reproduction, respectively, and a and b are system parameters. With discrete generations, it is usually desirable to maintain a stable population size across generations, which constrains the average number of offspring per individual ($\overline{w}$) to 1. This constrains the value of a to

$$a = \frac{1}{\text{ave}(p_i + b)} = \frac{1}{\overline{p} + b}. \tag{6}$$

Substituting (6) into (5) gives us a linear fitness function yielding a stable population size:

$$w_i = \frac{(p_i + b)}{(\overline{p} + b)}. \tag{7}$$

What value of b will maximize the rate of performance improvement? Recall from (3) that the one-generation improvement in average performance due to selection is a product of three quantities: σ_p, $\rho_{p\tilde{w}}$, and $\sigma_{\tilde{w}}$. The first of these is a fixed property of the population. The second is already maximized at 1 under linear fitness functions. This leaves only variance in individual reproduction$\sigma_{\tilde{w}}$ to be maximized in order to maximize the performance improvement Δp. When w_i is a linear function of p_i, its variance σ_w is maximized by maximizing the slope of the fitness function, which is defined in (5) as a. Equation (6) shows that a increases as b approaches $-\overline{p}$, so that b should be as close as possible to $-\overline{p}$ to maximize improvement. However, there is a constraint that individual reproduction (w_i) cannot be negative, which means that $b \geq -p_i$ for all i (5). If the worst performance in the population is denoted as $p_{\min}$, then the lowest possible value for b is $-p_{\min}$, which results in the individual(s) with the lowest performance having exactly zero offspring. Substituting this value for b into (7) yields the stable linear fitness function with the maximum rate of performance increase:

$$w_i = \frac{(p_i - p_{\min})}{(\overline{p} - p_{\min})}. \tag{8}$$

6. A Variable-Generation Algorithm for Efficient Selection

If a deterministic linear fitness function is the theoretical ideal, how can it be implemented in practice? As discussed above, inefficiency in commonly used fitness functions arises in part from easily avoidable sources of nonlinearity. However, all standard algorithms also contain nonlinearities arising from the fact that performance is a continuous variable, while the number of offspring is discrete. Stochastically converting real numbers of expected offspring to whole numbers of actual offspring reduces the linear correlation between performance and actual reproduction.

We can overcome this problem by recognizing that selection on genotypes acts through their rate of reproduction per unit time. Instead of varying the number of offspring, one can independently vary the generation time for each individual [7]. This requires an algorithm incorporating overlapping generations and a continuous representation of time. Individual reproductive rates can then vary continuously rather than discretely, and can correlate perfectly with individual performance.

To implement this idea, individual reproduction is treated as a growth rate, by analogy with population growth rates. A population growth rate tells us how large a population will be after a given time:

$$s_t = s_0 w^t, \tag{9}$$

where s_0 is initial population size, s_t is population size after t time units, and w is growth rate. Rearranging (9) tells us how long it will take the population size to change by a given factor s_t/s_0 under a given growth rate w:

$$t = \frac{\ln(s_t/s_0)}{\ln(w)}. \tag{10}$$

Our current problem concerns individuals rather than populations, but we can use the same reasoning to ask how long it will take an individual to die (equivalent to shrinking to size zero) or reproduce (equivalent to doubling in size) as a function of its individual growth rate w_i. Because individuals are discrete, we round off individual "size" to the nearest whole number. Thus for $w_i < 1$, we can ask how long it will take for the individual to fall below half its initial size, given its negative growth rate. At this point, the individual's size is closer to zero than one, and we recognize this by removing it from the population. Similarly, if an individual's growth rate is greater than one, we ask how long it will take for its size to rise above 1.5. At this point it is closer to being two individuals than one, and we recognize this by doubling it via reproduction. (Note that unlike rounding the number of offspring under stochastic fitness-proportionate algorithms, rounding individual size to whole numbers is not stochastic and does not introduce stochastic nonlinearity into the fitness function. Because waiting times vary continuously, genotype growth rates also vary continuously as a deterministic linear function of performance.)
For $w < 1$, waiting time to death is found by substituting 0.5 for s_t/s_0 in (10), giving

$$t_d = \frac{-0.693}{\ln(w)}. \tag{11}$$

For $w > 1$, waiting time to reproduction is found by substituting 1.5 for s_t/s_0, giving:

$$t_r = \frac{0.405}{\ln(w)}. \tag{12}$$

When an individual's reproductive rate is evaluated, its future death or reproduction is scheduled for a time point in the future designated as a real number on a time line. These events will be scheduled in the distant future when the reproductive rate is close to 1, and in the near future when it is far from 1 (Figure 4).

At the beginning of a run, each individual's performance is evaluated and its reproduction or death is scheduled. After this, the algorithm simply consists of repeatedly cycling through the following steps: (1) carry out the first event on the schedule. (2) If the event was a birth, evaluate the new individual's performance. (3) recalculate all waiting times

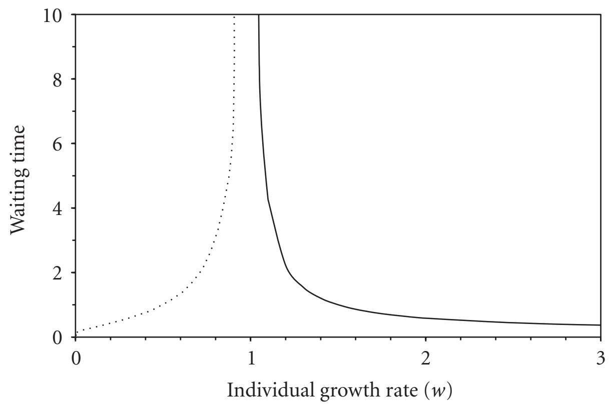


FIGURE 4: Waiting time to death (dotted line) or reproduction (solid line) as a function of individual growth rate. (From (11) and (12).)

to reflect the new average performance, and update the schedule. In practice, it might be useful to recalculate waiting times less often in order to reduce the computational load. For example, each individual's waiting time could be calculated at birth and then not recalculated until its scheduled event was within some specified time horizon.

7. Conclusions

In these results, truly linear fitness functions, in the form of deterministic proportionate selection, reduced population diversity and performance variation less than other fitness functions that improve performance the same amount in one round of selection. This strongly suggests that over multiple generations, the same rate of performance improvement would be sustained with less loss of diversity. Consequently, DPS should yield better solutions, particularly for tasks where premature convergence is otherwise a problem. The variable-generation algorithm outlined above allows actual reproductive rates to be exactly proportional to performance, providing one way to implement DPS. Although stochastic fitness functions may eventually prove useful on some fitness landscapes, intrinsically linear fitness functions provide the best foundation for designing them because they allow stochastic terms to be added in a controlled fashion.

One important caveat is that these conclusions are based on consideration of a single round of selection in isolation. Longer-term selection is also affected by the genetic operators that create variation, such as mutation and recombination, and by their interactions with selection. In particular, this paper does not address the issue of how selection interacts with recombination among epistatic loci (e.g., [8]). While I am not aware of any reason the conclusions reached here would not hold in the broader context of long-term evolution with recombination; this remains to be investigated.

Appendix

The purpose of this appendix is to quantify the extent to which unequal reproduction reduces diversity in a population. It will show that when selection is frequency-independent, unequal reproduction reduces diversity in direct proportion to the reproductive variance among individuals. I follow Section 3 above in quantifying diversity with the Shannon-Wiener diversity index, which is equivalent to the entropy of genotypes.

Before selection, variance in the frequencies of alternative genotypes is

$$\mathrm{var}(f) = \mathrm{E}(f^2) - \mathrm{E}(f)^2, \tag{A.1}$$

and after one round of selection it is

$$\mathrm{var}(\tilde{w}f) = \mathrm{E}(\tilde{w}^2 f^2) - \mathrm{E}(\tilde{w}f)^2, \tag{A.2}$$

where variance (var) and expectation (E) operate across genotypes, f is the frequency of each genotype, and $\tilde{w}$ is the reproduction of each genotype relative to the population mean. If selection is frequency-independent, then w and f are independent, so that (A.2) can be rewritten as

$$\mathrm{var}(\tilde{w}f) = [\mathrm{E}(\tilde{w}^2) \cdot \mathrm{E}(f^2)] - \left(\mathrm{E}(\tilde{w})^2 \cdot \mathrm{E}(f)^2\right). \tag{A.3}$$

Because $\mathrm{E}(\tilde{w})$= 1 by definition, (A.3) simplified to

$$\mathrm{var}(\tilde{w}f) = \mathrm{E}(\tilde{w}^2) \cdot \mathrm{E}(f^2) - \mathrm{E}(f)^2. \tag{A.4}$$

Let $\Delta\mathrm{var}(f)$ represent the change in $\mathrm{var}(f)$ caused by one round of selection. Subtracting (A.1) from (A.4) gives

$$\Delta\mathrm{var}(f) = \mathrm{E}(f^2) \cdot [\mathrm{E}(\tilde{w}^2) - 1]. \tag{A.5}$$

Because $\mathrm{E}(\tilde{w})$= 1, the second term on the right, $\mathrm{E}(\tilde{w}^2) - 1 = \mathrm{E}(\tilde{w}^2) - [\mathrm{E}(\tilde{w})]^2 = \mathrm{var}(\tilde{w})$. Substituting $\mathrm{var}(\tilde{w})$ for $\mathrm{E}(\tilde{w}^2) - 1$ gives

$$\Delta\mathrm{var}(f) = \mathrm{E}(f^2) \cdot \mathrm{var}(\tilde{w}). \tag{A.6}$$

Thus the decrease in the variance of genotype frequencies is proportional to the variance in reproduction. Thus minimizing variance in reproduction also minimizes loss of diversity (H).

Acknowledgments

This work was supported by and carried out at the Santa Fe Institute. The author thanks H. Bagheri-Chaichian, L. Pagie, and C. Shalizi for helpful discussions, and John. H. Holland for comments on an earlier draft, as well as for suggesting the idea of variable-generation selection methods.

References

[1] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, UK, 1996.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[3] S. A. Frank, "George Price's contributions to evolutionary genetics," *Journal of Theoretical Biology*, vol. 175, no. 3, pp. 373–388, 1995.

[4] G. R. Price, “Selection and covariance,” *Nature*, vol. 227, pp. 520–521, 1970.

[5] J. E. R. Baker, “Reducing bias and inefficiency in the selection algorithm,” in *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications*, J. J. Grefenstette, et al., Ed., pp. 14–21, Erlbaum Associates, Hillsdale, NJ, USA, 1987.

[6] W. Fontana and P. Schuster, “Continuity in evolution: on the nature of transitions,” *Science*, vol. 280, no. 5368, pp. 1451–1455, 1998.

[7] J. H. Holland, “Building blocks, cohort genetic algorithms, and hyperplane-defined functions,” *Evolutionary Computation*, vol. 8, no. 4, pp. 373–391, 2000.

[8] J. W. Pepper, “The evolution of evolvability in genetic linkage patterns,” *BioSystems*, vol. 69, pp. 115–126, 2003.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 409045, 19 pages
doi:10.1155/2010/409045

Research Article

Adaptive Representations for Improving Evolvability, Parameter Control, and Parallelization of Gene Expression Programming

Nigel P. A. Browne and Marcus V. dos Santos

Department of Computer Science, Ryerson University, ON, Canada M5B 2K3

Correspondence should be addressed to Nigel P. A. Browne, nbrowne@acm.org

Received 15 September 2009; Revised 6 December 2009; Accepted 11 February 2010

Academic Editor: Oliver Kramer

Gene Expression Programming (GEP) is a genetic algorithm that evolves linear chromosomes encoding nonlinear (tree-like) structures. In the original GEP algorithm, the genome size is problem specific and is determined through trial and error. In this work, a method for adaptive control of the genome size is presented. The approach introduces mutation, transposition, and recombination operators that enable a population of heterogeneously structured chromosomes, something the original GEP algorithm does not support. This permits crossbreeding between normally incompatible individuals, speciation within a population, increases the evolvability of the representations, and enhances parallel GEP. To test our approach, an assortment of problems were used, including symbolic regression, classification, and parameter optimization. Our experimental results show that our approach provides a solution for the problem of self-adaptive control of the genome size of GEP's representation.

1. Introduction

Evolutionary computation (EC) is a machine learning technique that uses processes often inspired by biological mechanisms to obtain a solution to a given problem. Applying an EC algorithm to a problem begins by defining how potential solutions are represented, which is known as the *problem representation*. A problem representation is defined by the type of input data (the *Terminal Set*) used to generate a solution, the desired number, and types of outputs and the operations (the *Function Set*) used to transform the inputs into the output values. An important step in applying an EC methodology to a particular problem is the specification of parameters that define the problem representation and control the algorithm. Finding appropriate parameter values that yield satisfactory results usually requires carefully developed heuristics or expert knowledge. In EC algorithms, the concept of a *population* of candidate solutions, or individuals, is used to represent a pool of possible solutions to a particular problem. The encodings, or genomes, used to represent a solution vary depending on the EC methodology. It can be as simple as binary code, or as complex as a full fledged programming language. The Gene Expression Programming (GEP) algorithm [1], developed by Candida Ferreira, is an EC algorithm which uses separate encodings for the genotype and phenotype.

This work introduces novel enhancements to the Gene Expression Programming (GEP) algorithm that enable flexible genome representations, endow self-adaptive characteristics, increase the diversity within a population, and enhances the parallelization of the algorithm. The following issues are particularly relevant to the work presented here.

(1) *Evolvability*. The structure of the problem representation does not vary during a run, as it is restricted to the initial values for the head domain length and number of genes. This constrains the algorithm to narrow bands of exploration and reduces its ability to produce meaningful change or a paradigm shift within a population.

(2) *Crossbreeding and Speciation*. In GEP, genetic operations and transformation are restricted to identically structured genomes, preventing different species, or disparately structured genomes, from evolving and competing within a population.

(3) *Distributed Evolution*. Parallelization is restricted by the inability for disparate populations to interact, slowing the exploration of the search space.

(4) *Parameter Tuning and Self-Adaptation*. The GEP algorithm lacks a self-adaptation mechanism and thus requires additional time and resources to systematically evaluate different control parameter sets and subjecting the algorithm to operator biases.

To address the evolvability of the problem representation, we developed two new operators to permit the structure of the GEP genome to be changed during a run. We call these new operators the *Adaptive Chromosome Size (ACS) Mutation* operator and the *Head Insertion Sequence (HIS) Transposition* operator.

The problems of speciation and genome interactions between disparately structured individuals were solved by replacing the canonical GEP recombination operators with modified versions that permit dissimilarly structured individuals to interact.

From the beginning of our explorations we wanted to improve the performance of the GEP algorithm when distributed. We quickly realized that transferring individuals between separate GEP populations was severely limited by the inability for structurally different individuals to recombine. This issue was eliminated by the introduction of our modified recombination operators.

Finally, to enable parameter tuning in the GEP algorithm, we designed our HIS and ACS mutation operators to eliminate the two critical parameters of the GEP algorithm: the head size and the number of genes. Additionally, the HIS and ACS mutation operators were designed to permit the algorithm to self-adaptively tune the optimal chromosome structure.

Our proposed methodology was empirically evaluated using an assortment of problem classes and complexity levels. Symbolic regressions evaluated were kinematics problems, a series of polynomial regressions, and the "Sunspot Problem". The classification problem tested was the LiveDescribe dataset from the *The Center for Learning Technology* at *Ryerson University*. Finally, the effectiveness of the proposed methodology for optimizing parameters was evaluated using the De Jong test functions [2].

The effectiveness of the proposed changes were evaluated by comparing the performance of the enhanced GEP algorithm against the original GEP algorithm. Additionally, the symbolic regression results were compared to the adaptive distributed GEP algorithm developed by Park et al. [3]. The results obtained using an application developed during the course of this work, known as *Syrah*, and the results were validated using the K-Fold method with 10 folds.

The specific contributions of this work are as follows:

(1) development of the Head Insertion Sequence (HIS) operator to self-adaptively tune the head size parameter in the GEP algorithm and to enable the structure of the individual to evolve during a run,

(2) creation of the Adaptive Chromosome Size (ACS) Mutation operator that self-adaptively tunes the number of genes of an individual in a GEP population, Therefore allows the genome structure to evolve.

(3) addition of new recombination operators to the GEP algorithm to enable structurally dissimilar genomes to interact, therefore enabling individuals to be transferred between separate GEP populations without any genomic structural constraints. This feature is particularly important to parallel GEP systems, as it permits unrestricted migration.

Following this introduction, we present the background material related to this work in Section 2, our methodology in Section 3, the results and discussions of our experiments in Section 4, and finally, in Section 5, the conclusion and potential future work.

2. Background

In this section we present the relevant existing research that pertains to the key issues addressed by this work, including: the canonical Gene Expression Programming algorithm, the evolvability of the problem representation, genome crossbreeding and speciation, distributed evolution, parameter control and, self-adaptation.

2.1. Canonical GEP Algorithm.

2.1. Canonical GEP Algorithm. The Gene Expression Programming (GEP) algorithm was first published by Ferreira in 2001 [1]. Like other EC methodologies, GEP derives its inspiration from biological processes and has been successfully applied to a variety of problems [4–9].

A significant difference in GEP is the separation of the phenotype and genotype. Many existing methodologies, such as Genetic Programming [10] and Genetic Algorithms [11], use a single representation for both the genotype and the phenotype. By separating the representation, the GEP algorithm is able to benefit from the speed of operating on a linear genotype and the flexibility offered by the tree-based phenotype. It also permits the physical representation to affect the genetic code of the individual, as is found in nature.

In the GEP algorithm, each individual or candidate program is referred to as a chromosome. Every chromosome in the population represents a syntactically correct program, because of the underlying nature of the chromosome's encoding and representation.

2.1.1. Chromosome Encoding. In GEP the genome or *chromosome* consists of a linear, symbolic string of one or more *genes*, with each gene coding for an expression tree (ET). A gene has two well-defined, adjacent regions called *head*, containing symbols that code for internal or leaf nodes of the encoded ET, and *tail*, containing terminal symbols (the leaf nodes) of the encoded ET. In canonic GEP, both the number of genes and the head size of a gene are input parameters for the algorithm. The tail size t is a function of the head size h, and is determined as follows:

$$t = h(n_{\max} - 1) + 1, \tag{1}$$

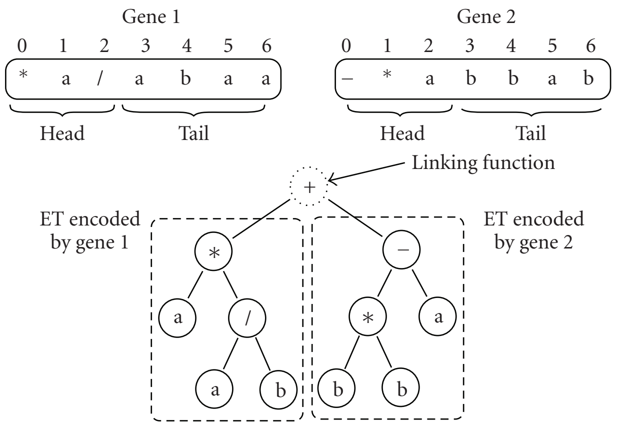


Figure 1: Chromosome with two genes: head size 3, tail size 4.

where $n_{\max}$ denotes the maximum arity found in the function set (Like in genetic programming, the function set is also a parameter to the GEP algorithm.).

In the case of multigenic chromosomes, all ETs are connected by their root node using a linking function. In the GEP system presented in this work we used the addition operator as the linking function. To illustrate, Figure 1 shows an example of a chromosome and the respective tree it encodes.

2.2. Evolvability. Evolvability refers to the ability of a genome to change over time and to occasionally produce offspring that are more effective at a particular problem (and thus perform an effective search) [12, 13]. For evolutionary computation, this becomes significant for representations, such as GEP, that separate the phenotype from the genotype. In the case of this work, we focus on the evolvability of the structure of the genotype (the encoding of the genome). This is particularly important in the case of GEP, since the genome structure of the canonical algorithm is fixed throughout a run and controlled by two problem-specific parameters.

Lopes and Weinert [14] proposed an enhanced GEP algorithm called *EGIPSYS* that varied the length of the head domain on a genome-level basis. The individuals, however, were composed of a fixed number of equal-length genes. This contrasts with the approach presented here, where each individual may have any number of genes and each gene may have a unique head length. Additionally, *EGIPSYS* neither implemented the one-point recombination operator nor introduced operators to vary a chromosome's length. It also restricted the operation of the gene recombination operator to like-sized individuals. All of these issues are resolved in the method presented here.

In an attempt to improve the evolvability of the individuals in GEP, Yue et al. [15] proposed a crossover strategy called *Valid Crossover Strategy* which would crossover all individuals in a population and create the subsequent population from the n-best valid chromosomes. This approach seemed to help the evolution of the solution, but not the evolution of the structure itself.

Several different strategies for improving the GEP algorithm were presented by Tang et al. in [16]. A feature of interest that they developed was an adaptive mutation mechanism, which was essentially a fitness proportional mutation rate. On an individual basis, the mutation rate applied to a chromosome was inversely proportional to its fitness. Thus, highly fit individuals would have a lower mutation rate applied to them, reducing the number of potentially disruptive changes to chromosome. Conversely, poorly fit individuals were more likely to have significant mutation performed on their chromosomes. The implementation of the Adaptive Chromosome Sizing Mutation Operator introduced in this work uses the idea of a fitness proportional mutation rate to preferentially mutate the number of genes in poorly fit individuals.

In this work we introduce new operators to improve the evolvability of GEP genomes. The new operators are the HIS transposition and ACS mutation operators, which allow the structure of a GEP genotype to change over time. The evolution of the genotype occurs in parallel, but fundamentally linked, to the exploration of the search space for a particular problem. The two evolutionary processes are interconnected because changes in the genotype can permit the algorithm to explore regions of the search space that may be inaccessible to other genome structures.

2.3. Crossbreeding and Speciation. The concept of crossbreeding and speciation embraced in this work is that of interactions between disparately structured, but fundamentally compatible, genomes. The idea of crossbreeding specifically refers to the ability for any individual, regardless of structure (or species), to reproduce and create viable offspring. The ability to crossbreed any individual permits a more genetically diverse population and enabled unrestricted exploration of the search space by the algorithm.

Speciation, on the other hand, can have several different interpretations. In particular, it can refer to the ability for "subpopulations" to exist within a single main population for the purpose of "niching" [17]. Speciation and niching has been used to promote diversity within a population, prevent (or limit) convergence, and address multimodal problems where different areas of the solution space require different individuals [13]. Two methods for using speciation, or niching, are *Crowding* [2] and *Fitness Sharing* [18].

The *EGIPSYS* algorithm [14] permitted different-sized chromosomes within a population, which other systems, such as canonical GEP, AdaGep [19], and PGEP-O [3], do not support. However, unlike our proposed methodology, all individuals in an *EGIPSYS* population were required to have the same number of genes. This contrasts with our proposed methodology, which supports (and, in fact, encourages) populations consisting of individuals that have both differing head domain lengths and gene counts.

Park et al. introduced a parallel system, PGEP-O [3], which attempted to dynamically tune specific parameters of the GEP algorithm. In the work, the individuals were constrained by the genome restrictions of canonical GEP; that is, only identically structured individuals were able to interact and exist within a single population or island. This methodology was limited because the transfer of individuals between islands, or migration, could only occur between islands with identical gene counts and head domain sizes. The methodology presented in this work eliminates these constraints by creating operators that do not restrict the interaction of genomes with fundamentally different structures.

The contributions presented in this work enable cross-breeding between disparately structured individuals in a GEP population, a feature unavailable in canonical GEP. This enables evolution of different species within a population, and while specifically implementing niching is beyond the scope of this work, it could be examined in the future.

2.4. Distributed Evolution. The intrinsic parallel nature of EC can often be further exploited by distributing a given EC algorithm. Parallelization techniques can generally be classified by their granularity, defined as either fine grained or coarse-grained models. Fine-grained techniques commonly have low computational requirements, but higher communication needs, and are well suited for multiprocessor systems. Coarse-grained models, on the other hand, tend to be computational intensive but have lower communication requirements and are better suited to discrete computational nodes. The *Island Model* is a coarse-grained technique that was introduced in [20] and has been shown to be fault tolerant [21]. The distributed system implemented to validate our methodology uses the Island Model.

The exchange of genetic material between islands, or *demes*, is referred to as migration. The structure of the connections between islands, or the topology, is bounded by the cases of isolated islands (no migration) and fully-connected (migration to all other demes) [22]. Additionally, dynamic topologies have been suggested [23]. In addition to the topology, the rate of migration, number of migrants, and the migration policy control the flow of individuals between islands [24]. One aspect of a migration policy is whether the migration occurs synchronously (migrations occur in specific intervals with specific partners) or asynchronously (migrations occurred whenever a deme has a migrant to exchange) [25]. Interested readers are directed to [23, 24, 26–30] for more detailed information regarding migration.

The PGEP-O system [3] is another example of a parallel GEP algorithm which used two island groups. The first island group was a standard Island Model implementation, in which a single population of individuals was evolved on each island. The second island group used the first group's island as their "individuals" in an attempt to use a GA to optimize the parameter settings of the island populations. Since the two island groups were needed, PGEP-O could only operate in a distribute mode. Additionally, since Park et al. did not address the interaction of differing genome structures, migration between the islands could only occur between like-structured populations. This limited the algorithm's ability to explore the search space.

Lin et al. proposed a fine-grained parallel GEP system [31] which exploited niching to improve the performance of the GEP algorithm. Based on their reported algorithm and data, they used a shared pool of like-structured individuals and empirically determined the GEP algorithm's parameter values.

A multiobjective parallel GEP system, *PGEP-AP* [32], also used the Island Model with migration. In addition to the standard migration mechanism PGEP-AP used a separate elitist population to store the best individuals from the various subpopulations.

The PED-GEP algorithm introduced in [33] used a measure of diversity to guide evolution among parallel clients; however, the details of their parallelization lacked further specifics.

Du et al., in [34], demonstrated a parallel GEP implementation that used Estimation of Distribution to improve the performance of the GEP algorithm. This system used asynchronous migration with a fully connected Island Model; that is, each island (population) could potentially interact with any other island.

Our approach to the distribution of the GEP algorithm was to use a fullyconnected coarse-grained model with random migration and to remove the restrictions placed on the migration mechanism by canonical GEP's inability to support dissimilarly structured chromosomes in a single population. By permitting unrestrained migration, populations in a parallel setting are now able to freely exchange candidate solutions to enhance the solution quality and diversity.

2.5. Parameter Control and Self-Adaptation. Most Evolutionary Computation algorithms require a set of control parameters, which influence the process evolution to be configured based on the particular problem being explored. The process of setting these parameters often requires complex heuristics, "rules of thumb", or specific knowledge from a domain expert. Thus, it is desirable to automatically tune the parameter values prior to executing the algorithm or to self-adaptively tune the parameters during the run.

In problem solving and optimization, the impossibility theory of "No Free Lunch" [35] has been postulated and roughly states that without *a priori* knowledge of a problem (to tailor the methodology to it) no single problem-solving method is inherently better for all problem classes [36]. This has implications for any evolutionary algorithm and parameter control method, especially those with attempt to optimize the parameters prior to executing an evolutionary run and then use static values throughout the run [37]. Additionally, it has been shown [38] that optimal parameter values can vary throughout a single run. This implies that, while it may be impossible to determine optimal values for all problems and situations, it should be possible to evolve values that are "good enough". Additionally, it implies that methodologies that are able to optimize their parameter values dynamically have an inherent advantage over those that do not.

The PGEP-O system presented in [3] approached the issue of parameter control as a separate optimization problem that ran in parallel to the main evolutionary algorithm.

This system used a parallel GEP implementation, using the Island Model, to evolve solutions to the target problem and a genetic algorithm (GA) running on a separate client to optimize the two GEP parameters. The head size and gene count parameters were optimized by using trial values on each GEP island and then reporting back to the GA parameter optimizer. This approach, while successful, suffered from several issues that are remedied by our proposed methodology. The PGEP-O algorithm required additional resources, since the parameter optimization was a separate calculation. Additionally, the GA optimizer had to wait for an entire run to complete before it was able to execute a new generation, which is problematic for long-running evolutions.

The *DM-GEP* algorithm [33] introduced a dynamic mutation rate operator in an attempt guide evolution. *DM-GEP* divided the execution of a run into three stages, the initial stage; the metaphase stage, and the anaphase stage. Each stage was then assigned a specific mutation rate and the mutation rate used in each generation was progressively scaled, by a fixed amount, from one value to the next. In this manner, the number of generations executed in a run was directly related to the mutation rates. This approach did not, strictly speaking, tune the mutation parameter and was not self-adaptive, but did dynamically alter the rate and showed improvement over the standard GEP implementation.

Bautu et al. introduced in [19] an algorithm, called AdaGEP, for automatically controlling the number of genes of a GEP representation. The approach involved adding to the genome a bit array that maps each bit to a gene in the chromosome. The bit in each position of the array indicates if whether gene would be included in the translation to an expression tree during the fitness evaluation. Specific genetic operators were designed to operate on this bit array, thus evolving an optimal mask. The AdaGEP algorithm was limited by the fact that the total number of genes in any chromosome could never change. Thus, there was little benefit to using that method versus using automatically defined functions, or homeotic genes, in GEP's jargon, to evolve the execution order of the genes. Additionally, the size of individuals in the algorithm's population could never change, so that, even if fewer genes were required, the genetic operators would still be performed on the full chromosome.

The work presented in [15] included a method to vary the mutation and crossover rates during a run, based on the *Cloud Model* [39]. This methodology improved the performance of the GEP algorithm, but was only applied to like-structured genomes.

Eiben et al. stated in their "Parameter Control in Evolutionary Algorithms" survey [37] that determining successful values for algorithm parameters in EC is a "grand challenge" problem.

The approach to parameter control and self-adaptation presented in this work was accomplished using multiple techniques which work together to self-adaptively tune GEP parameters. To tune the head domain length and number of genes, we developed the HIS transposition and ACS mutation operators. In addition to these operators, we created new recombination operators which allowed structurally disparate (and normally incompatible) genomes to be able to crossbreed and create viable offspring, which permits individuals with different head domain length and gene count parameters to compete within a single population.

3. Methodology

This section introduces the proposed enhancements to the GEP algorithm to address the issues identified in Section 1, to wit the evolvability of the problem representation, genome speciation and crossbreeding, distributed evolution, and parameter control and self-adaptation in the canonical GEP algorithm. The section will introduce our proposed enhancements, the details of the implementation of the framework used for evaluation, and the experiments used to validate our hypothesis.

3.1. Proposed GEP Algorithm Enhancements.

3.1. Proposed GEP Algorithm Enhancements. To address the issues of evolvability, crossbreeding, distributed evolution, and parameter control found in canonical GEP, our proposed modifications to GEP include several new operators and also modifications to the existing recombination operators. The new operators introduced in the following section offer solutions to the problems of evolvability and the control of two critical parameters in GEP. The modified recombination operators were developed to permit speciation within a GEP population and to enhance distributed GEP populations.

The original version of the GEP algorithm required that two critical parameters, the length of the head domain and the number of genes in the chromosome, to be set to fixed values prior to the execution of a run. These parameters are generally domain and problem specific, which further exacerbates the problem of finding "good" values (not even particularly optimal ones) for the parameters. By developing new operators which permit genome structure changes, we enabled the head domain length and number of genes to be implicitly tuned during a run. Our algorithm enhancements also permit each gene in a chromosome to have a unique head domain length. This extra feature enables the length of the gene to vary, and thus the length of the function encoded by that gene.

In addition to parameter control, our approach improves the evolvability, or the ability of the structure of the genome to evolve, by removing the fixed-length chromosome restrictions in canonical GEP and allowing the number of genes to vary during a run. Chromosome evolvability was specifically addressed by designing the new operators to increase the capacity of the genome for extracting and exploiting the underlying structure of the fitness function under consideration.

The new operators for parameter control and enhancing evolvability are presented in Sections 3.1.1 and 3.1.2.

3.1.1. Adaptive Chromosome Size Mutation Operator. In Algorithm 1 we present the pseudocode for the Adaptive Chromosome Size (ACS) mutation operator used in our enhanced GEP algorithm. The ACS operator mutates the number of genes in a chromosome, potentially increasing or

```
Data: Chromosome
Result: Mutated chromosome
begin
    /*Calculate the decay rate                    */
    decayRate = 1−(gen+maxGen∗factor)/maxGen

    /*Calculate the mutation rate,
      inverse to the fitness                      */
    muP = (1-chr.Ftn/bestFtn)
    /*Adjust the mutation rate if it is
      below the minimum                           */
    if $muP < minRate$ then
        muP = minRate
    end
    /*Apply the decay to the mutation
      rate                                        */
    muP = muP ∗ decayRate
    /*Determine if mutation will occur
      */
    if $RandProbability() \leq$ to $muP$ then
        /* Randomly decide to grow or
           shrink                                 */
        growChromosome = DoCoinToss()
        if growChromosome then
            /*Grow the chromosome by
              adding a new gene                   */
            insertionPoint = GetRnd(0, chr.NGenes)
            InsertGeneAt(insertionPoint)
        else
            /*Shrink the chromosome by
              deleting a gene, but only
              if we have at least two
              genes                               */
            if $chr.NGene > 1$ then
                deletionGene = GetRnd(1,
                chr.NGenes)
                DeleteGeneAt(deletionGene)
            end
        end
    end
end
```

Algorithm 1: ACS mutation operator pseudocode.

decreasing the total number of genes when it is applied. The ACS operator is applied to the entire population during each generation.

The *AcsGeneMutation*$(\cdots)$ method takes a chromosome (*chr*) as a parameter and mutates it according to the following procedure. Initially, it calculates the *decayRate*, which is used to decrease the operator's application as the run progresses. In the *decayRate* calculation the *factor* is a user-defined value that scales the *decayRate* and is set to 0.2 for all experiments. This scales the *decayRate* to zero for the final 20 percent of the run.

Next, the algorithm calculates the probability of mutation *muP*. The probability of mutation is inversely proportional to the individuals fitness when compared to the best fitness in the current generation. If the *muP* is less than the user-defined minimum mutation rate, *minRate*, then *muP* is set equal to *minRate*. The mutation rate, *muP*, is then scaled by *decayRate* to arrive at the final *muP* value. The operator then generates a random probability using *RandProbability* () and compares it to *muP* to determine whether the *AcsGeneMutation* will be applied to the chromosome. Next, the operator performs a coin toss using *DoCoinToss* () to determine whether a gene should be added or removed. When a gene is added, the operator selects an insertion point, *insertionPoint*, at a random position in the sequence of genes of the chromosome.

It then calls the worker method, *InsertGeneAt*$(\cdots)$, to insert a randomly created gene at the insertion point. When a gene is removed, the operator first verifies that there is more than one gene (*chr.NGenes*) in the chromosome. It then randomly selects a gene in the chromosome using the *GetRnd*$(\cdots)$ method and calls the *DeleteGeneAt*$(\cdots)$ method to remove the gene from the chromosome.

The mutation operator always uses a step size equal to one. Thus, it modifies a single gene in the chromosome during each application of the operator. Alternative step sizes were not investigated, but will be examined in future work.

3.1.2. HIS Transposition Operator. To dynamically tune the size of a gene, we introduced a new transposition operator called *head insertion sequence transposition*, HIS transposition, for short. The transposable elements (also called *transposons*) in this case are fragments of the genome, located in the head of a gene, that can be activated and jump to (possibly) another gene head in the chromosome. Two features make this operator different from the canonic transposition operators used in GEP, to wit that

(i) the transposable element is necessarily located in the head of a gene,

(ii) during transposition the transposon is cut from the place of origin (instead of copied, like in canonic transposition in GEP), thus *shortening* the length of the respective gene, and then inserted in the place of destination located necessarily in the head of (possibly) another gene, thus *elongating* the gene length at the target site.

Specifically, the HIS transposition operator works as follows. Initially the operator randomly chooses the chromosome, the start and end sites of the transposon, and the target site. As mentioned above, these start and end sites are located in the head of a gene. Moreover, transposons contain at most three elements. Next, the operator cuts the transposon from the site of origin, making the necessary arrangements to maintain the structural integrity of the gene. That is, if the transposon locates in the middle of the head of a gene, then the left and right remaining segments of the head are concatenated, thus forming the new gene head. Next, the operator inserts the transposon at the target site, thus elongating the head of the gene. Notice that the gene heads at the place of origin and at the target site have now changed; the latter is now longer by, say, k elements, and the former is k elements shorter. Finally, using (1), the operator adjusts the respective new tail sizes of those genes. If the tail requires extra material, it is taken from the remaining genetic material in the source gene's tail.

3.1.3. Recombination Operators for Nonuniform Chromosomes. The notion of *species* is not present in canonic GEP, as all chromosomes have the same structure; that is, all individuals in a population have the same gene head size, same gene tail size, and the same number of genes. The possibility of different species within a single GEP population is highly desirable feature for the parallelization of the algorithm, particularly when using a migration mechanism in a distributed setting. By modifying the existing GEP recombination operators to handle genomes with different structures, our enhanced GEP algorithm now supports crossbreeding and speciation within both a single population and distributed islands.

To support different-sized chromosomes created by ACS mutation and HIS transposition operations, we created modified versions for the one-point and two-point recombination operators used in GEP. These operators also facilitate integrating individuals with differing genome structures (i.e., a differing number of genes and head domain lengths) into a target population during migration, when distributed. Recombination via these operators works as follows: initially the first positions for the head and tail sections of the two-parent chromosomes are paired (see Figure 2). Then the crossover point (or points, in the case of two-point recombination) is randomly chosen from the overlapping sections of the chromosome. Then the crossover point locates either in the head of a gene or in the tail. If it falls in the *head*, then the genetic material is exchanged (the strands swapped) at the crossover point (see Figure 2(a)). For this case, there is no need to adjust the structure (tail size) of the gene containing the crossover point. If it locates in the *tail* of a gene, then we use the following process to exchange the genetic material of the genes where the crossover point is located. First we exchange the genetic material at the point of crossover. Then, we verify that the tail sizes of the resulting genes comply with the respective resulting head sizes. If the tail size of a recombined gene is s elements shorter than the allowed size, then we append to it s elements from the tail of the other parent gene, thus making the final tail size of the recombined gene compliant with its head size (notice the strand added to O_1 in Figure 2(b)). On the other hand, if the tail size of the recombined gene is s symbols longer than the allowed size, then we cut its s last symbols out (notice the strand removed from O_2 in Figure 2(b)).

The rest of genetic material is exchanged as in normal crossover, with a caveat: for the case of GEP-RNC (GEP with real number constants [7]), if the crossover point locates in the tail of a gene, the genetic material in the domain of constants (Dc) is exchanged as normal and the lengths of the Dc domains are adjusted. If the crossover point falls in the Dc domain, then recombination proceeds via the same procedure used for the tails, as illustrated in Figure 2(b). The arrays containing the gene's real number constants are exchanged in their entirety [40].

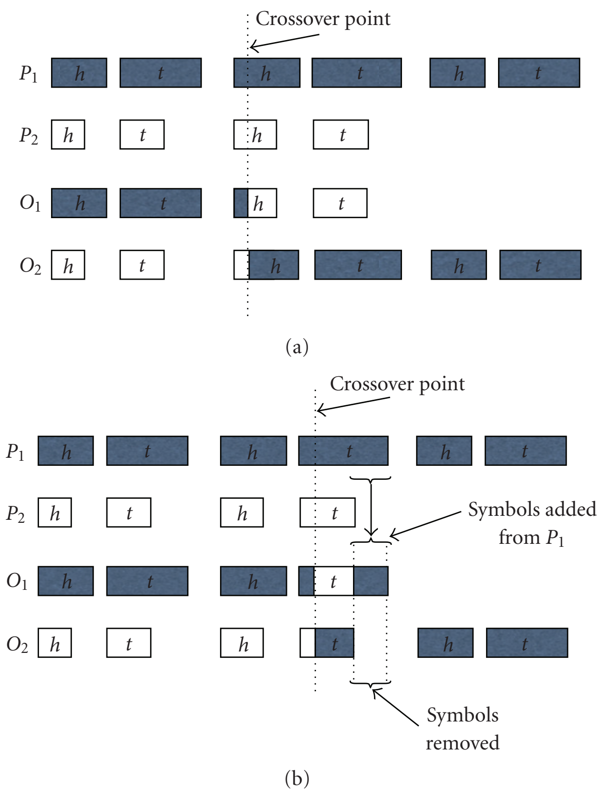


Figure 2: One-point recombination of two chromosomes, P_1 and P_2, containing 3 and 2 genes, respectively; h and t denote the head and tail portions of each gene, respectively. In Figure 2(a) the crossover point locates in the head of a gene. In Figure 2(b) the crossover point locates in the tail of a gene.

Analogous to GEP, our recombination operators also produce two children from the parents, with one child having the same length as one of the parents, and the other child having the same length as that of the other parent.

3.2. Syrah Implementation. In this study, a parallel capable GEP system called *Syrah*, which dynamically tunes the number of genes and gene size, was developed. To test this system, a suite of nontrivial symbolic regressions was used and the quality of the models was benchmarked against models obtained via a canonic GEP system and competing methodologies.

Syrah's system requirements differ from GEP-RNC (GEP with real number constants [7]) in regards to the genetic operators it uses, which are detailed in Sections 3.1.1, 3.1.2, and 3.1.3.

In *Syrah*'s implementation, tournament selection with elitism was used. Many GEP implementations use Roulette Wheel selection, but as long as elitism is used, various selection methods will produce equally good results [7].

When the *Syrah* system is operating in parallel, it uses a coarse-grained model (or Island Model [20]) to distribute the populations. *Syrah* uses the proposed genetic operators to permit disparate genome structures to be integrated into a given population during a migration event.

3.2.1. Development and Runtime Environments. All components of the research system *Syrah* were written in C# using the Microsoft.Net Framework version 3.5 and developed using Microsoft Visual Studio 2008. Data storage and management was accomplished using Microsoft SQL Server 2005 running on *Microsoft Windows XP Professional*. The client computers also used the *Microsoft Windows XP Professional* operating system.

3.2.2. Parallelization. Different methods and techniques exist for operating an EC algorithm in parallel. Generally parallel techniques can be divided into two categories: fine grained and coarse grained [24]. Fine-grained techniques involve parallelizing the evaluation of the test cases and usually have more intensive communication requirements. Alternatively, coarse-grained techniques distribute populations and have lower communication requirements, but higher computational needs. Our experimental system uses a common coarse-grained technique known as the *Island Model* [20] to distribute populations to discrete computational nodes. The Island Model implemented in *Syrah* is a fullyconnected topology that supports random-random migrations, meaning that a migration event can (randomly) involve any node in the system. Details regarding migration can be found in [22–24, 26, 27, 29, 41].

The network communication between nodes was implemented using the HTTP v1.1 protocol over an SSL connection. The server node is designed to listen for client requests on port 443, the standard port used by SSL web servers. Additionally, the communication between the client and the server is always initiated by the client. This combination of techniques was selected so that the communication would be relatively secure and to facilitate communication between the client and server, when the client was located behind a firewall. This was done to circumvent firewall issues in the original network used for testing.

Finally, based on [21], the nodes do not implement any special handling for detecting and preventing network topology faults. When a client is unable to complete a run (i.e., the host was restarted, network failure, etc.), the client is simply to starts a new run when it rejoins the *Syrah* topology.

3.2.3. Population Initialization. With the use of our recombination operators, the population is able to support individuals with different chromosome sizes. To take advantage of this feature, the population is seeded with randomly sized chromosomes. Both the number of genes and the head domain length of each gene are varied during this phase. The number of genes in each individual is randomly selected between 1 and 10. During the creation of the chromosome, each gene selects a random head domain length between 5 and 15. These values were empirically determined during initial testing and were found to provide good genetic diversity. Additionally, we selected the random initialization method over a "ramped half-and-half" method [10] as a result of early experimentation.

The elements of the head are selected from a weighted bag. If the function set is smaller than that of the terminals, then the probability of selecting a function is 1/2; otherwise they are equally weighted.

3.3. Experimental Design. An assortment of problems, of varying types and difficulty, were selected to evaluate the performance of our approach. The problems were selected from three areas to which Evolutionary Computation is commonly applied:

(i) symbolic regression, or the automatic synthesis of functions,

(ii) classification, or generating boolean results (or labels) from a set of input values,

(iii) parameter optimization, or the automatic discovery of parameter values which produce a maximum and/or minimum for a given function.

3.3.1. Validation of Results. Each experiment was performed using K-Fold validation with 10 folds and 30 runs per fold. Each experiment consisted of two sets: a baseline set and an adaptive set. The baseline runs were executed using the standard GEP-RNC algorithm implemented as a part of the *Syrah* system with parameter control disabled. The adaptive runs were then executed in the same manner, but using the methodologies outlined previously.

Each experiment was executed using the *Syrah* framework's parallel mode, which uses the Island Model to distribute the populations to separate computational nodes. The experiments used 32 islands that were executed on 16 dual-core Intel computers, running the Windows XP Professional operating system. The *Syrah* system supports migration between the islands But to facilitate the statistical analysis of the results, these experiments were run without this feature.

The baseline experiments were performed repeatedly using the values presented in Table 1. During the adaptive evolution runs, the number of genes and the size of the head domain were tuned using our new operators. The details of the initial chromosome lengths can be found in Section 3.2.3.

3.3.2. Symbolic Regression Experiments. The first three problems selected were the same problems used by Park et al. in [3]. These were selected so that the performance of this methodology could be compared to an existing (parallel) GEP-based self-adaptive approach. The fourth experiment was a regression of a sawtooth wave, while the fifth experiment was a more difficult time series analysis problem. The baseline experiments all produced poor results for gene counts of 1 through 3, which required 900 (3×10 folds $\times$ 30 runs per fold) runs to determine.

Experiment 1. The first problem evaluated was a kinematics symbolic regression that modeled the movement of a vertically fired object. The kinematic equation for the position of the object at time t is defined by the following equation:

$$S(t) = S_0 + V_0 t + \frac{at^2}{2}. \quad (2)$$

Table 1: Common experiment run parameters.

Parameter	Problem	
	Symbolic Regression & Parameter Optimization	Classification
Selection method	Elitist Tournament	Elitist Tournament
Number of generations	100	175
Population size	100	75
Initial head size	5–15	5–15
Initial number of genes	1–10	1–10
One point recombination rate	0.5	0.5
Two point recombination rate	0.1	0.1
Gene recombination rate	0.1	0.1
Mutation rate	0.07	0.07
Minimum ACS mutation rate	0.05	0.05
IS transposition rate	0.1	0.1
RIS transposition rate	0.1	0.1
HIS transposition rate	0.1	0.1
Gene transposition rate	0.1	0.1
Function set	F_1	F_2
Linking function	+	+
K-Fold validation	10 folds	10 folds
Evolutionary Clients (*Syrah*)	31	31

$F_1 = \{+, -, *, /\}$, $F_2 = \{+, -, *, /, \text{sqrt}, \exp, \sin, \cos, \tan, \text{floor}, \text{ceiling}, \text{OR}, \text{AND}, <, >, \leq, \geq, ==, !=\}$.

If we use an initial velocity, $V_0 = 25\,\text{m/s}$, and an initial position of $S_0 = 0$ and assume that the acceleration is equal to earth's gravity, $a = -9.8\,\text{m/s}^2$, then we can simplify the equation as

$$S(t) = 25t + \frac{-9.8t^2}{2} = 25t - 4.9t^2. \tag{3}$$

For this experiment, fifty data points were sampled from the interval $t = 0.1$ to $t = 5$ and used as the test cases.

Experiment 2. Our second experiment extended the first, using two independent variables instead of one. Modifying (2) with the same assumptions as in Experiment 1, but with an independent initial velocity, gives:

$$S(t) = vt - 4.9t^2. \tag{4}$$

The test cases for this experiment were generated using V_0 values of 20, 25, and 30. The values of t were the same as in the first experiment.

Experiment 3. The third symbolic regression experiment used a fourth-order polynomial that was used in [3] and similar to the ones used in [1, 7]:

$$y = -2.5x^4 + 4.6x^3 + 3x^2 + 2x + 1. \tag{5}$$

The algorithm attempted to evolve the function from 10 equally spaced samples taken from values of the Polynomial (5), in the interval $x = [1, 10]$.

Experiment 4. The fourth experiment was a regression of a sawtooth wave, which has been used as a benchmark in other works [42]. The function is defined by

$$F(x) = \sum_{i=0}^{n} \left(\frac{1}{i} \sin(i \;\; x) \right) : \quad n = 1, \ldots, 9. \tag{6}$$

The dataset consisted of 250 equally spaced data points in the range $x = [-8, 8]$. This range was selected instead of the 40 points in $[-1, 1]$ used in [42] after discovering that the algorithm required a more challenging set of inputs.

Experiment 5 (Wolfer Sunspot Time Series Prediction). The final experiment attempted to create a predictive model using 100 observations from the well-known Wolfer Sunspot Series [43]. The data were formatted for time series analysis, using a delay time of 1 and an embedding dimension of 10. This dataset has also been used to evaluate other GEP systems, including those in [7] and [14].

3.3.3. Classification Experiment.

3.3.3. Classification Experiment. Classification is a common and important task for evolutionary computation algorithms. The classification experiment performed in this work used a large, real-world classification problem from the *The Centre for Learning Technology* (CLT) at *Ryerson University*.

The evaluation of the classification experiments was accomplished using the "Hits with Penalty" method, as described in [7].

The LiveDescribe project [44] is a software application developed by the *Center for Learning Technology* (CLT) at *Ryerson University* to added video descriptions (for the deaf) to video content. The project had originally used a manual process to select regions of dialog versus nondialog, so that descriptive video captions could be programmatically added to the non-dialog sections. Since the process of selecting the nondialog regions was a manual and user-intensive process, the CLT modified their application using a human-designed classifier system. This system was, on average, 70% effective.

The dataset consists of six real-value inputs and a single boolean output. Part of what makes this dataset a challenge is its size. The initial dataset consisted of approximately 90,000 records. The input variables are audio metrics and include RMS standard deviation, RMS average, a measure of audio entropy, zero crossing above to below, zero crossing left skew, and a zero crossing low-energy measurement. These inputs were sampled once for every 1 second of audio.

3.3.4. Parameter Optimization Experiments. The five parameter optimization test functions were selected from the the well-known De Jong test functions [2]. These test functions were originally selected by De Jong to test the effectiveness of a given EC algorithm over a broad class of problems. While attempts have been made to improve the test set, it remains the *de facto* standard for parameter optimization validation. The five functions are presented here in their original form, but were modified (where necessary) to change them all to maximization functions, which allows for simpler evaluation with the GEP algorithm.

De Jong F1: Sphere Model. The first function in the De Jong test set is a three-dimensional parabola that is convex, unimodal, and continuous. The function has a maximum of 78.6 at $(x_1, x_2, x_3) = (\pm 5.12, \pm 5.12, \pm 5.12)$:

$$f(x) = \sum_{i=1}^{3} x_i^2 : -5.12 \le x \le 5.12. \quad (7)$$

De Jong F2: Rosenbrock's Function. The second function in the De Jong test set was first proposed by Rosenbrock [45] and is commonly referenced in optimization literature. This function is nonconvex, unimodal, and continuous, with a maximum of 3905.93 at $(x_1, x_2) = (-2.048, -2.048)$:

$$f(x) = 100 \times \left(x_1^2 - x_2\right)^2 + (1 - x_1)^2 : -2.048 \le x \le 2.048 \quad (8)$$

De Jong F3: Step Function. The third De Jong test function is a five-dimension step function that is discontinuous, non-convex, unimodal and, piecewise constant. De Jong original selected this function to test the ability for algorithms to handle discontinuities [2]. This function is restricted to $-5.12 \le x \le 5.12$ for testing. This function has a known maximum of 25 when the inputs are held at 5.12:

$$f(x) = \sum_{i=1}^{5} x_i : -5.12 \le x \le 5.12. \quad (9)$$

De Jong F4: Quadratic Function with Noise. The fourth test function in the De Jong collection is a noisy quadratic function that is continuous, unimodal, convex, and haing a high dimensionality. The function uses a Gaussian function to add noise. The function was limited to $-1.28 \le x \le 1.28$. This experiment used alternative values for the number of generations and the population size of 350 generations and 500 individuals in the population.

Table 2: Summary of symbolic regression experimental results.

Exper.	Ours		Comparison	
Number	Length	Fitness	Length	Fitness
1[1]	254	99.984%	266	99.496%
2[1]	87	99.983%	282	99.907%
3[1]	155	99.735%	470	96.187%
4[2]	62	99.987%	185	99.966%
5[2]	55	99.179%	186	98.936%

1: Compared to PGEP-O.
2: Compared to canonical distributed GEP.

The maximum of this function is approximately 1248.2 and occurs when all inputs are equal to ± 1.28:

$$f(x) = \sum_{i=1}^{30} i \times x_i^4 + \text{Gauss}(0,1) : -1.28 \le x \le 1.28. \quad (10)$$

De Jong F5: Shekel's Foxholes. This is a two-dimension function that is continuous, nonquadratic, and non-convex, with 25 local maximums and was originally suggested by Shekel [46]. This version [47] of the function has maximum of approximately 499.002:

$$f(x, y) = 500 - \frac{1}{0.002 + \sum_{j=0}^{24} 1/\left[1 + i + (x - a(i))^6 + (y - b(i))^6\right]}, \quad (11)$$

where

$$a(i) = 16 \times (i \bmod 5 - 2),$$
$$b(i) = 16 \times \left(\left\lfloor \frac{i}{5} \right\rfloor - 2\right) - 65.523 \le x \le 65.523. \quad (12)$$

4. Results and Discussion

This section presents the results of the experiments outlined in Section 3 that were used to validate our enhancements to the GEP algorithm that address the issues identified in Section 1.

4.1. Symbolic Regression Results. Table 2 shows a summary of the experiment results, including the best individual's fitness and chromosome size (Note that the size of a chromosome (i.e., the length of the chromosome string) depends on its number of genes and the head size of each gene). The best fitness is expressed as a percentage of the number of fitness cases solved. The visualized results and performance of the experiments are shown by Figures 3, 2, 4, 5, 6, 7, 8, 9, 10, 11 and 12.

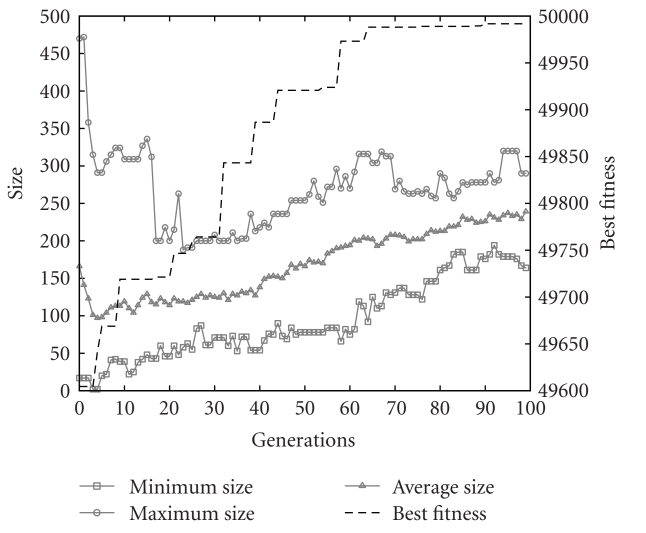


Figure 3: Symbolic regression Experiment 1: chromosome sizes and best fitness.

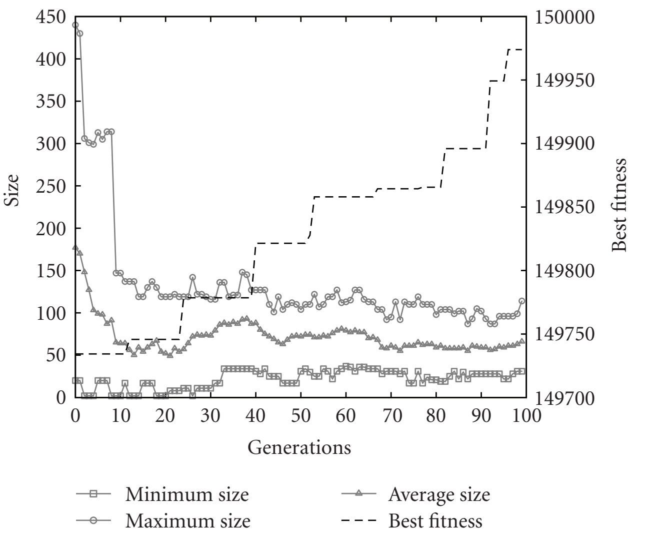


Figure 4: Symbolic regression Experiment 2: chromosome sizes and best fitness.

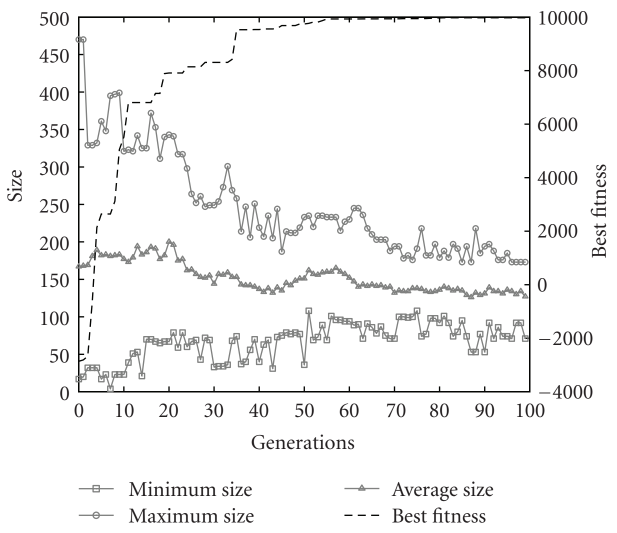


Figure 5: Symbolic regression Experiment 3: chromosome sizes.

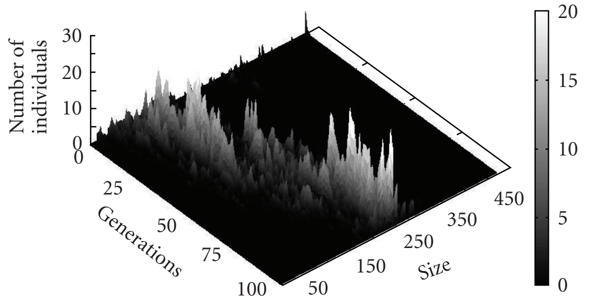


Figure 6: Symbolic regression Experiment 1: chromosome size in the population.

4.1.1. Discussion of Symbolic Regression Experiments. There are two figures for the first four experiments performed. The first figure of each pair shows the minimum, maximum, and average chromosome lengths in the population for each generation with respect to the generation number in the run. The other figures display a surface visualization of the distribution of the chromosome lengths in the population, with respect to the generation number in the run. For the final experiment, the surface plot was omitted because of the rapid convergence to a narrow range of chromosome lengths. Figure 11 compares the evolved model's performance to the target data. Since K-Fold validation was used, every tenth data point in Figure 11 was previously unseen by the model.

The figures show that while the algorithm was optimizing the chromosome length it would initially explore a wide search space, then focus on a band of neighboring chromosome sizes.

A significant result was that the best solutions found using our new operators evolved better individuals with smaller representations than the PGEP-O system presented in [3] and the canonical GEP algorithm. It is interesting to note that the best chromosomes evolved for the two most difficult problems were significantly smaller than those evolved by the PGEP-O. Specifically, during the second and third experiments, the best evolved individuals were approximately 30% to 33% of the size of the individuals evolved using the PGEP-O methodology. Similarly, in Experiments 4 and 5, where our methodology was compared to a distributed canonical GEP algorithm (based on *Syrah*), our methodology produced results 33% and 30% the size of the alternative's results.

The results of the experiments, as shown in Table 2, show that our new operators are significantly more efficient and produced better results for symbolic regression problems. This may have been because our populations were evolving smaller solutions and were able to explore the search space more effectively.

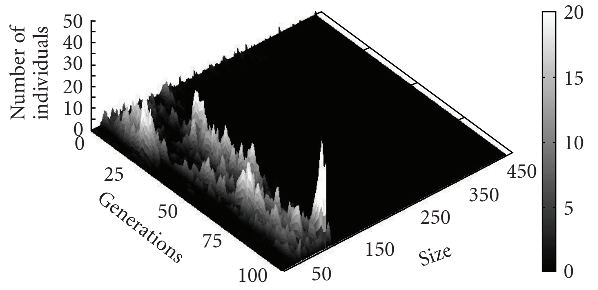


FIGURE 7: Symbolic regression Experiment 2: chromosome size in the population.

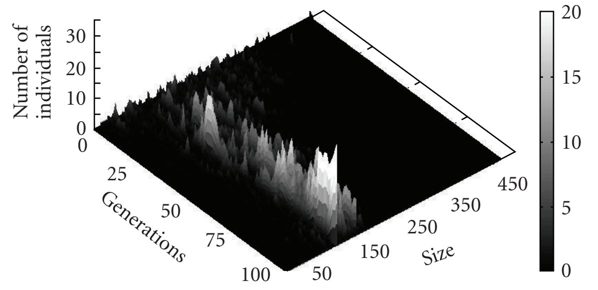


FIGURE 8: Symbolic regression Experiment 3: chromosome size in the population.

4.2. *Classification Results.* Table 3 shows the results of the classification experiments. These include the chromosome size and the best fitness found, expressed as a percentage of the number of fitness cases solved. The visualized results and performance of the experiments are shown by Figures 13 and 14.

4.2.1. *Discussion of Classification Experiments.* As stated in Section 3, the full LiveDescribe dataset consisted of approximately 90,000 entries, each with 6 real number variables and grouped into two classes. One of the challenges of this experiment was the computational resources required to evolve candidate solutions.

The two methods both evolved individuals with similar performance, with both systems evolving a classifier capable of successfully identifying 80%-81% of the fitness cases. This is a substantial improvement over the original, human-written classifier (developed by the CLT at Ryerson [44]), which was able to correctly classify approximately 70% of the fitness cases. After discussions with the CLT lab, it is believed that 85% may be the practical limit for identifying non-dialog sections of video using the current variable set. The CLT is currently working to modify its data acquisition software to collect additional parameters.

Examining the solutions evolved by our enhanced algorithm and canonical GEP, it is important to note that our methodology evolved a solution 32.1% the size of the one evolved by the standard algorithm. Since the size of the candidate solution's genome has a direct impact on the evaluation of the fitness cases (and live data, once implemented in the real world), the reduction in representation size may

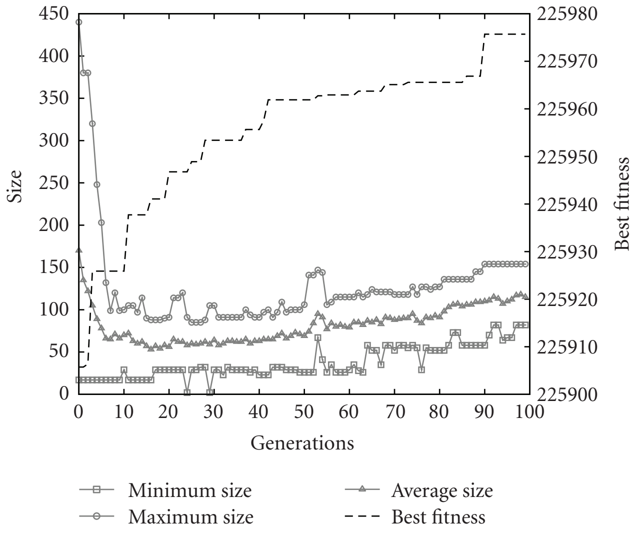


FIGURE 9: Symbolic regression Experiment 4: chromosome sizes and best fitness.

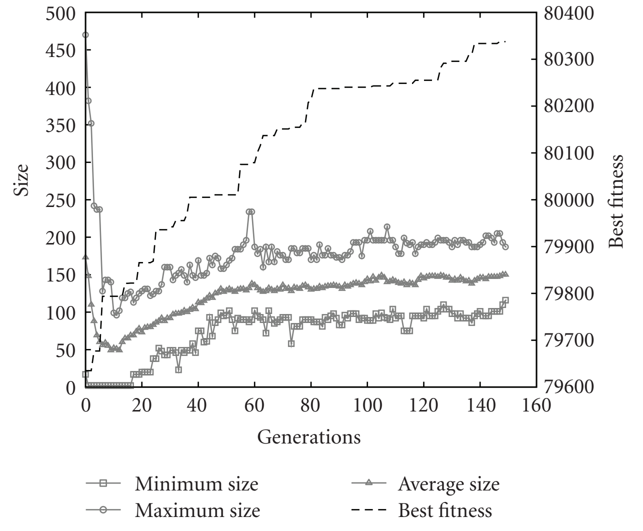


FIGURE 10: Symbolic regression Experiment 5: chromosome sizes and best fitness.

improve the overall performance of the system, even after considering the additional computation requirements of our new operators.

The small number of classes in this experiment may have been possible limitation. With only two possible classes, the evolutionary process may not have been significantly challenged. However, it is felt that the number of test cases may have offset this. In the future, more complex classification problems should be investigated.

What the summary of results do not show is the number of additional epochs (and thus processing time) required to evaluate different values for the head domain length and number of genes for the canonical GEP algorithm that was used for comparison.

Table 3: Summary of classification experimental results.

Ours				Canonical GEP			
Total Len.	Genes	Avg. Gene Len.	Fitness	Total Len.	Genes	Gene Len.	Fitness
247	8	30.9	81.08%	770	10	77	80.31%

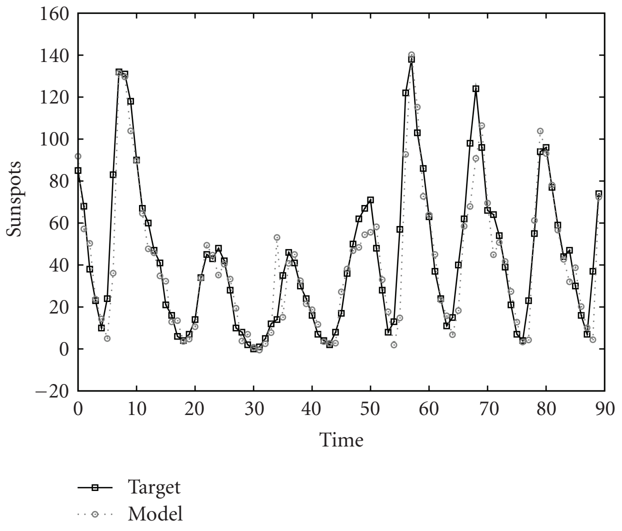


Figure 11: Symbolic regression Experiment 5: Target versus Model.

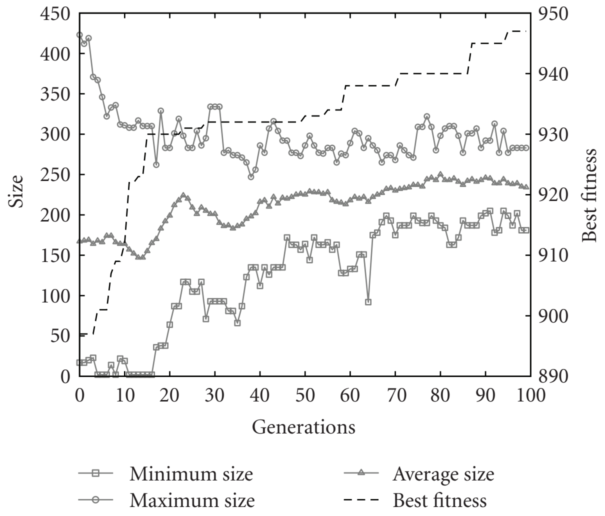


Figure 13: LiveDescribe experiment: chromosome sizes and best fitness.

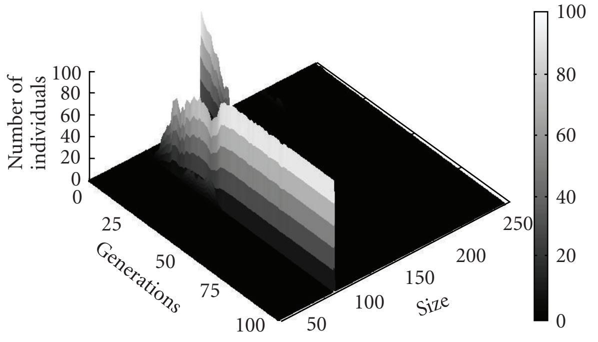


Figure 12: Symbolic regression Experiment 4: chromosome size in the population.

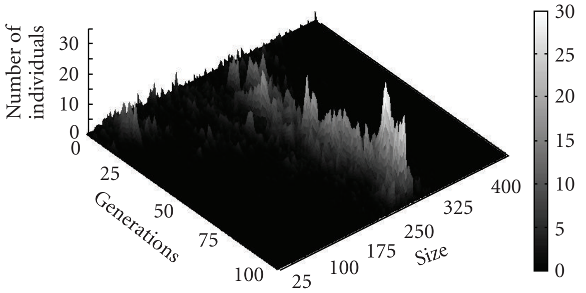


Figure 14: LiveDescribe experiment: chromosome size in the population.

4.3. Parameter Optimization Results. Table 4 shows a summary of the results of parameter optimization experiments. Included in the summary are the maximum value found, the number of genes used by the best solution, the average gene length (static for canonical GEP), and the total genome size. The visualized results and performance of the experiments are shown by Figures 15, 16, 17, 18, 19, 20, 21, 22, 23 and 24.

4.3.1. Discussion of Parameter Optimization Experiments. The results of the parameter optimization experiments show that both our methodology and canonical GEP are effective at evolving either optimal or near-optimal solutions to the problems in the De Jong test suite. As seen in previous experiment series, our enhancements enabled the algorithm to consistently evolve solutions which were significantly smaller than those evolved by canonical GEP.

The solutions evolved by our enhanced algorithm in Experiments 2 and 3 were remarkably smaller than those found by canonical GEP. Specifically, they were 5.4% and 6.5% the size of those found by standard GEP.

Both methodologies had difficulty with the high-dimension problem found in parameter optimization Experiment 4. However, our enhanced GEP algorithm evolved a slightly better result and had a representation size 54.6% the size of the one evolved by the standard algorithm. It is believed that the difficultly of this problem and the inability of the algorithm to locate the optimal parameter values contributed to the evolved size of the genome. Similarly, the numerical results of Experiment 1 were comparable, but the solutions evolved using the HIS

Table 4: Summary of parameter optimization experimental results.

Exper.		Ours			Canonical GEP	
Number	Total Len.	Avg. Gene Len.	Maximum	Total Len.	Gene Len.	Maximum
1	105	35	78.30	231	77	78.51
2	10	5	3904.62	184	92	3902.40
3	25	5	25	385	77	25
4	426	14.2	1233.87	780	26	1125.61
5	22	11	499.002	94	47	499.002

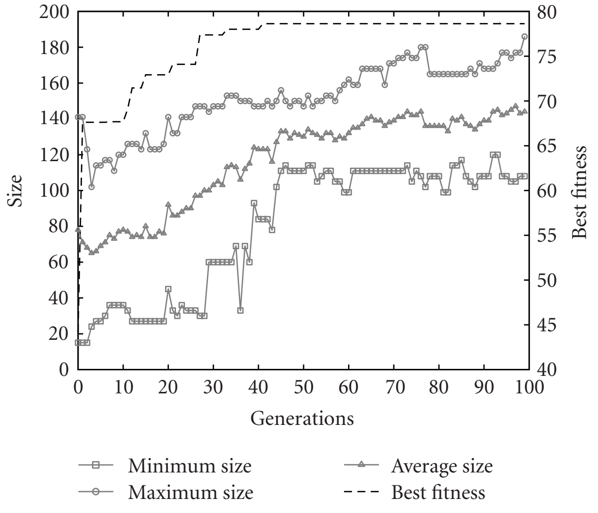


Figure 15: Parameter optimization Experiment 1: chromosome sizes and best fitness.

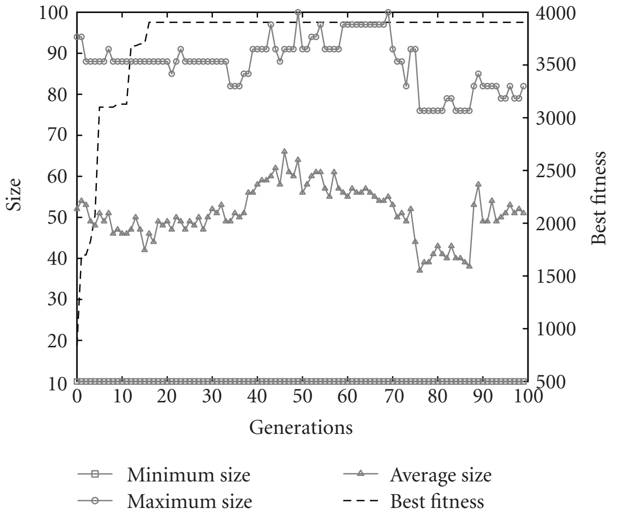


Figure 16: Parameter optimization Experiment 2: chromosome sizes and best fitness.

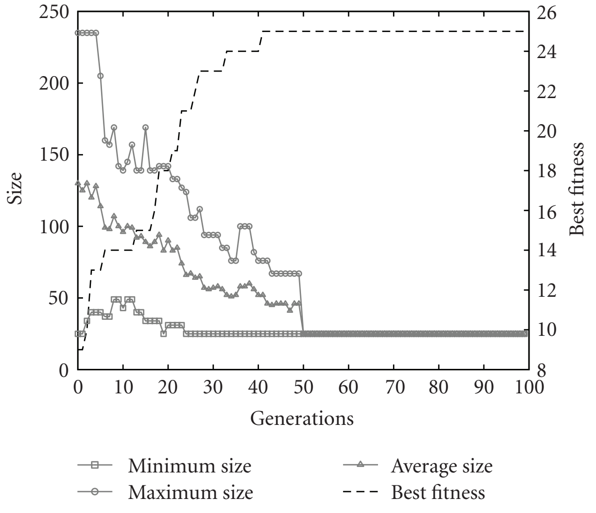


Figure 17: Parameter optimization Experiment 3: chromosome sizes and best fitness.

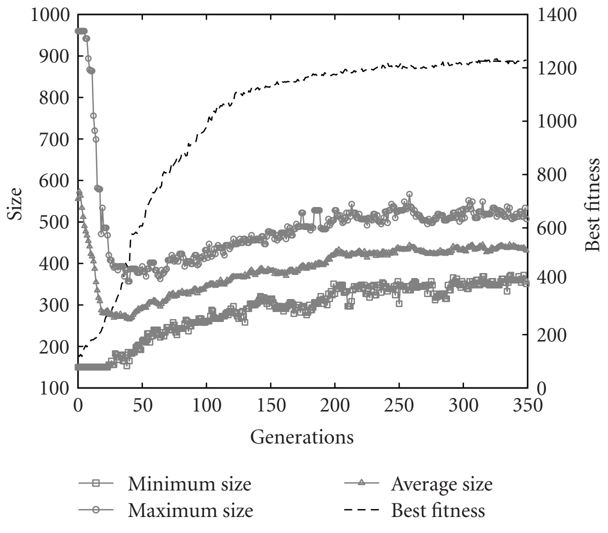


Figure 18: Parameter optimization Experiment 4: chromosome sizes and best fitness.

operator and our other enhancements were 45.5% the size of standard GEP's solutions.

The results of the final parameter optimization experiment were closer to what we had observed during the Symbolic Regression and Classification experiments, with our evolved solutions being approximately 23.4% the size of those evolved by canonical GEP. In this case, both methodologies successfully found the maximum value of Shekel's foxholes.

All of the parameter optimization experiments have shown that our enhancements retained GEP's problem

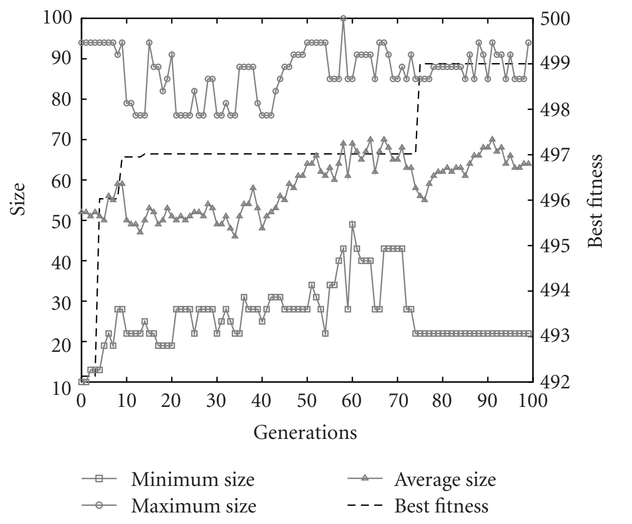


Figure 19: Parameter optimization Experiment 5: chromosome sizes and best fitness.

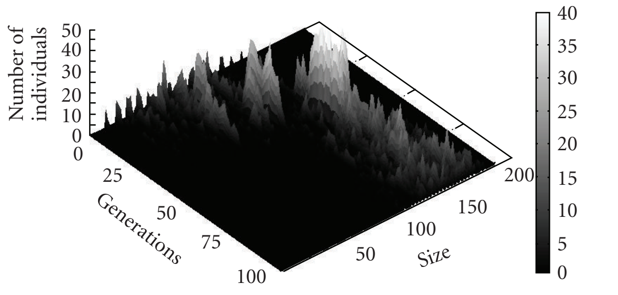


Figure 20: Parameter optimization Experiment 1: chromosome size in the population.

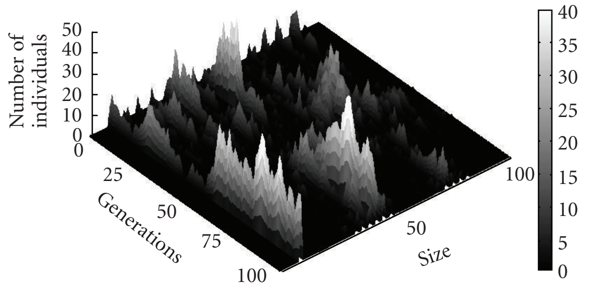


Figure 21: Parameter optimization Experiment 2: chromosome size in the population.

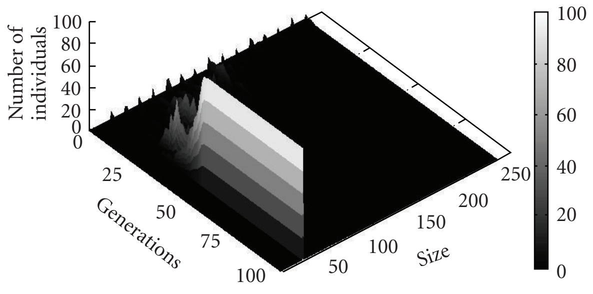


Figure 22: Parameter optimization Experiment 3: chromosome size in the population.

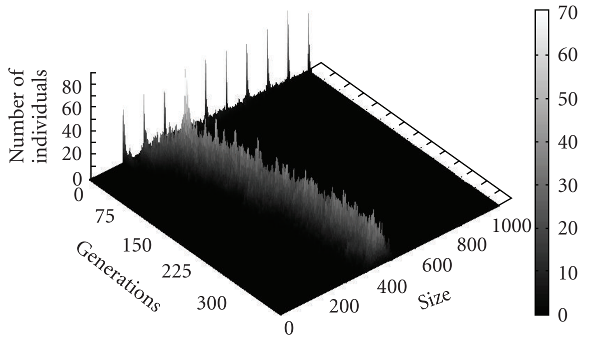


Figure 23: Parameter optimization Experiment 4: chromosome size in the population.

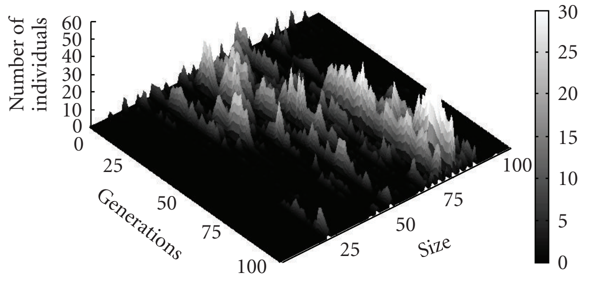


Figure 24: Parameter optimization Experiment 5: chromosome size in the population.

solving ability while allowing it to evolve smaller genomes. While the De Jong functions have been reported [48] to not be an effective test set, they have been repeatedly shown to provide a good metric of the effectiveness of algorithms for a broad range of optimization problems.

A possible limitation is that it is not currently possible to use the ACS mutation operator with our current experimental setup. Since we have not used Automatically Defined Functions (ADFs) [1], we must use a fixed number of genes, one per parameter requiring optimization. While we were still able to obtain good results, we can only speculate that using ADFs and allowing the number of "normal" genes to evolve (similarly to our symbolic regression and classification experiments) would enhance the solutions of more difficult parameter optimization problems.

4.4. General Discussions. Reviewing the results of our experiments, we see that our enhancements to the GEP algorithm consistently produced smaller solutions (sometimes significantly so) than canonical GEP. Since the representation size of a genome has a direct impact on the evaluation of the fitness cases, the reduction in representation size may improve the overall performance of the system, even after considering the additional computation requirements of our new operators. This was indirectly observed during the classification experiments while waiting for the two methodologies to complete their evolutionary runs. When our enhanced algorithm was running, it was noticeably faster than when the standard GEP algorithm was processing the same problem.

The control of the number of genes and the head size of each gene was an implicit part of our GEP run and, thus, we did not require separate clients for optimization. This reduced the overall computational resources required to evolve solutions.

For all of the parameter optimization experiments the ACS mutation operator was disabled and, thus, we were unable to evaluate its potential effectiveness for this class of problems. The operator was disabled because of the evaluation method used. Since our GEP implementation did not use ADFs, it required one gene per parameter to optimize. It is possible that, if we implemented automatically defined functions and used the ACS mutation operator to evolve the number of "normal" genes, we would see different results.

The decision to randomly initialize the genes that were inserted during the ACS mutation phase appears successful. However, it would be interesting to investigate the use of gene cloning, or other methods, in the future.

We observed that the insertion point in the ACS mutation operator for classification and symbolic regression problems was not important because we used a commutative linking function during testing. The insertion point, however, may have been significant because of the way the gene would mix within the population during recombination. Additionally, since the Gene Transposition operator was used, good genes could be reordered within the chromosome. Had we used a noncommutative linking function or homeotic (ADF) genes, the insertion location could have had a greater impact.

Based on the results of our experiments, our new operators were able to successfully self-adaptively tune the two critical parameters of the GEP algorithm, the head domain length, and the number of genes. While our new operators have additional computational costs associated with them, it is hoped that they are offset by the shorter time required to evaluate the fitness functions, because of the smaller representations it evolved.

Our new recombination operators have also been empirically shown to permit crossbreeding and speciation within a single GEP population. They have also been shown to be effective in a distributed environment. However, additional research into the effects of our operators on migration is required.

5. Conclusion

This work presented enhancements to the Gene Expression Programming algorithm that enabled flexible genome representations, endowed self-adaptive characteristics, assisted with maintaining diversity within a population and enhanced the parallelization of the algorithm. In particular, the enhancements addressed issues of evolvability, crossbreeding and speciation, parameter control, and parallelization in canonical GEP.

Through a series of experiments that used an assortment of problem classes, including symbolic regression, classification, and parameter optimization, we have shown that our proposed methodology produced better results and, generally, smaller genome representations than the canonical GEP algorithm and the PGEP-O system [3] (for symbolic regression).

Specifically, the contributions presented in this work were

(1) creation of a new transposition operator, the *Head Insertion Sequence* (HIS), which self-adaptively tunes the head domain length of a gene,

(2) development of a new mutation operator, the *Adaptive Chromosome Size* (ACS) mutation, which mutates the number of genes in an individual to tune the-gene count parameter,

(3) addition of new GEP recombination operators to permit structurally dissimilar individuals to interact. which removed the structural constraints imposed when transferring an individual from one population to another and permitted both crossbreeding and speciation.

Our enhancements to the GEP algorithm also simplified its use, by implicitly controlling the head domain length and number of genes throughout an evolutionary run. By removing the need to set these two critical GEP parameters prior to executing a run, the level of "expert knowledge" required to use GEP is reduced and allows EC novices to use the algorithm more effectively.

The simplification of the algorithm's configuration and the implicit parameter control of the two critical parameters are still subject to the concept of "No Free Lunch" [35]. The "No Free Lunch" theorem states that without *a priori* knowledge of a problem all potential solution methods are equal. While the values of the parameters evolved during a run may not be optimal for all problem types, they are frequently "good enough" and "No Free Lunch" is partially offset by the ease of using the new algorithm. This was seen during our experimental verification of the algorithm and when comparing our methodology to canonical GEP. To determine the GEP experimental baselines, several runs with different head domain length and number of gene parameter values were required, to obtain usable results. Comparatively, with our enhanced algorithm we only needed to start a run sequence and let the algorithm evolve the parameters.

While our enhancements to the GEP algorithm have proven to be successful, they are not without costs and limitations. Since we have added extra operators to enable our metaevolution of the parameters, we also have added additional computational overhead. In particular, the ACS mutation operator has significant overhead when it generates a new gene from random elements. The overhead associated with the new operators may be partially offset by the reduced size of the solution representations (as experienced during our trials), but further experimentation and analysis are required to confirm this.

Another side effect of our self-adaptive method is that we have increased the search space available to the algorithm. This is both a benefit and a liability, since the algorithm can traverse the entire space defined by any combination of head domain length and number of genes. This allows the algorithm to find novel solutions, but also increases the number of potential solutions dramatically, possibly increasing the search time and allowing the algorithm to get stuck in at a nonoptimal solution.

When developing the enhancements to the GEP algorithm, the possibility of introducing bloat, or the excessive creation of introns to protect a genome's functionality, was a major concern. By eliminating the fixed chromosome size (which was necessary to remedy the issues we saw with GEP), the potential for the genome representation and size to grow unchecked became a possibility; even with the parameter control inherent in the new operators. One conjecture for not observing bloat is that the HIS Transposition operator, which is responsible for controling the head size, restructures the genome by adding sections from one domain to another, instead of simply inserting or deleting material. This does not account for the effect of the ACS mutation operator, which mutates the number of genes in a chromosome. However, the selection pressure from the Tournament Selection with Elitism selection method may have provided resistance to unnecessary gene additions. It is possible that in more difficult problems (that require longer runs or larger datasets) we may begin to observe bloat and need to take steps to measure and constrain it.

Related to the previous topic of bloat and introns is the matter of genetic diversity within a population. Our current research did not include any specific mechanisms to measure the diversity of individuals within a population (either in a single population or distributed multipopulation setting), but the genome length statistics, recorded during the experiments, can be used as a simple metric. Using the surface plots of the chromosome lengths (found in Section 4), we can suppose that our methodology maintains a level of genetic diversity throughout a run. While the populations were initially very diverse and chaotic, as the runs progressed, the outliers were reduced and a narrower band of chromosome sizes (and thus diversity) was maintained.

Overall, our enhancements have been shown to be effective at addressing the issues of evolvability, crossbreeding and speciation, parameter control, and parallelization in the canonical GEP algorithm.

5.1. Future Work. Though our enhancements have been effective, there is still work that can be done to further our understanding of them, their relationship and application to Evolutionary Computation in general, and the workings of the GEP algorithm itself.

A detailed study of the effects of our enhancements on the levels of genetic diversity in a population would aid in understanding the mechanisms that make the operators effective. Additionally, applying the "Nonsynonymous to Synonymous Substitution Ratio (Ka/Ks)" [49] to study the rate of evolution, in conjunction with a diversity study, could show where further improvements could be made in the GEP algorithm.

Applying our enhancements to Automatically Defined Functions (ADFs) in GEP could potentially provide interesting results and bears further investigation. This could be particularly useful for difficult or complex parameter optimization problems, since, when using GEP-PO, the number of genes must always equal the number of parameters being optimized. Using ADFs would allow the number of normal genes to be adaptively tuned using the ACS mutation operator.

Further research into the potential of unrestrained chromosome growth, or bloat, and selection pressure in our enhanced GEP algorithm would be interesting, as we did not observe significant bloat during our experiments. In evolutionary computation, any algorithm or representation that allows unrestrained growth and yet demonstrates resistance to bloat warrants further investigation.

The impact of our operators on migration and the exchange of genetic material in distributed setting requires further study. In particular, a thorough examination of our system when running in a distributed, multi-island settings with different connection topologies and migration strategies would be useful for determining the optimal configuration (if possible).

While the enhancements presented in this work enabled crossbreeding and the evolution of different species within a population, we did not specifically implement any niching methods. This could prove to be an interesting avenue of exploration in the future, as it could enhance the algorithm's performance with multimodal problems.

Finally, adapting our enhancements to *neuroevolution*, or the evolution of neural networks, using GEP (such as the GEP-nets algorithm [7]) has great potential. This is because our enhancements could permit size and structure changes to the evolved neural networks, allowing a more dynamic and complicated structure to be evolved.

References

[1] C. Ferreira, "Gene expression programming: a new adaptive algorithm for solving problems," *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[2] K. A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. dissertation, University of Michigan, 1975.

[3] H.-H. Park, A. Grings, M. V. dos Santos, and A. S. Soares, "Parallel hybrid evolutionary computation: automatic tuning of parameters for parallel gene expression programming," *Applied Mathematics and Computation*, vol. 201, no. 1-2, pp. 108–120, 2008.

[4] K. Zhang, S. Sun, and H. Si, "Prediction of retention times for a large set of pesticides based on improved gene expression programming," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pp. 1725–1726, ACM, Atlanta, Ga, USA, July 2008.

[5] Z. Xie, X. Li, B. Di Eugenio, P. C. Nelson, W. Xiao, and T. M. Tirpak, "Using gene expression programming to

construct sentence ranking functions for text summarization," in *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, p. 1381, Association for Computational Linguistics, Morristown, NJ, USA, 2004.

[6] J. Venter and A. Hardy, "Generating plants with gene expression programming," in *Proceedings of the ACM International Conference on Computer Graphics, Virtual Reality and Visualisation in Africa (AFRIGRAPH '07)*, pp. 159–167, ACM, 2007.

[7] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, Springer, Berlin, Germany, 2nd edition, 2006.

[8] M. Ostaszewski, P. Bouvry, and F. Seredynski, "Multiobjective classification with moGEP: an application in the network traffic domain," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference (GECCO '09)*, pp. 635–642, ACM, 2009.

[9] J. Yin, L. Huo, L. Guo, and J. Hu, "Short-term load forecasting based on improved gene expression programming," in *Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA '08)*, pp. 5647–5650, Chongqing, China, June 2008.

[10] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Mass, USA, 1992.

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

[12] J. Reisinger, K. O. Stanley, and R. Miikkulainen, "Towards an empirical measure of evolvability," in *Proceedings of the Workshops on Genetic and Evolutionary Computation (GECCO '05)*, pp. 257–264, ACM, Washington, DC, USA, June 2005.

[13] K. O. Stanley, *Efficient evolution of neural networks through complexification*, Ph.D. dissertation, The University of Texas at Austin, Austin, Tex, USA, 2004.

[14] H. S. Lopes and W. R. Weinert, "Egipsys: an enhanced gene expression programming approach for symbolic regression problems," *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 3, pp. 375–384, 2004.

[15] J. Yue, T. Chang-Jie, Z. Hai-Chun, et al., "Adaptive gene expression programming algorithm based on cloud model," in *Proceedings of the 1st International Conference on BioMedical Engineering and Informatics (BMEI '08)*, vol. 1, pp. 226–230, May 2008.

[16] C. Tang, L. Duan, J. Peng, H. Zhang, and Y. Zong, "The strategies to improve performance of function mining by gene expression programming: genetic modifying, overlapped gene, backtracking and adaptive mutation," in *Proceedings of the 17th Data Engineering Workshop*, 2006.

[17] O. M. Shir and T. Back, "Niching methods: speciation theory applied for multi-modal function optimization," in *Algorithmic Bioprocesses*, pp. 705–729, Springer, Berlin, Germany, 2009.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[19] E. Bautu, A. Bautu, and H. Luchian, "AdaGEP—an adaptive gene expression programming algorithm," in *Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '07)*, pp. 403–406, 2007.

[20] M. Gorges-Schleuter, "Explicit parallelism of genetic algorithms through population structures," *Parallel Problem Solving from Nature*, pp. 150–159, 1991.

[21] J. I. Hidalgo, J. Lanchares, F. Fernández de Vega, and D. Lombraña, "Is the island model fault tolerant?" in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 2737–2744, ACM, London, UK, July 2007.

[22] E. Cantú-Paz and D. E. Goldberg, "Efficient parallel genetic algorithms: theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 221–238, 2000.

[23] J. Berntsson and M. Tang, "Dynamic optimization of migration topology in internet-based distributed genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 1579–1580, ACM, Washington, DC, USA, June 2005.

[24] E. Cantu-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[25] E. Alba and J. M. Troya, "Analyzing synchronous and asynchronous parallel distributed genetic algorithms," *Future Generation Computer Systems*, vol. 17, no. 4, pp. 451–465, 2001.

[26] Z. Skolicki and K. De Jong, "The influence of migration sizes and intervals on island models," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 1295–1302, ACM, Washington, DC, USA, June 2005.

[27] Z. Skolicki and K. De Jong, "The importance of a two-level perspective for island model design," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 4623–4630, 2007.

[28] S.-K. Oh, C. T. Kim, and J.-J. Lee, "Balancing the selection pressures and migration schemes in parallel genetic algorithms for planning multiple paths," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3314–3319, 2001.

[29] S.-C. Lin, W. F. Punch III, and E. D. Goodman, "Coarse-grain parallel genetic algorithms: categorization and new approach," in *Proceeedings of the 6th IEEE Symposium on Parallel and Distributed Processing*, pp. 28–37, Dallas, Tex, USA, October 1994.

[30] E. Alba and J. M. Troya, "Influence of the migration policy in parallel distributed GAs with structured and panmictic populations," *Applied Intelligence*, vol. 12, no. 3, pp. 163–181, 2000.

[31] Y. Lin, H. Peng, and J. Wei, "A niching gene expression programming algorithm based on parallel model," in *Proceedings of the 7th International Symposium on Advanced Parallel Processing Technologies (APPT '07)*, vol. 4847 of *Lecture Notes in Computer Science*, pp. 261–270, Guangzhou, China, November 2007.

[32] J. Wu, C. Tang, T. Li, S. Qiao, Y. Jiang, and S. Ye, "Parallel multi-objective gene expression programming based on area penalty," in *Proceedings of the International Conference on Computer Science and Information Technology (ICCSIT '08)*, pp. 264–268, Singapore, August-September 2008.

[33] Q. Liu, T. Li, C. Tang, Q. Liu, C. Li, and S. Qiao, "Multi-population parallel genetic algorithm for economic statistical information mining based on gene expression programming," in *Proceedings of the 3rd International Conference on Natural Computation (ICNC '07)*, vol. 3, pp. 461–465, August 2007.

[34] X. Du, L. Ding, and J. Jia, "Asynchronous distributed parallel gene expression programming based on estimation of distribution algorithm," in *Proceedings of the 4th International Conference on Natural Computation (ICNC '08)*, vol. 1, pp. 433–437, Jinan, China, October 2008.

[35] 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.

[36] Y. C. Ho and D. L. Pepyne, “Simple explanation of the no-free-lunch theorem and its implications,” *Journal of Optimization Theory and Applications*, vol. 115, no. 3, pp. 549–570, 2002.

[37] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith, “Parameter control in evolutionary algorithms,” *Studies in Computational Intelligence*, vol. 54, pp. 19–46, 2007.

[38] T. Back, “Self-adaptation in genetic algorithms,” in *Proceedings of the 1st European Conference on Artificial Life*, pp. 263–271, MIT Press, 1992.

[39] M. Hai-Jun, LI De-Yi, and S. Xue-Mei, “Membership clouds and membership cloud generators,” *Journal of Computer Research and Development*, pp. 15–20, 1995.

[40] C. Ferreira, “Q&a from personal correspondence,” http://www.gene-expression-programming.com/Q&A03.asp.

[41] J. Branke, A. Kamper, and H. Schmeck, “Distribution of evolutionary algorithms in heterogeneous networks,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO ’04)*, vol. 3102 of *Lecture Notes in Computer Science*, pp. 923–934, Seattle, Wash, USA, June 2004.

[42] R. I. McKay, X. H. Nguyen, J. R. Cheney, M. Kim, N. Mori, and T. H. Hoang, “Estimating the distribution and propagation of genetic programming building blocks through tree compression,” in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference (GECCO ’09)*, pp. 1011–1018, ACM, 2009.

[43] T. W. Anderson, *The Statistical Analysis of Time Series*, John Wiley & Sons, New York, NY, USA, 1971.

[44] T. C. for Learning Technology (CLT) at Ryerson University, “Livedescribe video description software,” September 2009, http://www.livedescribe.com/.

[45] H. Rosenbrock, “An automatic method for finding the greatest or least value of a function,” *The Computer Journal*, vol. 3, pp. 175–184, 1960.

[46] J. Shekel, “Test functions for multimodal search techniques,” in *Proceedings of the 5th Annual Princeton Conference on Information Science and Systems*, 1971.

[47] J. Alami, A. E. Imrani, and A. Bouroumi, “A multipopulation cultural algorithm using fuzzy clustering,” *Applied Soft Computing Journal*, vol. 7, no. 2, pp. 506–519, 2007.

[48] L. D. Whitley, K. E. Mathias, S. B. Rana, and J. Dzubera, “Building better test functions,” in *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 239–247, 1995.

[49] T. Hu and W. Banzhaf, “Nonsynonymous to synonymous substitution ratio ka/ks: measurement for rate of evolution in evolutionary computation,” in *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature*, pp. 448–457, Springer, 2008.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 168653, 10 pages
doi:10.1155/2010/168653

Research Article

Dividing Genetic Computation Method for Robust Receding Horizon Control Design

Tohru Kawabe

Department of Computer Science, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

Correspondence should be addressed to Tohru Kawabe, kawabe@cs.tsukuba.ac.jp

Received 21 August 2009; Revised 2 December 2009; Accepted 15 January 2010

Academic Editor: Oliver Kramer

A new robust Receding Horizon Control (RHC) design approach for the sampled-data systems is proposed. The approach is based on a dividing genetic computation of minimax optimization for a robust finite receding horizon control problem. Numerical example is given to show the effectiveness of the proposed method.

1. Introduction

In last few decades, the Receding Horizon Control (RHC) has been widely accepted in the industries [1]. RHC is an online powerful control method which solves a control problem with respect to each sampling frequency [2]. A significant merit of RHC is easy handling of constraints during the design and implementation of the controller [2, 3].

In the standard RHC formulation, the current control action is derived by solving a finite or infinite horizon quadratic cost problem at every sample time using the current state of the plant as the initial state [1]. It is an online optimization with big calculation amount. Then, the RHC has been applied conventionally to systems with relatively slow-moving dynamics such as petrochemical plants and so on. However, recent advance of computer performance has made it possible to use it for systems with relatively fast-moving dynamics, for example, the mechatronics and so on. Therefore, it is important to develop a practical RHC method for such systems.

Incidentally, a drawback of the standard RHC is explicitly lack of robust property with respect to model uncertainties or disturbances since the online minimized cost function is defined in terms of the nominal systems. A possible strategy for realizing the robust RHC is solving the so-called minimax optimization problem, namely, minimization problem over the control input of the performance measure maximized by plant uncertainties or disturbances. An earliest work was proposed by Campo and Morari [4] and Zheng and Morari [5]. Kothare et al. solve minimax RHC problems with state-space uncertainties through LMIs [6]. Cuzzola et al. improve Kothare's method [6] to reduce conservativeness in [7]. Other methods of minimax RHC for systems with model uncertainty can be found in [8–12]. However, the number of available work of the robust RHC is still limited compared with many methods of robust control synthesis for linear systems being proposed. Main reason of this fact is that the robust (minimax) RHC problem is hard to solve in real-time due to the trade-off with an objective function and constraint conditions in the problem generally. The issue of robust RHC therefore still deserves further attention [2, 3] and the effective approach for the robust RHC is still required, especially in the optimization technique.

Optimization techniques by using evolutionary computation such as Genetic algorithms (GAs) [13, 14] have been recognized to be well suited to many kinds of engineering optimization problem. Multiple individuals can search for multiple solutions in parallel, eventually taking advantage of any similarities available in the family of possible solutions to the problem. Extensions of GAs to many kinds of optimization problems were proposed in several manners [15–17]. For example, Schaffer [15] proposed an extension of the simple GA to accommodate vector-valued fitness measures, which he called the Vector Evaluated Genetic Algorithm (VEGA). Moreover, Goldberg [13] firstly proposed the Pareto-based approach as a means of assigning equal probability of reproduction to all nondominated individuals in the population. Fonseca and Fleming [16] proposed a multiobjective ranking method with the Pareto-based fitness assignment. However, it is uncertain whether to be able to apply these methods effectively to the minimax RHC problem.

Therefore, in this paper, a new minimax robust finite RHC design approach based on a new dividing genetic computation for constrained sampled-data systems with structured uncertainties and disturbance is proposed. This approach is one of the general and practical framework of the minimax robust finite RHC problem of bounded constrained systems. Since the dividing genetic computation uniformly controls the convergence of solutions of optimization problems with some trade-off conditions, it can be effectivefor the minimax RHC problem. Using this approach, we can expect to reduce the conservativeness of control performance and to improve the control performance. Numerical example is given to show these facts.

2. Problem Formulation

The target system in this paper is the sampled-data control system. Hence, the control object with uncertainties is a continuous-time system and the controller is designed in discrete-time. Then, let us consider the following discrete-time LTI (Linear Time-Invariant) state-space model with structured uncertainties and disturbances:

$$\begin{aligned} x(k+1) &= (A + L\Delta R_A)x(k) + (B + L\Delta R_B)u(k), \\ y(k) &= Cx(k) + \eta(k), \end{aligned} \tag{1}$$

where $L\Delta R_A$ and $L\Delta R_B$ denote the structured uncertainties expressed by perturbation of elements in A and B, respectively. A, B,C, L, R_A, and R_B are constant matrices. Δ ($\Delta = \mathrm{diag}\{\delta_1, \delta_2'', \dots\}$) is a structured uncertainties parameters matrix satisfied $\Delta^T\Delta \leq r$. (r is a given constant) And $x(k)$, $u(k)$, $y(k)$, and $\eta(k)$ denote the state, input, measured output, and disturbance vector, respectively. All these vectors and matrices have appropriate dimensions.

Then, we can transform this system as

$$x(k+1) = Ax(k) + Bu(k) + Lw(k), \tag{2}$$

$$z(k) = R_A x(k) + R_B u(k), \tag{3}$$

$$y(k) = Cx(k) + \eta(k), \tag{4}$$

where $w(k) = \Delta z(k)$. We assumed that the system is constrained with following conditions with $N-1$ steps:

$$\begin{aligned} &w^T(k+j)P_w w(k+j) \leq 1, \\ &\eta^T(k+j)P_\eta \eta(k+j) \leq 1, \\ &u^T(k+j)P_u u(k+j) \leq 1, \\ &(j = 0, \dots, N-1), \end{aligned} \tag{5}$$

where P_w, P_η, P_u ($P_w, P_u, P_\eta \succ 0$) are positive symmetric matrices for weights of constraints.

For this systems, to use the RHC, a quadratic performance measure with finite horizon with positive weighting constant matrices Q and R ($Q, R \succ 0$) as

$$J(k) = \sum_{j=0}^{N-1} \|y(k+j+1 \mid k)\|_Q^2 + \|u(k+j \mid k)\|_R^2 \tag{6}$$

is used. $x(k+j \mid k)$, $y(k+j \mid k)$, and $u(k+j \mid k)$ are the predicted state of the plant, the predicted output of the plant and the future control input at time $k+j$, respectively.

Then, the robust RHC is need to solve the following minimax optimization problem at each step k:

$$\begin{aligned} &\min_{u(k+j|k)} \max_{9w(k+j|k),\eta(k+j|k)} J(k) \\ \text{subject to}\quad &w^T(k+j)P_w w(k+j) \leq 1, \\ &u^T(k+j)P_u u(k+j) \leq 1 \\ &\eta^T(k+j)P_\eta \eta(k+j) \leq 1 \\ &(j = 0, \dots, N-1). \end{aligned} \tag{7}$$

Generally, the following state feedback schema is employed:

$$u(k+jk) = -F_{k+j}x(k+jk) \quad (j = 0, 1, \dots N-1), \tag{8}$$

where F_{k+j} is a feedback gain matrix.

Finally, the procedure of robust RHC is summarized as follows. At the current step k, future state values $x(k+j \mid k)$ $(j = 0, \dots, N-1)$ are predicted by using the state space model (2)–(4), and future feedback gain matrix candidates $F_{k+j}(j = 0, \dots, N-1)$ are calculated by solving (7) under (8). Only first gain matrix $F(k+1)$ is employed for an actual control input and the others are discarded. Then, go to next step $k+1$ and repeat same operations.

Namely, the robust RHC requires an online minimax optimization. However, it is difficult to solve this problem as it is at each step, generally. Therefore, the method to eliminate the maximization procedure and transform this problem to simple minimization problem is shown in the next section.

3. Transformation of Minimax Finite RHC Problem

Firstly, introducing the following vectors

$$\begin{aligned} X &:= \left[x(k+1k) \;\; x(k+2 \mid k) \;\; \cdots \;\; x(k+N \mid k)\right]^T, \\ Y &:= \left[y(k+1 \mid k) \;\; y(k+2 \mid k) \;\; \cdots \;\; y(k+N \mid k)\right]^T, \\ U &:= \left[u(k \mid k) \;\; u(k+1 \mid k) \;\; \cdots \;\; u(k+N-1 \mid k)\right]^T, \\ W &:= \left[w(k \mid k) \;\; w(k+1 \mid k) \;\; \cdots \;\; w(k+N-1 \mid k)\right]^T, \\ \Lambda &:= \left[\eta(k \mid k) \;\; \eta(k+1 \mid k) \;\; \cdots \;\; \eta(k+N-1 \mid k)\right]^T \end{aligned} \tag{9}$$

and using state space equations (2)–(4) recursively, we can derive

$$\begin{aligned} X &= \widetilde{A}x(k) + \widetilde{B}U + \widetilde{L}W, \\ Y &= C\widetilde{A}x(k) + C\widetilde{L}W + \Lambda, \end{aligned} \tag{10}$$

where

$$\widetilde{A} := \begin{bmatrix} A & A^2 & \cdots & A^{N-1} \end{bmatrix}^T,$$

$$\widetilde{B} := \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N-2}B & A^{N-3}B & \cdots & B \end{bmatrix}, \tag{11}$$

$$\widetilde{L} := \begin{bmatrix} L & 0 & \cdots & 0 \\ AL & L & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N-2}L & A^{N-3}L & \cdots & L \end{bmatrix}.$$

Hence, we can transform the minimax problem (7) to

$$\begin{aligned} &\min_U \gamma \\ &\max_{W,\Lambda} \ \Pi \le \gamma, \\ \text{subject to} \quad & w^T(k+j) \quad P_w \quad w(k+j) \le 1 \\ & u^T(k+j) \quad P_u \quad u(k+j) \le 1 \\ & \eta^T(k+j) \quad P_\eta \quad \eta(k+j) \le 1 \\ & \qquad (j = 0, \dots, N-1), \end{aligned} \tag{12}$$

where $\gamma > 0$ (scalar parameter) and Pi is defined as follows:

$$\Pi := \left\{ \left\| \widetilde{A}x(k) + \widetilde{B}U + \widetilde{L}W + \Lambda \right\|_{\hat{Q}}^2 + \|U\|_{\hat{R}}^2 \right\},$$

$$\hat{Q} := \begin{bmatrix} Q & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & Q \end{bmatrix}, \qquad \hat{R} := \begin{bmatrix} R & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & R \end{bmatrix}. \tag{13}$$

To eliminate the maximization procedure, we have to remove W and Λ terms in the first constraint. For this, in the first place, following basis for all variables and transformation matrices are defined,

$$\zeta = \begin{bmatrix} x(k) & W^T & \Lambda^T & 1 \end{bmatrix}^T, \tag{14}$$

$$U = H_u\zeta, \tag{15}$$

$$Y = H_y\zeta, \tag{16}$$

$$\Lambda = H_\eta\zeta, \tag{17}$$

$$1 = (H_1\zeta)^T(H_1\zeta), \tag{18}$$

where

$$\begin{aligned} H_u &:= \begin{bmatrix} F\widetilde{A} & F\widetilde{L} & F & 0 \end{bmatrix}, \\ H_y &:= \begin{bmatrix} C\widetilde{A} & C\widetilde{L} & I & 0 \end{bmatrix}, \\ H_\eta &:= \begin{bmatrix} \mathbf{0} & \mathbf{0} & I & 0 \end{bmatrix}, \\ H_1 &:= \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}, \end{aligned} \tag{19}$$

and where

$$I := \text{(identity matrix with appropriate dimension)},$$

$$F := \begin{bmatrix} -F_k & 0 & \cdots & 0 \\ 0 & -F_{k+1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -F_{k+N-1} \end{bmatrix}. \tag{20}$$

By using these, we can express the first constraint condition of problem (12):

$$\max_{W,\Lambda} \left\{ \left\| H_y\zeta \right\|_{\hat{Q}}^2 + \|H_u\zeta\|_{\hat{R}}^2 \right\} \le (H_1\zeta)^T \lambda (H_1\zeta). \tag{21}$$

Please take notice that both the left side and the right side of this inequality are expressed by the quadratic forms and they have positive scalar values. Hence, if the inequality is held by maximum values of W and Λ in left side, this inequality must be held by any other values of them. This fact means that we can eliminate the maximization procedure in the first constraint. We can only check the following condition instead of the first constraint of problem (12):

$$\left\{ \left\| H_y\zeta \right\|_{\hat{Q}}^2 + \|H_u\zeta\|_{\hat{R}}^2 \right\} \le (H_1\zeta)^T \lambda (H_1\zeta). \tag{22}$$

In the second place, $H_w(j)$ is defined. This matrix picks out the suitable block from W and satisfies the relation of $w(k+j) = H_w^{(j)}\zeta$. Then, we can derive

$$\left(H_w^{(j)}\zeta\right)^T P_w \left(H_w^{(j)}\zeta\right) \le (H_1\zeta)^T(H_1\zeta) \quad (j = 0, \dots, N-1). \tag{23}$$

For the constraints of η, u and z, we can derive the following relations in the same way:

$$\begin{aligned} \left(H_\eta^{(j)}\zeta\right)^T P_\eta \left(H_\eta^{(j)}\zeta\right) &\le (H_1\zeta)^T(H_1\zeta), \\ \left(H_u^{(j)}\zeta\right)^T P_u \left(H_u^{(j)}\zeta\right) &\le (H_1\zeta)^T(H_1\zeta) \quad (j = 0, \dots, N-1). \end{aligned} \tag{24}$$

Furthermore, by using (14)–(21), all constraints in minimax problem (12) can be transformed into

$$\begin{aligned}
&\text{subject to}\quad \forall\zeta\neq 0;\quad \zeta^T\left(H_1^T\lambda H_1 - H_x^T\hat{Q}H_x - H_u^T\hat{R}H_u\right)\zeta \geq 0\\
&\zeta^T\left(H_1^T H_1 - \left(H_w^{(j)}\right)^T P_w H_w^{(j)}\right)\zeta \geq 0\\
&\zeta^T\left(H_1^T H_1 - \left(H_u^{(j)}\right)^T P_u H_u^{(j)}\right)\zeta \geq 0\\
&\zeta^T\left(H_1^T H_1 - \left(H_\eta^{(j)}\right)^T P_\eta H_\eta^{(j)}\right)\zeta \geq 0\\
&(j = 0,\ldots,N-1).
\end{aligned} \tag{25}$$

Then, we can transform the original minimax problem (7) to the following one by using S-procedure [18]:

$$\begin{aligned}
&\min_F \gamma\\
&\text{subject to}\quad H_1^T\gamma H_1 - H_y^T\hat{Q}H_y - H_u^T\hat{R}H_u\\
&\qquad - \sum_{j=0}^{N-1}\left[\tau_j^w S_j^w + \tau_j^u S_j^u + \tau_j^\eta S_j^\eta\right] \geq 0\\
&\qquad (j = 0,\ldots,N-1),
\end{aligned} \tag{26}$$

where

$$\begin{aligned}
S_j^w &= \left(H_1^T H_1 - \left(H_w^{(j)}\right)^T P_w H_w^{(j)}\right),\\
S_j^u &= \left(H_1^T H_1 - \left(H_u^{(j)}\right)^T P_u H_u^{(j)}\right),\\
S_j^\eta &= \left(H_1^T H_1 - \left(H_\eta^{(j)}\right)^T P_\eta H_\eta^{(j)}\right),
\end{aligned} \tag{27}$$

and where τ_j^w, τ_j^u, τ_j^η, and τ_j^z are positive semidefinite scalars. It must be noted that this transformation satisfies only a sufficient condition of S-procedure, since S-procedure is not the so-called "*lossless*" in this case. We cannot therefore avoid that the design results are slightly conservative. Nevertheless, we can expect the reduction of conservativeness in design result by this technique in contrast with the results by preexisting methods, because the conservativeness caused by S-procedure is too small to put a matter for practical purposes.

Finally, using "*Schur-complement*" [19], we can transform the minimax optimization problem (7) into the following problem:

$$\begin{aligned}
&\min_F \gamma\\
&\text{subject to}\quad \begin{bmatrix} H_1^T\gamma H_1 - \Sigma & H_y^T & H_u^T\\ H_y & \hat{Q}^{-1} & 0\\ H_u & 0 & \hat{R}^{-1}\end{bmatrix} \geq 0,\\
&\tau_j^w, \tau_j^u, \tau_j^\eta \geq 0 \quad (j = 0,\ldots,N-1),
\end{aligned} \tag{28}$$

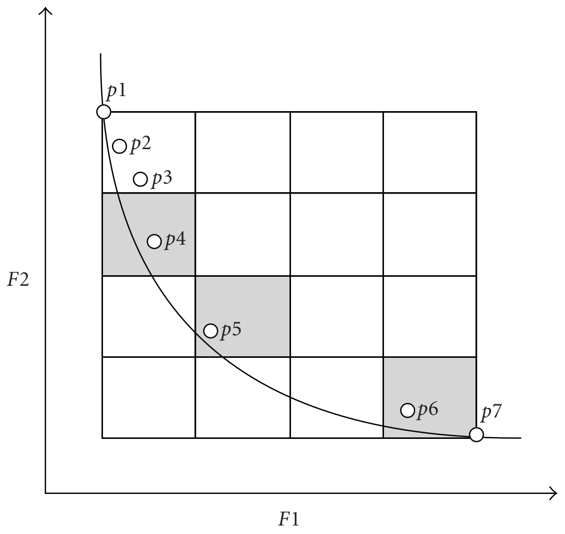


FIGURE 1: Dividing strategy.

where

$$\Sigma := \sum_{j=0}^{N-1}\left[\tau_j^w S_j^w + \tau_j^u S_j^u + \tau_j^\eta S_j^\eta\right]. \tag{29}$$

It is known that this problem has trade-off with the objective function and the constraint condition. Therefore, the fast method of finding nondominated solutions with respect to the trade-off as a lot as possible is needed. Then, the method using dividing genetic computation is shown in the next section.

4. RHC Method with Adaptive DA Converter Using Dividing Genetic Computation

4.1. Dividing Genetic Computation. To find the best possible nondominated solutions for the RHC problem (28), a partial convergence of nondominated solutions must be avoided. Therefore, a dividing method which uniformly controls the distribution of solutions is proposed. The proposed method assigns all nondominated individuals to prespecified regions. An example of the dividing strategy in two objective minimizing problem is shown in Figure 1. The dividing genetic computing method consists of following procedure. First, the objective space is divided into prespecified regions. The edge points of the whole region corresponds to the best solutions for each objective function. In Figure 1, the individuals p_1 and p_7 match them. Then, the fitness f_i of the individual p_i is defined as $f_i = 1/n_i$. The value of n_i denotes the number of nondominated solutions in the identical region with the individual p_i. In the example, the fitness of the individuals illustrated in the figure corresponds to the following values $(f_1, f_2, f_3, f_4, f_5, f_6, f_7) = (1/3, 1/3, 1/3, 1, 1, 1/2, 1/2)$. In the proposed evolutionary computing method, let us define a neighborhood to every individual as follows: two objective functions of a multiobjective problem are selected by using prespecified selective probabilities. Individuals are arranged on the two-dimensional coordinates, and the neighborhood of an individual is calculated by using the relative distance between all individuals.

The crossover operator is made locally in each neighborhood in parallel. Even if the fitness of an individual is relatively very high in a population, it can spread over the succeeding populations only through an overlap of the neighborhood. This prohibits a rapid increase of an relatively high-performance individual, and then, the population diversity is favorably maintained. The evolutionary operators are defined as follows.

(a) The selection is done by considering the number of individuals in the 2-dimensional objective space. That is, the fitness Γ_i of the individual p_i is defined as $\Gamma_i = 1/n_i$. The value of n_i denotes the number of individuals in the identical region with the individual p_i. The proportional fitness method is employed in the selection process.

(b) BLX-α method is employed for crossover. The mate of crossover is chosen randomly in the neighborhood.

(c) The real-code string representation is employed for candidate solution.

(d) Mutation is designed to perform random exchange; that is, it selects some bits randomly in an individual and exchanges their values. Boundary mutation and nonuniform mutation are used to avoid the premature convergence of the solutions.

The procedure of dividing genetic computation approach consists of the following steps.

Step 1. Set a generation number $t = 0$. Randomly generate an initial population $P(t)$ of M individuals.

Step 2. Calculate the fitness of each individual in the current population according to the distribution of the objective space.

Step 3. Select M individuals according to above fitness; then the mate of the individuals is chosen randomly in the neighborhood.

Step 4. Generate a new population $P'(t)$ from $P(t)$ by using a crossover operator.

Step 5. Apply a mutation operator to the newly generated population $P'(t)$.

Step 6. Calculate the fitness both of $P(t)$ and $P'(t)$.

Step 7. Select M individuals from all population members on the basis of the fitness.

Step 8. If a terminal condition is satisfied, stop and return the best individuals. Otherwise set $t = t + 1$ and go to Step 2.

In this procedure, update of the current population size is always constant M. Here, to avoid the rapid loss of genetic diversity, multiple equivalent individuals are eliminated from the current population.

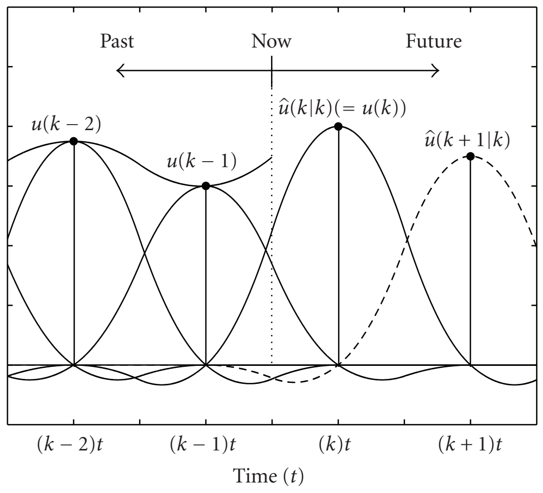


FIGURE 2: Interpolation based on a 2nd-order spline sampling function using predictive future control inputs.

As a result of exploratory experiments using the multiobjective ranking [16], the following facts have been obtained by comparison with the standard genetic algorithm (GA).

(i) By using the proposed method, the solutions are widely distributed in the trade-off surface, and the search performance does not deteriorate significantly.

(ii) The standard GA approaches cause the partial convergence of the solutions because of stochastic errors in the iterative process.

(iii) It is clear that the proposed method can seek for the widely distributed solutions in comparison with the standard GA.

Therefore, it is judged that the proposed dividing genetic computation method is expected to be effective for the minimax RHC problem.

4.2. Interpolation Using Predictive Control Inputs. Generally, the interpolation of samples in the DA conversion is executed by a convolution of samples by using sampling function. In the case of sampled-data control system, to interpolate the current interval, the future sample is need. But, the information about future sample is unable to be obtained in the present time. Hence, we need to wait to obtain this future sample information. As a result, the time-delay is occurred. However, in the case of controlling systems with relatively fast-moving dynamics, such as robots or vehicles, the method with long time-delay is unable to be applied. That is the point. Therefore, a new idea to use the predictive control inputs obtained by RHC for interpolation is proposed.

In RHC, the optimal control inputs $\{\hat{u}(k \mid k), \hat{u}(k+1 \mid k), \dots, \hat{u}(k+N-1 \mid k)\}$ are calculated in each step, and only the first control input $\hat{u}(k \mid k)$ is used as a real control input. Therefore, we consider to use the other optimal control inputs $\{\hat{u}(k+1 \mid k), \hat{u}(k+2 \mid k), \dots\}$ as virtual future sampling

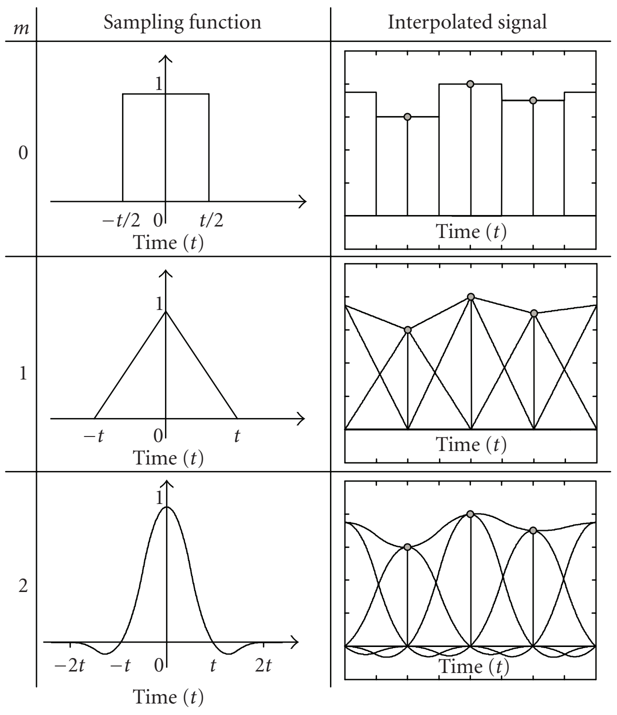


FIGURE 3: Sampling functions Ψ and their interpolations ($m = 1, 2, 3$) (τ: sampling interval).

points. Actually, it is only necessary to use the optimal control inputs which are need for interpolation according to the sampling function. Figure 2 shows an example of this way by using the 2nd-order spline function for interpolations. Only $\hat{u}(k+1 \mid k)$ is used as a virtual future sampling point in this case.

Hence, by using the predictive control inputs for interpolation, it becomes possible to reduce the time-delay in the DA conversion, and the total waiting-time is just only computation time of optimization in current step.

Of course, one needs to take account that there is a difference between virtual future sampling points and real sampling points like $\hat{u}(k+1 \mid k) \neq u(k+1)$ in future step. However, we consider that this point is not a critical problem because the influence on interpolated waveform due to prediction error is not so big compared to the scale of prediction error. Although the differentiability of each sampling function is lost at sampling points, this also does not become a critical problem compared to the zero-order hold, and it is possible to keep a certain level of smoothness.

4.3. Adaptive DA Converter. The spline functions provide various sampling functions with all kinds of orders. Therefore, we consider switching the spline functions optimally according to the system status in the adaptive DA converter. In this paper, we use the spline functions with the order $m = 0, 1, 2$ as sampling functions. Namely, in the case of $m = 0$, the sampling function is equivalent to the staircase function. In the case of $m = 1$, it is the 1st-order piecewise polynomial function, and in $m = 2$, 2nd-order one as shown in Figure 3. Appropriate selecting the values of m according to the object enables to deal with DA conversion flexibly and precisely in the interpolation operation. Although the interpolation is more precisely in the case of using the spline function with $m = 3$ or more, it is difficult to apply to fast-moving dynamic systems due to the bigger amount of calculation. Therefore we use only the spline functions with the order $m = 0, 1, 2$.

The interpolated signals in the closed-open interval $[k\tau, (k+1)\tau)$ using these sampling functions are obtained as follows:

$$u(t) = \sum_{l=k}^{k+1} \left\{ u(l) \cdot {}^{1,2}\Psi(t - l\tau) \right\} \quad (m = 0, 1),$$
$$u(t) = \sum_{l=k-1}^{k+2} \left\{ u(l) \cdot {}^{3}_{[c]}\Psi(t - l\tau) \right\} \quad (m = 2), \tag{30}$$

where $\Psi(\cdot)$ is sampling function as shown in Figure 3, and where $u(t)$ and $u(l)$ are analog signal and digital signal, respectively.

Figure 4 shows the entire controlled system with the proposed RHC and the adaptive DA converter. As shown in Figure 4, the adaptive DA converter has internal model with sampling interval τ/d. Please take notice that this internal model 2 is different from interval model 1 in which sampling interval is just τ. The interval to be interpolated is also partitioned to d sections, and the partitioning points $u_m(j;k), (j = 1, 2, \ldots, d-1)$ on interpolated waveforms are used for the selection of parameter m, that indicates the degree of spling sampling functions:

The partition points $u_m(j;k)$ are calculated as follows,

$$u_m(j;k) = \sum_{1=k-\phi}^{k+\phi-1} \left\{ u(l) \cdot {}^{m}\psi\left((k-1)\tau + \frac{\tau}{d} \cdot j - l\tau \right) \right\} \qquad (j = 1, 2, \ldots, d-1), \tag{31}$$

where ϕ is the number of samples which the sampling function needs for interpolation, and it is adjusted according to the sampling function.

Figure 5 shows the difference of the interpolation and partition points according to the sampling function with $m = 0, 1, 2$ in the case of $d = 5$. How to select the values of m in each interval is summarized as follows. Each value of sample on the partition point is calculated from the state-space equation (2) in the current interval. The evaluation values using $J(k)$ in (6) are calculated in each m. Then, the parameter m whose evaluation value is the smallest is selected for this interval.

From several test simulation results, it has been obtained that the partition number $d = 5$ is most appropriate by the trade-off between computation time and precision.

5. Numerical Example

In this section, an example that illustrates the effectiveness of the proposed method is given. The example is adopted from the benchmark problem in [20] as shown in Figure 6.

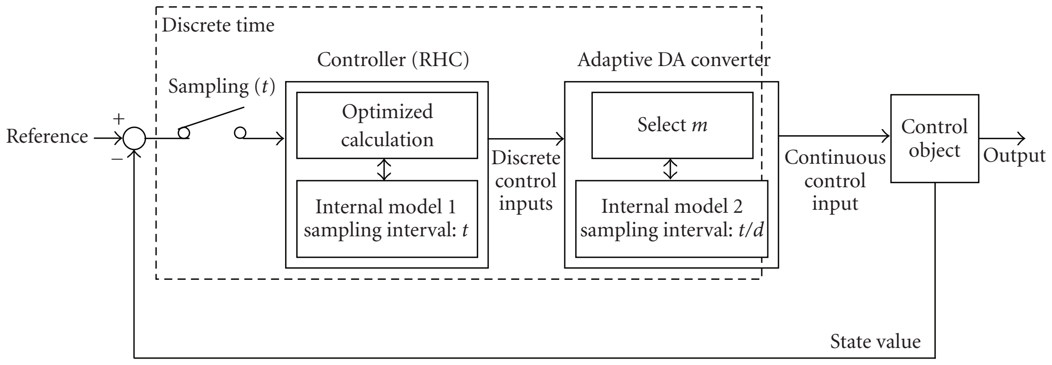


Figure 4: Proposed RHC systems with adaptive DA converter.

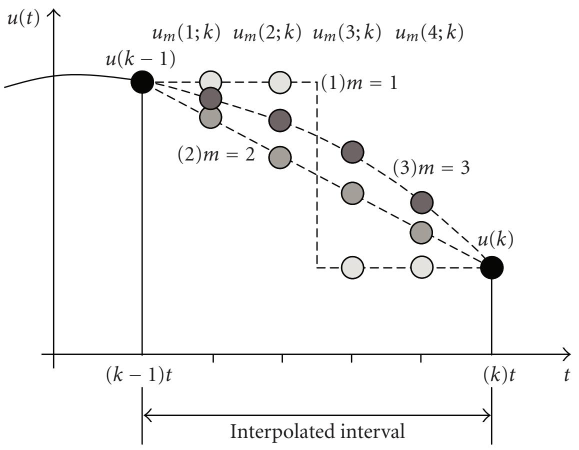


Figure 5: Interpolation ways ($d = 5$).

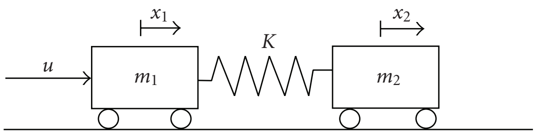


Figure 6: Coupled spring-mass system.

Although the original system consists of a two-mass-spring system without noise for output, the bounded noise is added to demonstrate the proposed method. The state-space equations are obtained as follows:

$$\frac{d}{dt}\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.1 & 0 \\ 0 & 1 & 0 & 0.1 \\ -0.1K/m_1 & 0.1K/m_1 & 1 & 0 \\ 0.1K/m_2 & 0.1K/m_2 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.1/m_1 \\ 0 \end{bmatrix} u(k), \tag{32}$$

Table 1

α in the BLX-α method	0.5
Population size	50
Mutation rate	0.10
Maximum generation	100
The number of dividing region	$50 \times 50 = 2500$

Table 2: Upper bound values of $K_{\max}$.

Method	$K_{\max}$
Proposed method	79.47
Cuzzola's method [7]	68.28
Kothare's method [6]	45.36

where m_1 and m_2 are the two masses and K is the spring constant. State variables x_1 and x_2 are the positions of the two masses, respectively, and x_3 and x_4 are their velocities. Now we assume the following perturbations of m_2 and K:

$$\begin{gathered} m_2 \in \{m_2 \mid 1 \le m_2 \le 10\}, \\ K \in \{K0.5 \le K \le K_{\max}\}, \end{gathered} \tag{33}$$

and m_1 is constant equaled to 1. The weighting matrices are fixed as $Q = I$ and $R = 0.5$. The constraint condition of input $|u(k)| \le 1$ must be satisfied. This means that $P_u = 1$. And the constraints of external disturbance, η, are set $P_\eta = 36.0$. A prediction horizon of the RHC is set $N = 10$.

Furthermore, the following GA parameter specifications are used in the simulation. These values have been selected from several tests(see Table 1).

The results as follows are the best ones in having repeated 20 times.

Figure 7 shows the closed-loop response of the output. From this figure, we can say that the proposed method has good robust performance.

Figure 8 shows the change of the parameter m of the adaptive DA converter. From this figure, we can see that the spline function with $m = 0$ (staircase function) is likely to be selected when the control input stays flat, and the function with $m = 1$ (piecewise linear function) is selected when the control input changes rapidly. The function with

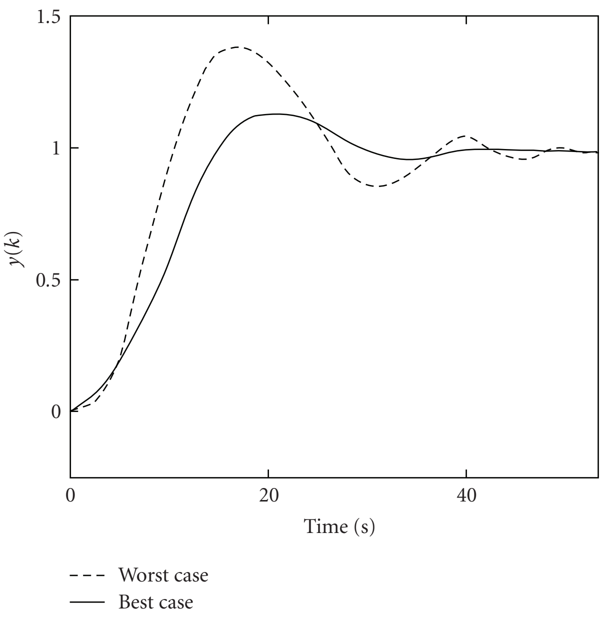


Figure 7: Closed-loop response.

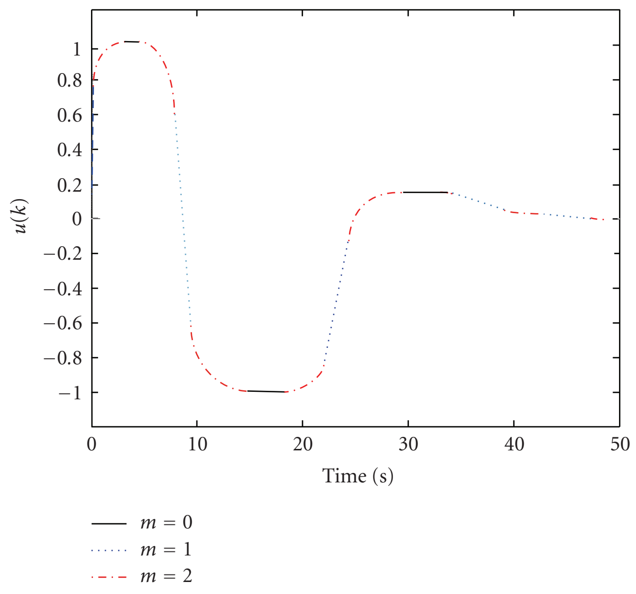


Figure 8: Illustration of Switching ways of sampling functions in the interpolation.

$m = 2$ (piecewise quadratic function) is also likely to be selected when the control input changes smoothly. These are natural results, but please take notice that the tiny difference of control input causes a big influence on the result, in the case of the systems with fast-moving dynamics. Therefore, it is important to select the appropriate value of m in each interval flexibly for better control performance. Moreover, longer the sampling interval, the improvement of control performance is expected to be more conspicuous using the proposed method.

To show the fact that the proposed method can reduce the conservativeness, the maximum values of $K_{\max}$ by the proposed method, the technique in [6], and the one in [7] are searched, respectively, within the limits of holding the feasibility of the robust RHC problem. Then results obtained are indicated in Table 2.

From Table 2, we can see that the result by proposed method is much better than the one in [6] and slightly better than the one in [7]. We can see therefore that the proposed method with the dividing genetic computation can realize the less conservative control performance than the preexisting methods in [6, 7].

To examine the performance of the dividing genetic computation method, comparisons of computation time with NSGA-II [21] and SPEA-II [22] are done.

Computation environment using a software, "MATLAB 7.8.0", is asshown in Table 3.

Table 3

CPU	Intel Core2 Duo U7700 1.33 GHz
Memory	2 GB RAM
OS	Windows Vista Business

Then, maximum computation time of the feedback gain matrix F per each step in the robust RHC by using the dividing genetic computation method is 0.04 second. On the other hand, the times by NSGA-II and by SPEA-II are indicated by the same value 0.02 second. Although the proposed method takes twice time compared with NSGA-II and SPEA-II, it can be said that the proposed method can be practicable in such a time.

Moreover, the upper bound values of $K_{\max}$ by the proposed method, by NSGA-II [21], and by SPEA-II [22] are calculated, respectively. As a result, values are obtained: 79.47 by proposed method as above mentioned, 78.98 by NSGA-II, and 79.28 by SPEA-II, respectively. Judging from this, we can say that the proposed method might be somewhat excellent for the reducing the conservativeness of control performance compared with them.

6. Conclusion

The new approach of minimax robust finite RHC method based on dividing genetic computation for constrained sampled-data control systems with structured uncertainties and disturbance has been proposed. At the same time, a numerical example is given to show that the proposed method improves the control performance.

Although the dividing genetic computation method is able to uniformly control the convergence of solutions of the minimax RHC design problems, more performance investigation of this method is compared with other GA methods, for example, NSGA-II, SPEA-II, and so on, or other heuristic optimization methods, for example, Particle Swarm Optimization [23], Ant Colony Optimization [24], and so on, as future works.

Moreover, I also need to develop the selection method of the best sampling function according to the control object, and need to make sure the effectiveness of the proposed method in various control objects as future works.

Appendix

The proposed minimax approach is easily extended to systems with other constraints which are specified by ellipsoidal bounds, for example, state estimation errors, and so on as follows.

In the case that $x(k)$ is not full measured, we need to estimate $x(k)$, where the bound of estimation error $e(k) = x(k) - \hat{x}(k)$ is guaranteed as an ellipsoidal set as

$$e^T(k)P_e e(k) \leq 1, \tag{A.1}$$

where P_e is a positive symmetric matrix for weight. This specification of estimation error is standard one. Now we introduce H_e as

$$H_e := \begin{bmatrix} 1 & 0 \cdots 0 & -\hat{x}(k) \end{bmatrix}, \tag{A.2}$$

and then the relation of $e(k) = H_e\zeta$ is hold. And the condition below is also held:

$$\zeta^T \left(H_1^T H_1 - H_e^T P_e H_e \right) \zeta \geq 0. \tag{A.3}$$

Since this condition has same form as other constraints in (7), we can include this condition into the condition of problem (26) by using a new variable τ_e. Furthermore, in this case, a new output equation with measurement noise $\psi(k)$ is needed as follows instead of (4):

$$y(k) = Cx(k) + \psi(k) \quad \left(\psi^T(k) P_\psi \psi(k) \leq 1 \right). \tag{A.4}$$

We can also include this constraint into the condition of problem (26) by using a new variable τ_ψ.

Acknowledgment

The author gratefully thanks the anonymous reviewers for their valuable comments and editorial assistance.

References

[1] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[2] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[3] A. Bemporad and M. Morari, "Robust model predictive control: a survey," in *Robustness in Identification and Control*, A. Garulli, A. Tesi, and A. Vicino, Eds., vol. 245 of *Lecture Notes in Control and Information Sciences*, pp. 207–226, Springer, London, UK, 1999.

[4] P. J. Campo and M. Morari, "Robust model predictive control," in *Proceedings of the American Control Conference*, pp. 1021–1026, 1987.

[5] Z. Q. Zheng and M. Morari, "Robust stability of constrained model predictive control," in *Proceedings of the American Control Conference*, pp. 379–383, 1993.

[6] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[7] F. A. Cuzzola, J. C. Geromel, and M. Morari, "An improved approach for constrained robust model predictive control," in *Proceedings of the European Control Conference*, 2001.

[8] J. C. Allwright and G. C. Papavasiliou, "On linear programming and robust model-predictive control using impulse-responses," *Systems and Control Letters*, vol. 18, no. 2, pp. 159–164, 1992.

[9] J. H. Lee and Z. Yu, "Worst-case formulations of model predictive control for systems with bounded parameters," *Automatica*, vol. 33, no. 5, pp. 763–781, 1997.

[10] A. Bemporad, "Reducing conservativeness in predictive control of constrained systems with disturbances," in *Proceedings of the 37th IEEE Conference on Decision and Control (CDC '98)*, vol. 2, pp. 1384–1389, December 1998.

[11] A. Bemporad and A. Garulli, "Output-feedback predictive control of constrained linear systems via set-membership state estimation," *International Journal of Control*, vol. 73, no. 8, pp. 655–665, 2000.

[12] P. O. M. Scokaert and D. Q. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, New York, NY, USA, 1989.

[14] Z. Michalewicz and C. Janikow, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, London, UK, 1992.

[15] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms (ICGA '85)*, pp. 93–100, 1985.

[16] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[17] D. H. Loughlin and S. Ranjithan, "The neighborhood constraint method: a genetic algorithm-based multiobjective optimization technique," in *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA '95)*, pp. 666–673, 1995.

[18] S. Boyd and C. H. Barratt, *Linear Controller Design: Limits of Performance*, Prentice-Hall, Upper Saddle River, NJ, USA, 1991.

[19] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.

[20] B. Wie and D. S. Bernstein, "Benchmark problems for robust control design," *Journal of Guidance, Control, and Dynamics*, vol. 15, pp. 1057–1059, 1992.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm," in *Proceedings of the International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN '01)*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., pp. 95–100, 2001.

[23] 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.

[24] T. Stützle and H. H. Hoos, "MAX-MIN ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 781598, 14 pages
doi:10.1155/2010/781598

Research Article

Modeling and Evolutionary Optimization on Multilevel Production Scheduling: A Case Study

Ruifeng Shi,[1] Chunxia Shangguan,[2] and Hong Zhou[3]

[1] *School of Control & Computer Engineering, North China Electric Power University, Beijing 102206, China*
[2] *College of Engineering Science & Technology, Shanghai Ocean University, Shanghai 201306, China*
[3] *School of Economics & Management, Beihang University, Beijing 100083, China*

Correspondence should be addressed to Ruifeng Shi, shi.ruifeng@126.com

Received 31 August 2009; Revised 27 February 2010; Accepted 16 April 2010

Academic Editor: Chuan-Kang Ting

Multilevel production scheduling problem is a typical combinatorial optimization problem in a manufacturing system, which is traditionally modeled as several hierarchical sublevel problems and optimized at each level, respectively. An integrated model, which can cope with the whole multilevel scheduling information simultaneously, is proposed in this paper, and a specific evolutionary algorithm is designed to solve the integrated model with a twin-screw coding strategy. In order to evaluate the performance of the new algorithm, a real 3-level production scheduling problem is employed for case study, and two typical metaheuristic algorithms, a genetic algorithm (GA) and a simulated annealing (SA), are also employed for comparison study. Experimental simulation results show that our proposed modeling and optimization method has outperformed the other ones.

1. Introduction

Multi-level production scheduling problem is a typical combinatorial optimization problem in manufacturing system, which is traditionally modeled as several hierarchical sublevel problems, and solved by single-level scheduling methods level by level [1–4]. As we know, the trends of manufacturing system development are increasing the competitivity of the company, promoting the customers' service level, integrating the global manufacturing information, and processing flexible task flows in a more efficient way [5, 6]. In order to cope with these trends, researchers need to optimize plant operations and total activities from suppliers to customers and help manufacturers to find their ways in global optimization and support the needs of manufacturing at the same time. Along with all these scenarios, problem modeling and its optimization techniques play important roles in achieving these goals [7–10].

Generally speaking, multi-level production scheduling problem (MLPS) is one kind of hierarchical production planning problem, which considers a set of jobs on given machines with predefined sequence, while discarding to cope with a lot size problem [11–13]. Jobs in an MLPS problem belong to different product levels, and the higher level product cannot begin its process operation until all its subproducts in the lower level are finished. The general way to solve an MLPS problem is decomposing the entire problem into several sublevel problems according to its product level structure and optimize these subproblems within each level [14]. The product level structure of a typical MLPS problem can be illustrated as in Figure 1, in which job 1 belongs to level 1; jobs 2, 3 belong to level 2; jobs 4, 5, 6, 7, 8 belong to level 3. Besides, the process operation precedence between adjacent levels is as follows: job 2 (level 2) cannot begin its process operation, until all its sub-jobs 4, 5 (level 3) are finished; job 1 (level 1) cannot begin its process operation, until all its subjobs 2, 3 (level 2) are finished.

2. Related Works

During the past decades, some researches have tried their efforts to demonstrate that a hierarchical production planning technique is the most efficient way in solving a multistage, multilevel production planning problem [15–17], while other researchers believed in that a monolithic/integrated method is better than a hierarchical one.

These researchers proposed many integrated multi-level production planning models for various problems and obtained some exciting conclusions. But most of them are focused on coping a scheduling problem with a lot sizing problem simultaneously; few attention is taken on the MLPS described as in Figure 1. The reason may lied in the fact that a batch production module plays a major role in most manufacturing companies during the past decades, but a small batch, multiclass production module becomes more and more popular in current factories, especially in some high technological industries. This leads researchers to find more efficient models and optimization techniques to solve the MLPS in recent years [18–20].

Since the occurrence of modern heuristic optimization algorithms, like evolutionary algorithm (EA), simulated annealing (SA), coevolutionary algorithm (COEA), ant colony (AC), and particle swarm optimization (PSO), and so forth, solving a large scale MLPS with an integrated model becomes possible within an acceptable computational cost. More and more attentions are attracted to solve this NP complete (or NP-Hard) problem with an integrated model, and various problem-dependent heuristic algorithms are proposed to enhance the efficiency of the algorithms, among which evolutionary algorithm is the most attractive/preferred one [21–23].

Usually, an evolutionary algorithm is designed to solve a single-level scheduling problem, and an MLPS integrated model can be solved with a hierarchical looped EA, in which precedence prerequisite information can be satisfied with transferring the ready time and release time between adjacent levels. The result is greatly improved by contrast to a hierarchical technique. But the computational cost with this scheme may grow rapidly as the problem scale is increasing. Besides, the optimality of the final result is doubted by many classical optimization mathematicians. This leads researchers turning to new gene expression and evolving strategies to overcome those drawbacks.

In the following sections, Section 3 builds up a general integrate 3-level production scheduling optimization model. Section 4 proposes a twin-screw-coded hybrid evolutionary algorithm to solve the integrated model. Section 5 employs a real 3-level production scheduling case study to evaluate the performance of the proposed modeling and optimization technique, and Section 6 gives the conclusion of this study and highlights some perspectives for future study.

3. Mathematical Integrated Model for a Typical Three-level Production Scheduling Problem

In general, a multi-level production scheduling problem (Figure 1) can be described as follows: a final product consists of several (n) assemblies, each assembly consists of some ($n_{(ij)}(i = 1, 2, \ldots, n)$) subassemblies, and each subassembly consists of several ($n_{(ij)}$ ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n_{(ij)}$)) parts,.... All the jobs, including assemblies, subassemblies, and parts, should be processed through $m_{(i)}$, $m_{(ij)}$, and $m_{(ijk)}$ operational sequences with given orders, where $m_{(i)}$ ($i = 1, 2, \ldots, n$) represents the maximum operation number of

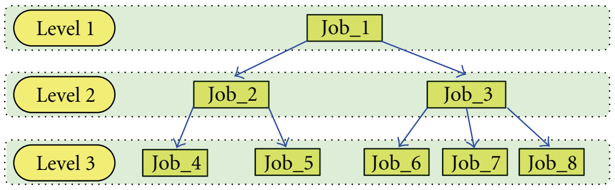


Figure 1: An example of hierarchical structure of MLPS problem.

assembly (i), $m_{(ij)}$ ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n_{(i)}$) represents the maximum operation number of subassembly (ij), and $m_{(ijk)}$ ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n_{(i)}$, $k = 1, 2, \ldots, n_{(ij)}$) represents the maximum operation number of part (ijk). Besides, one assembly cannot begin its process operation until all of its subassemblies are finished and assembled into the assembly. The final product is assembled by the assemblies. Generally, **min**$\{makespan\}$ is considered as the optimization objective. In this section, an integrated 3-level MLPS model is built up, which can be stated as (1)–(24):

$$\min \quad f = \min\{C_{\max}\} = \min\{\max\{C_{(i)}\}\} \tag{1}$$

$$\text{s.t.} \quad t^S_{[a](i)} \geq t^S_{[a](ij)} + p_{[a](ij)}; \quad a = 1, 2, \ldots, m_{(i)}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, n_{(i)}; \tag{2}$$

$$t^S_{[a](ij)} \geq t^S_{[a](ijk)} + p_{[a](ijk)}; \quad a = 1, 2, \ldots, m_{(ij)}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, n_{(i)}; \quad k = 1, 2, \ldots, n_{(ij)}; \tag{3}$$

$$t^S_{[a](i)} + p_{[a](i)} \leq t^S_{[b](i)} + M \cdot (1 - q_{[a][b](i)}); \quad a, b = 1, 2, \ldots, m_{(i)}; \quad i = 1, 2, \ldots, n; \tag{4}$$

$$t^S_{[a](ij)} + p_{[a](ij)} \leq t^S_{[b](ij)} + M \cdot \left(1 - q_{[a][b](ij)}\right); \quad a, b = 1, 2, \ldots, m_{(ij)}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, n_{(i)}; \tag{5}$$

$$t^S_{[a](ijk)} + p_{[a](ijk)} \leq t^S_{[b](ijk)} + M \cdot \left(1 - q_{[a][b](ijk)}\right); \quad a, b = 1, 2, \ldots, m_{(ijk)}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, n_{(i)}; \quad k = 1, 2, \ldots, n_{(ij)}; \tag{6}$$

$$t^S_{[g](i)} + p_{[g](i)} \leq t^S_{[g](j)} + M \cdot \left(1 - q_{[g]((i)(j))}\right); \quad g = 1, 2, \ldots, m_{(i)}; \quad i, j = 1, 2, \ldots, n; \tag{7}$$

$$t^S_{[g](ij)} + p_{[g](ij)} \leq t^S_{[g](ik)} + M \cdot \left(1 - q_{[g]((ij)(ik))}\right); \quad g = 1, 2, \ldots, m_{(ij)}; \quad i = 1, 2, \ldots, nj, \quad k = 1, 2, \ldots, n_{(i)}; \tag{8}$$

$$t^S_{[g](ijk)} + p_{[g](ijk)} \le t^S_{[g](ijl)} + M \cdot \left(1 - q_{[g]((ijk)(ijl))}\right); \quad g = 1,2,\ldots,m_{(ijk)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \quad k,l = 1,2,\ldots,n_{(ij)}; \tag{9}$$

$$t^S_{[g](i)} + p_{[g](i)} \le t^S_{[g](jk)} + M \cdot \left(1 - x_{[g]((i)(jk))}\right); \quad i,j = 1,2,\ldots,n; \quad k = 1,2,\ldots,n_{(i)}; \quad g \in \{1,2,\ldots,m_{(i)}\} \cap \left\{1,2,\ldots,m_{(jk)}\right\}; \tag{10}$$

$$t^S_{[g](jk)} + p_{[g](jk)} \le t^S_{[g](i)} + M \cdot \left(1 - x_{[g]((jk)(i))}\right); \quad i,j = 1,2,\ldots,n; \quad k = 1,2,\ldots,n_{(i)}; \quad g \in \{1,2,\ldots,m_{(i)}\} \cap \left\{1,2,\ldots,m_{(jk)}\right\}; \tag{11}$$

$$t^S_{[g](i)} + p_{[g](i)} \le t^S_{[g](jkl)} + M \cdot \left(1 - x_{[g]((i)(jkl))}\right); \quad i,j = 1,2,\ldots,n; \quad k = 1,2,\ldots,n_{(i)}; \quad l = 1,2,\ldots,n_{(jk)}; \quad g \in \{1,2,\ldots,m_{(i)}\} \cap \left\{1,2,\ldots,m_{(jkl)}\right\}; \tag{12}$$

$$t^S_{[g](jkl)} + p_{[g](jkl)} \le t^S_{[g](i)} + M \cdot \left(1 - x_{[g]((jkl)(i))}\right); \quad i,j = 1,2,\ldots,n; \quad k = 1,2,\ldots,n_{(i)}; \quad l = 1,2,\ldots,n_{(jk)}; \quad g \in \{1,2,\ldots,m_{(i)}\} \cap \left\{1,2,\ldots,m_{(jkl)}\right\}; \tag{13}$$

$$t^S_{[g](ij)} + p_{[g](ij)} \le t^S_{[g](kls)} + M \cdot \left(1 - x_{[g]((ij)(kls))}\right); \quad i,k = 1,2,\ldots,n; \quad j,l = 1,2,\ldots,n_{(i)}; \quad s = 1,2,\ldots,n_{(ij)}; \quad g \in \left\{1,2,\ldots,m_{(ij)}\right\} \cap \{1,2,\ldots,m_{(kls)}\}; \tag{14}$$

$$t^S_{[g](kls)} + p_{[g](kls)} \le t^S_{[g](ij)} + M \cdot \left(1 - x_{[g]((kls)(ij))}\right); \quad i,k = 1,2,\ldots,n; \quad j,l = 1,2,\ldots,n_{(i)}; \quad s = 1,2,\ldots,n_{(ij)}; \quad g \in \left\{1,2,\ldots,m_{(ij)}\right\} \cap \{1,2,\ldots,m_{(kls)}\}; \tag{15}$$

$$t^S_{[a](i)} \ge 0; \quad a = 1,2,\ldots,m_{(i)}; \quad i = 1,2,\ldots,n; \tag{16}$$

$$t^S_{[a](ij)} \ge 0; \quad a = 1,2,\ldots,m_{(ij)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \tag{17}$$

$$t^S_{[a](ijk)} \ge 0; \quad a = 1,2,\ldots,m_{(ijk)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \quad k = 1,2,\ldots,n_{(ij)}; \tag{18}$$

$$x_{[g]((i)(j))} + x_{[g]((j)(i))} = 1, \quad \text{where } x_{[g]((i)(j))} = 0 \text{ or } 1; \quad g = 1,2,\ldots,m_{(i)}; \quad i,j = 1,2,\ldots,n; \tag{19}$$

$$x_{[g]((ij)(ik))} + x_{[g]((ik)(ij))} = 1, \quad \text{where } x_{[g]((ij)(ik))} = 0 \text{ or } 1; \quad g = 1,2,\ldots,m_{(ij)}; \quad i = 1,2,\ldots,n; \quad j,k = 1,2,\ldots,n_{(i)}; \tag{20}$$

$$x_{[g]((ijk)(ijl))} + x_{[g]((ijl)(ijk))} = 1, \quad \text{where } x_{[g]((ijk)(ijl))} = 0 \text{ or } 1; \quad g = 1,2,\ldots,m_{(ijk)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \quad k,l = 1,2,\ldots,n_{(ij)}; \tag{21}$$

$$x_{[g]((i)(jk))} + x_{[g]((jk)(i))} = 1 \quad (i) \ne (j), \quad \text{where } x_{[g]((i)(jk))} = 0 \text{ or } 1; \quad i,j = 1,2,\ldots,n; \quad k = 1,2,\ldots,n_{(i)}; \quad g \in \{1,2,\ldots,m_{(i)}\} \cap \left\{1,2,\ldots,m_{(jk)}\right\}; \tag{22}$$

$$x_{[g]((i)(jkl))} + x_{[g]((jkl)(i))} = 1 \quad (i) \ne (j), \quad \text{where } x_{[g]((i)(jkl))} = 0 \text{ or } 1; \quad i,j = 1,2,\ldots,n; \quad k = 1,2,\ldots,n_{(i)}; \quad l = 1,2,\ldots,n_{(jk)}; \quad g \in \{1,2,\ldots,m_{(i)}\} \cap \left\{1,2,\ldots,m_{(jkl)}\right\}; \tag{23}$$

$$x_{[g]((ij)(kls))} + x_{[g]((kls)(ij))} = 1 \quad (ij) \ne (kl), \quad \text{where } x_{[g]((ij)(kls))} = 0 \text{ or } 1; \quad i,k = 1,2,\ldots,n; \quad j,l = 1,2,\ldots,n_{(i)}; \quad s = 1,2,\ldots,n_{(ij)}; \quad g \in \left\{1,2,\ldots,m_{(ij)}\right\} \cap \{1,2,\ldots,m_{(kls)}\}. \tag{24}$$

The definition of the parameters in the model is as follows.

FS represents the feasible solution set; $o_{[a](i)}$ represents the ath operation of job assembly (i), $o_{[a](ij)}$ represents the ath operation of job subassembly (ij); $o_{[a](ijk)}$ represents the ath operation of job part (ijk); $p_{[a](i)}$ represents the operation time of $o_{[a](i)}$; $p_{[a](ij)}$ represents the operation time of $o_{[a](ij)}$; $p_{[a](ijk)}$ represents the operation time of $o_{[a](ijk)}$; M is an infinite positive multiply factor:

$$q_{[a][b](i)} = \begin{cases} 1, & \text{if the operation of assembly } (i) \text{ on machine } [b] \\ & \quad \text{is exactly after its operation on machine } [a], \\ 0, & \text{otherwise,} \end{cases}$$

$$q_{[a][b](ij)} = \begin{cases} 1, & \text{if the operation of subassembly } (ij) \text{ on} \\ & \quad \text{machine } [b] \text{ is exactly after its operation} \\ & \quad \text{on machine } [a], \\ 0, & \text{otherwise,} \end{cases}$$

$$q_{[a][b](ijk)} = \begin{cases} 1, & \text{if the operation of part } (ijk) \text{ on machine } [b] \\ & \quad \text{is exactly after its operation on machine } [a], \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

The definition of the decision variables in the model is as follows.

$t^S_{[a](i)}$ represents the start time of operation $o_{[a](i)}$, $t^S_{[a](ij)}$ represents the start time of operation $o_{[a](ij)}$, $t^S_{[a](ijk)}$ represents the start time of operation $o_{[a](ijk)}$ and $C_{(i)}$ represents the end time of assembly (i); thus we have

$$C_{(i)} = \max_{a\in 1,2,\ldots,m_{(i)}} \left(t^S_{[a](i)} + p_{[a](i)}\right), \tag{26}$$

and the 0-1 programming variables are defined as

$$x_{[g](i)(j)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{assembly } (i) \text{ on machine } [g] \text{ is } (j), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](ij)(ik)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{subassembly } (ij) \text{ on machine } [g] \text{ is } (ik), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](ijk)(ijl)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{part } (ijk) \text{ on machine } [g] \text{ is } (ijl), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](i)(jk)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{assembly } (i) \text{ on machine } [g] \text{ is} \\ & \quad \text{subassembly } (jk), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](jk)(i)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{subassembly } (jk) \text{ on machine } [g] \text{ is} \\ & \quad \text{assembly } (i), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](i)(jkl)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{assembly } (i) \text{ on machine } [g] \text{ is} \\ & \quad \text{part } (jkl), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](jkl)(i)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{part } (ijk) \text{ on machine } [g] \text{ is assembly } (i), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](ij)(kls)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{subassembly } (ij) \text{ on machine } [g] \text{ is part } (kls), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](kls)(ij)} = \begin{cases} 1, & \text{if the immediate successive operation of part} \\ & \quad (kls) \text{ on machine } [g] \text{ is subassembly } (ij), \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

The physical explanation for the model (shown as (1)–(24)) is as follows.

Equation (1) gives the optimization objective as makespan; (2) constrains that the start time of an assembly must be later than the completion time of all its subassemblies; (3) constrains that the start time of a subassembly

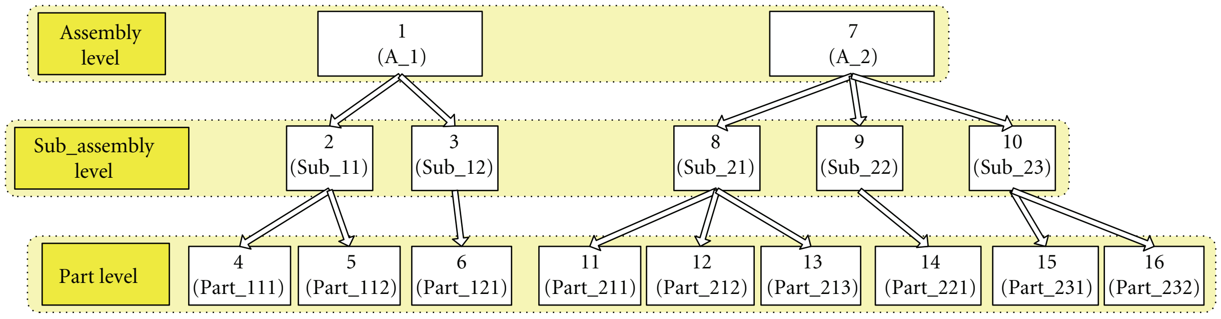


FIGURE 2: Example of a 3-level production scheduling problem.

must be later than the completion time of all its parts; (4) defines the precedence relationship of assembly (i)'s two successive operations $o_{[a](i)}$ and $o_{[b](i)}$; that is, the start time of operation $o_{[b](i)}$ must be later than the end time of operation $o_{[a](i)}$ if $q_{[a][b](i)} = 1$, and vice versa; (5) defines the precedence relationship between subassembly (ij)'s two successive operation $o_{[a](ij)}$ and $o_{[b](ij)}$; (6) defines the precedence relationship between part (ijk)'s two successive operation $o_{[a](ijk)}$ and $o_{[b](ijk)}$; (7)–(15) define that there can only process at most one job on machine $[g]$ at one time; (16)–(18) constrain the start time of each operation as a non-negative variable; (19)–(21) define the constraints between each related-pair of decision variables in each level to guarantee the feasibility of the solution; (22)–(24) define the constraints between each pair of decision variables in different levels.

4. Twin-Screw-Coded Evolutionary Algorithm for Multilevel Production Scheduling Optimization

Since the computational complexity of the multi-level production scheduling problem is very high, it is hard to solve it with current existing optimization methods efficiently (either the precise methods or the problem-dependent heuristic algorithms). In order to overcome the drawbacks of current existing methods, a twin-screw-coded evolutionary algorithm is proposed in this paper, which encodes a possible multi-level scheduling scheme in a twin-screw chromosome, and a metaheuristic-based population gap for elitist exchange and local search is employed to enhance the convergence of the algorithm.

A typical 3-level MLPS example is shown in Figure 2, which includes two assembles (1,7), five subassemblies (1's subassemblies include 2, 3; 7's subassemblies include 8, 9, 10) and nine parts (2's parts are 4, 5; 3's part is 6; 8's parts are 11, 12, 13; 9's part is 14; 10's parts are 15, 16). The hierarchical process precedence relationship between the jobs can be easily told from the figure, in which the process scheduling optimization covers all 3-levels' jobs at the same time.

4.1. Gene Expression: Twin-Screw Coding. Current existing evolutionary coding can only express one single level's scheduling information; it is hard to employ them to express a multi-level production scheduling information with one

15	1	5	7	7	12	4	16	3	5	2	3	14	11	8	12	13	6	5	9	10
3	*1*	*3*	*1*	*1*	*3*	*3*	*3*	*2*	*3*	*2*	*2*	*3*	*3*	*2*	*3*	*3*	*3*	*3*	*2*	*2*

FIGURE 3: The feasible gene expression for a three-level shop scheduling solution.

chromosome. Hence, we propose an operation-based twin-screw coding strategy to solve this problem: we define each job number with an implicit subsidiary coding (subcoding) to label its level information and construct a twin-screw module to express the hierarchy of the scheduling scheme. Assume that, in the example of Figure 2, each job in $\{3, 7, 12\}$ has two operations, respectively, job 5 has three operations, and all the other jobs have only one operation, which is a typical mixed flexible job shop problem. A feasible solution for this problem can be coded as a twin-screw gene (shown as Figure 3), in which the *italic* gene in the second row shows the level of the job above it. Thus, we can construct a feasible chromosome for the 3-level job process scheduling problem with the twin-screwed gene expression.

In order to guarantee the feasibility of a twin-screw-coded chromosome, a specifically designed decoding strategy (shown as Figure 4) is employed to cope with this issue. We employ the example mentioned above to illustrate the process of our proposed decoding method, in which process scheduling information is obtained from lower level to the higher one. The genotype codes of a chromosome are scanned from the beginning to the end; those operations that labeled with a "3" as its implicit twin-code (in the second line) are *Recognized* as part level's jobs and labeled to phenotype; after a round-robin scan, all the jobs belonging to the part level are kicked off from the original chromosome, a second round scan is taken to label the subassembly level, and then the assembly level, until all the levels' operations are labeled and the whole chromosome is decoded to a complete 3-level job scheduling solution with phenotype status.

In the decoding process (Figure 4), $^{o5}_{1}$ represents the first operation of job 5; $o4$ represents the operation of job 4. The final decoded solution of the above example is as follows:

(i) part level: $o15$, $^{o5}_{1}$, $o12$, $o4$, $o16$, $^{o5}_{2}$, $o14$, $o11$, $o12$, $o13$, $o06$, $^{o5}_{3}$;

(ii) subassembly level: $^{o3}_{1}$, $o2$, $^{o3}_{2}$, $o8$, $o9$, $o10$;

(iii) assembly level: $o1$, $^{o7}_{1}$, $^{o7}_{2}$.

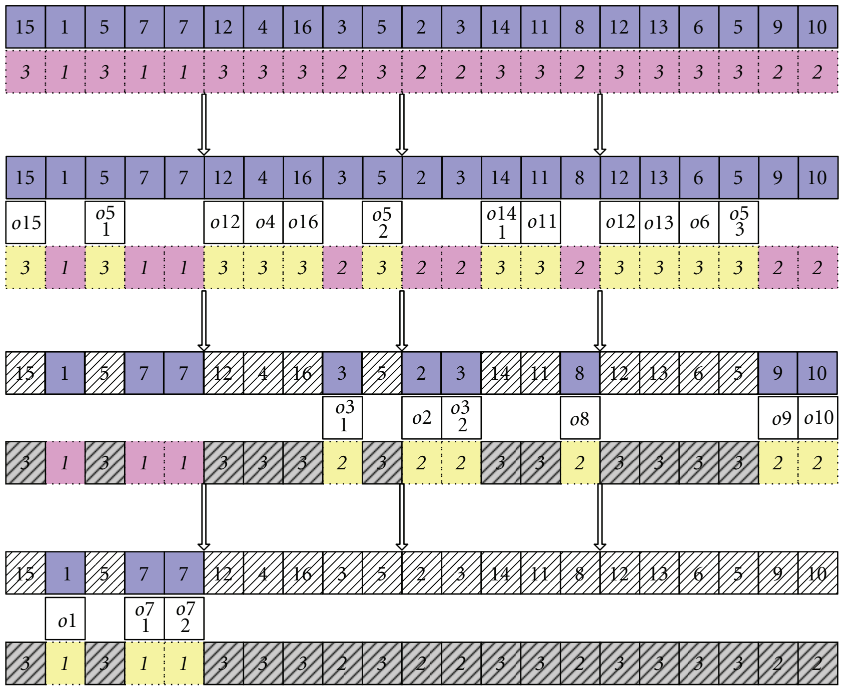


FIGURE 4: A decoding example.

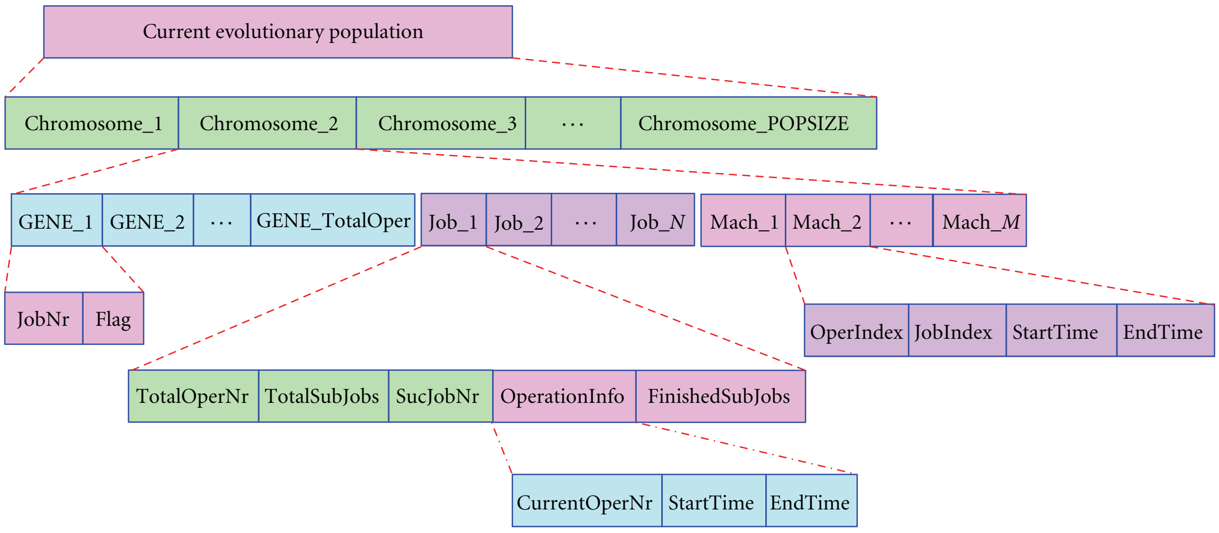


FIGURE 5: The data structure for decoding process.

In order to promote the decoding efficiency, an object-oriented data structure is designed to fulfill this task (Figure 5). The principle of designing the data structure and some variable abbreviations are noted as the follows.

(1) A chromosome composes three parts: twin-screw codes for sequence and "level" information, job-related information, and machine-related information.

(2) "GENE_i" represents the twin-screw coding structure of a chromosome; "JobNr" represents the job number (the number of each job is equal to its operation number); The initial "Flag" of job "i" indicates the production level, which job "i" belongs to.

During the decoding process, when all the operations of one subjob of job "i" are finished, we add "Job_i"'s "FinishedSubJobs" by (1): when the value is equal to "i"'s "TotalSubJobs", that means all the subjobs of "i" have been finished we turn the "Flag" of "i" from 1 to 0 under this circumstance, and return the scan pointer into the beginning of the twin-screw, and rescan the left operations.

(3) "OperationInfo" records the start time and end time of each job.

(4) "TotalOperNr", "TotalSubJobs", and "SubJobNr" are the initial status of the jobs; these information can be obtained from the configuration document.

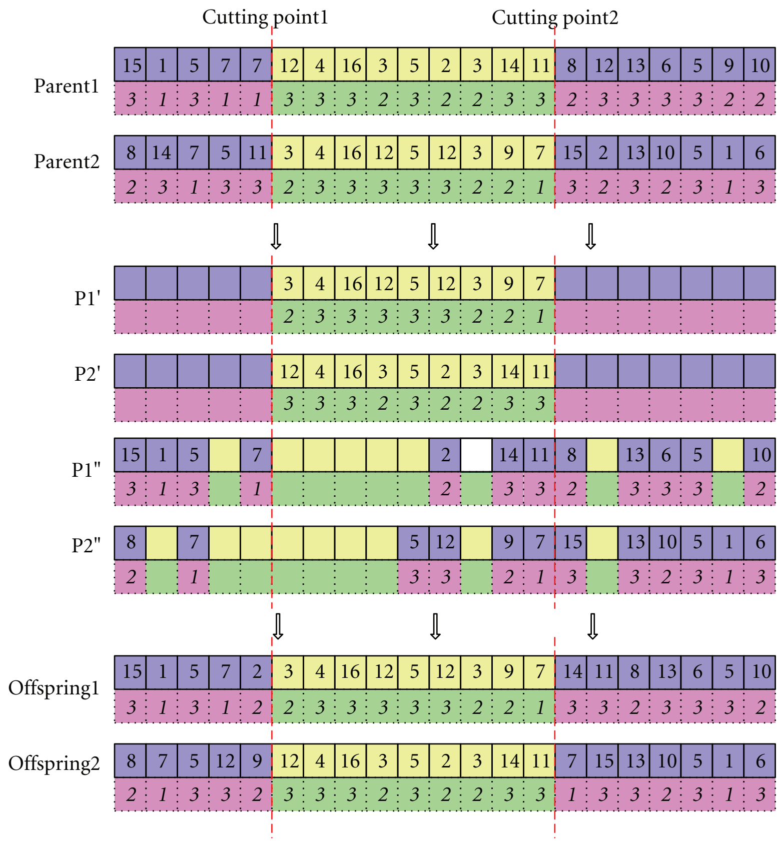


Figure 6: The crossover operator: an example.

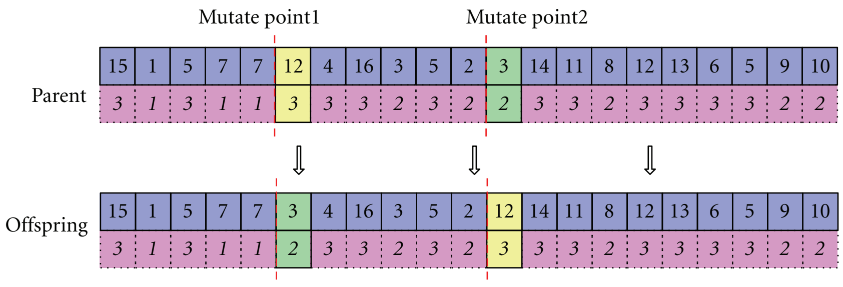


Figure 7: The mutation operator: an example.

(5) “Mach_i” records the job sequences at machine “i”; these pieces of information are employed to verify the validity of a solution, and to make preparation data for Gantt graph, while not participating the decoding process.

4.2. Evolutionary Operators: Crossover and Mutation. In order to guarantee the feasibility and validity of the chromosome during the population evolution, we propose improved crossover and mutation operators for the twin-screw-coded chromosome based on previous work in permutation scheduling problem studies (see [22]). The principle of designing the evolutionary operators is similar to that of PMX crossover and swap mutation operators (see [22]), which can be illustrated by examples shown in Figures 6 and 7.

Crossover operator for a twin-screw-coded chromosome is as follows.

Step 1. Randomly generate two cutting points c_1, c_2 (assume $c_1 < c_2$) on the two parents chromosome $chrom_1$, $chrom_2$.

Step 2. Exchange the partial chromosome between c_1 and c_2 (not only the process sequence information but also the “level” information) to get the two proto-children, shown as P_1' and P_2' in Figure 6.

Step 3. Scan and eliminate the existing elements of P_1' from parent 1 and P_2' from parent 2 to determine the mapping relationship between two mapping sections. After the mapping operation, we get two subsidiary partial mapping information as the P_1'', P_2'' shown in Figure 6.

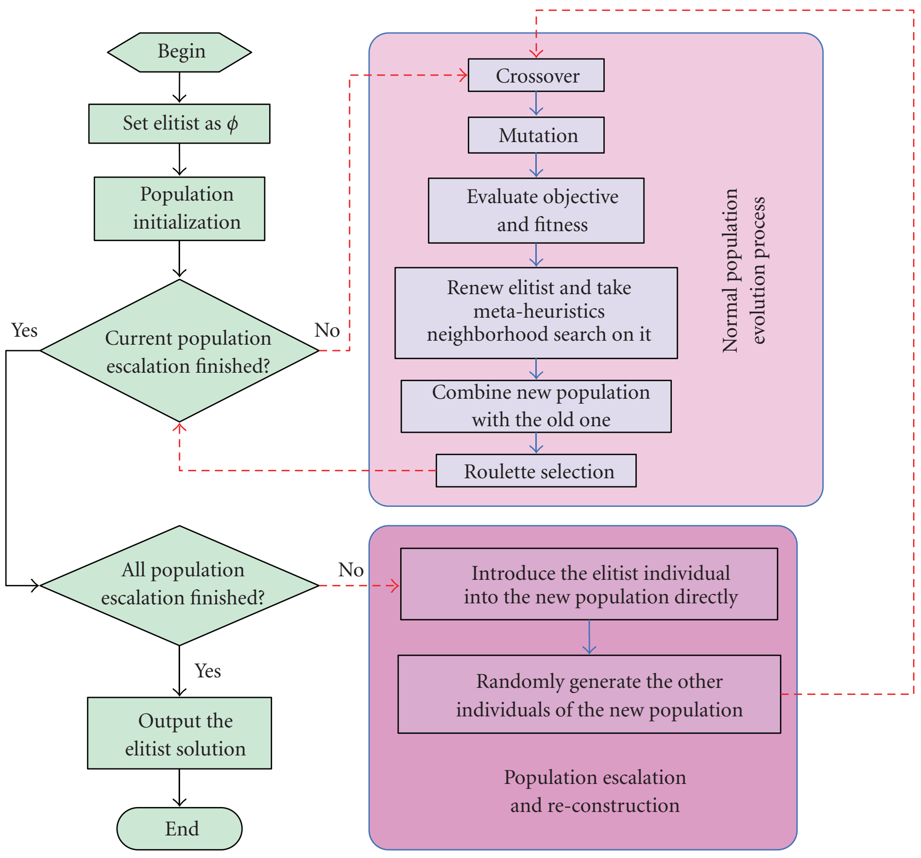

Begin
Set elitist as ϕ
Population initialization
Current population escalation finished?
Yes
No
Crossover
Mutation
Evaluate objective and fitness
Renew elitist and take meta-heuristics neighborhood search on it
Combine new population with the old one
Roulette selection
Normal population evolution process
All population escalation finished?
No
Yes
Introduce the elitist individual into the new population directly
Randomly generate the other individuals of the new population
Population escalation and re-construction
Output the elitist solution
End

FIGURE 8: Flowchart of the Twin-screw Coded Evolutionary Algorithm (TCEA) for MLPS.

Step 4. Legalize the offspring with the mapping relationship information, and obtain two feasible offspring children (*offspring1* and *offspring2* in Figure 6).

Please note that the whole crossover process is guided and dominated by the sequence information, but not the "level" information. Because in our proposed twin-screw-coded chromosome, each bit of "level" information is bound strength with a corresponding operation information, it does not make any sense except in the decoding process.

Mutation operator for twin-screw-coded chromosome is as follows.

Step 1. Randomly generate two mutate points m_1, m_2 on the parent chromosome to be mutated.

Step 2. Swap the two position's sequence and "level" information to generate a legal offspring child (as shown in Figure 7).

4.3. Population Reconstruction with Elitism Strategy. In our previous work, we have proposed an escalating evolutionary structure (shown in Figure 8), which has outperformed several other modern heuristic algorithms with applications to flow shop scheduling problems under the similar computational cost. In order to solve the integrated MLPS model efficiently, we introduce the escalating strategy into the twin-screw-coded EA to enhance its convergence performance. The brief idea of escalating strategy can be explained as follows.

A population evolves from a random beginning status, the probability of an individual to bring its offspring lies on its fitness. After some generations' evolution, the population may keep in evolving with no progress further more in some successive generations; then the elitist individual (the best one in the population from the beginning to current generation) is kept and introduced into a new population directly, and all the other individuals of the new population are generated randomly (reconstruction/reinitialization). Thus, the new population continues the evolution process until the stop criterion is satisfied.

The escalation process implies two meanings: (1), the elitist individual will be introduced into the new population directly; (2), other individuals of the new population will be generated randomly, where (1) makes it possible to utilize the previous level's search information, and (2) is designed to keep the population diversity, which helps the algorithm escaping from premature.

5. Case Study

5.1. Problem Description. In order to evaluate the performance of our proposed modeling and optimization technique, a 3-level production scheduling problem from one of Chinese famous satellite production factory is employed

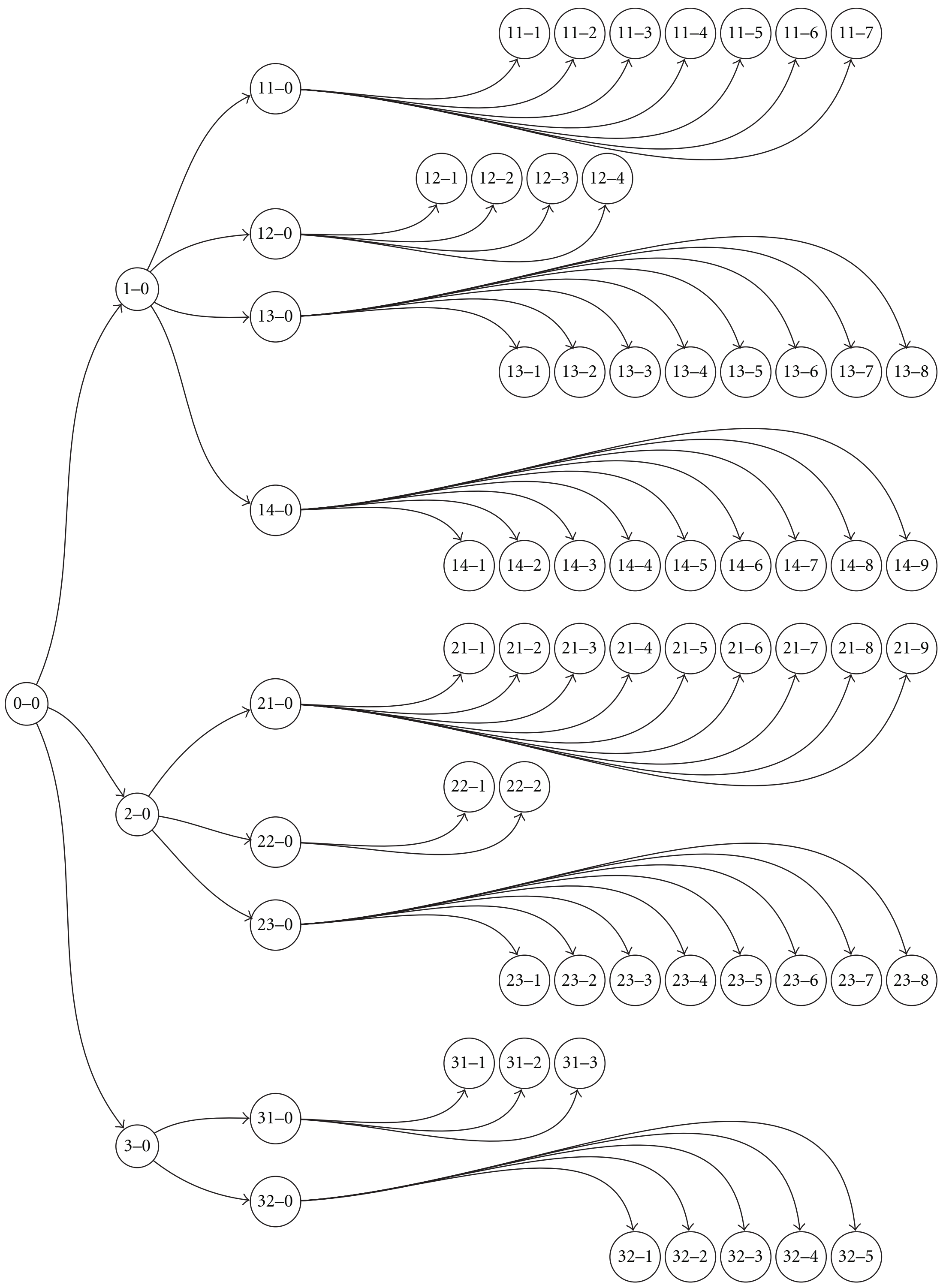


Figure 9: Problem description.

for case study. All the processing information has been necessarily deposed with pre-declassification need beforehand.

The hierarchical product structure of the problem is similar to the model described as (1)–(24): product 0–0 consists of 3 assemblies, each assembly consists of 4, 3, 2 subassemblies, respectively, and each subassembly consists of some given number of parts. The job of a higher level could begin its process operation only if all its sub-products are finished and are assembled. The process information includes technical constraints within levels and between levels, process time, and predefined operational sequence between jobs. The hierarchical logic relationship of the problem can be highlighted as shown in Figure 9.

Table 1: The raw data of case study problem.

	Oper.1		Oper.2		Oper.3		Oper.4		
JobNr	machine	time	machine	time	machine	time	machine	time	· · ·
11–1	O1	16	O4	7.1	O1	16	DM	38	· · ·
11–2	O1	8	RM	5.3	O3	7	PM	10	· · ·
11–3	O2	20	DM	44.8	O2	11.2	PM	80	· · ·
11–4	RM	3.3	O3	0.4	PM	4.6	O1	8	· · ·
11–5	O2	10.4	DM	44.2	O1	16	O2	12	· · ·
11–6	O2	5	DM	14	RM	2	O1	8	· · ·
11–7	O2	20	DM	56	RM	33.2	O3	2	· · ·
11–0	RM	31	O7	20.3	O1	16	PM	12	· · ·
12–1	RM	29	DM	20	O1	8	RM	22	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
12–5	O7	7	RM	11	O1	8	O6	12	· · ·
12–0	RM	27	O2	12					
13–1	O1	8	O4	19.5	RM	21.2	O4	2	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
13–8	O7	21	RM	33	O1	8	O6	22	· · ·
13–0	RM	36	OM	54	PM	21	O5	8	
14–1	RM	9	O9	60	O5	6			
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
14–9	O2	2.4	O1	8	DM	12.4	O3	2	· · ·
14–0	O2	9	O2	5	O1	8	O2	6	· · ·
1–0	O3	82	O2	14	RM	61	O3	24	· · ·
21–1	O7	27.2	O7	10.4	O1	64	O6	144	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
21–9	RM	1152	OM	768	PM	768	O5	384	· · ·
21–0	O3	208	RM	136	DM	84	O5	32	· · ·
22–1	RM	61.5	DM	170	O1	40	DM	720	· · ·
22–2	RM	19	DM	60	O1	16	DM	248	· · ·
22–0	O3	23	RM	54					
23–1	O11	8.5	DM	5.2	O1	8	O9	7	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
23–8	O11	8.5	DM	5.2	O1	8	O9	7	· · ·
23–0	O3	43	DM	44	O3	4			
2–0	O3	84	DM	20	O9	32	DM	88	· · ·
31–1	O2	4.6	RM	4	DM	6.4	O3	3	· · ·
31–2	O2	8	O1	8	O2	3.3	DM	16	· · ·
31–3	O2	8	OM	4.4	O5	8	O3	0.5	· · ·
31–0	O2	32	OM	58	O1	8	O2	28	· · ·
32–1	O2	20	OM	12	O3	9.3			
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
32–5	O4	4	DM	1	O3	1			
32–0	O2	5.2	O2	5.2	RM	116	O13	6	· · ·
3–0	O13	40	DM	52	O3	8			
0–0	O3	32	OM	81	O3	23	OM	68	· · ·

Table 1 gives the detail process information of the problem. There are some complementary comments to the problem.

(1) The *O*1–*O*13 in Table 1 represent the process operations on those machines, whose process ability can be greatly increased with a bit costs and the workloads on them can be considered as light as possible. So the capability of these machines can be considered as infinite. These machines include common low-precise manufacturing machines, like lathe, planer, grinder, and so forth.

While the workloads on the other kind of machines are obviously heavy, not only the fixed expensive purchasing costs but also the expensive unit time process costs on them are much higher than the common ones. There are 4 units of such machines in the factory that we investigated, whose name can be listed as Rough Milling machine (RM), Precise Milling machine (PM), Digital Milling machine (DM), and Other Milling machine (OM), respectively.

(2) The definition of job numbers (JobNr) in Table 1 is as follows.

(i) 11–01 represents the 01 part of subassembly 11;

(ii) 11–0 is the label for subassembly 11;

(iii) 1–0 is the label for assembly 1;

(iv) 0–0 is the label of the final product.

After analyzing the process information, we make two assumptions to deduce the computational complexity of the raw problem, in which only the most "expensive" and "crowed" machines are specifically treated, while we neglect the scheduling planing on those "cheap" or "loosely required" machines.

Assumption 1 (Machine Scarce/Nonscarce Assumption). As we know, a satellite product requires more precision than a civil product; its large size and highly precision quality leads to the need of high-performance milling machines in many operations. We discriminately treat the operations that need to be operated on milling machines with those operations that need to be operated on other machines and call the operations on these two kind of machines as *Scarce machine* operations and *nonscarce machine* operations. Consecutive operations on *nonscarce machines* can be combined into "Dummy Operations".

The adjacent operations on *nonscarce machines* can be combined as one operation time, and the new combined operation can be considered as operations that have no constraints on machine availability; we only take its operation time into account but neglect the operation's machine information.

Assumption 2 (Machine Specification Assumption). As different machines for the same type of task in the satellite factory take different costs, the milling operations are allocated to different machines according to its precision requirements, which aims at strengthening the economic profit of the whole. After the hypothesis of dummy operation and machine operation specialization, we focus our effort on the scheduling of scarce machines, which can help us avoid to waste time on insignificant operations or wasting costly machines on simple or nonprecise operations; thus can we possibly obtain a better solution in a given time.

5.2. Data Preprocessing. There are 4 milling machines (RM/PM/DM/OM) in the factory. Since the operations on these milling machines are the bottle neck of the scheduling problem, we allocate the milling operations to these machines with regard to each machine's operational precision and cost.

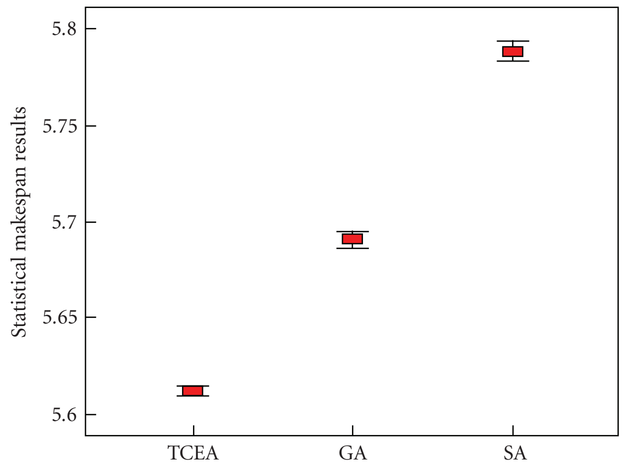


Figure 10: Statistical results of 3-level MLPS with TCEA, GA, and SA.

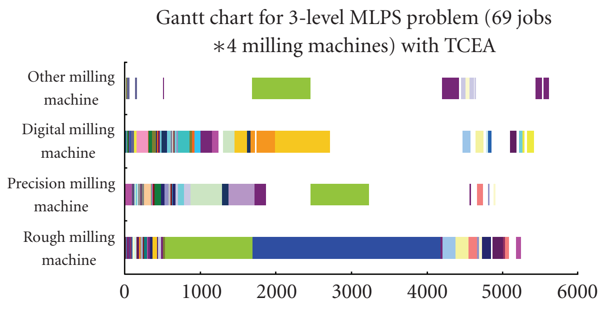


Figure 11: A typical solution of 3-level MLPS with TCEA.

(i) Rough milling operations are allocated to the machine "RM".

(ii) Precise milling operations are allocated to the machine "PM".

(iii) Digital milling operations are allocated to the machine "DM".

(iv) Other milling operations are allocated to the machine "OM".

All the other nonmilling operations are considered as "dummy operations" (as mentioned in Section 5.1). After we combined the "dummy operations", we get the new modified process data of the problem (shown in Table 2).

5.3. Comparison Study Algorithms and Parameters Setting. In order to evaluate the performance of our proposed TCEA with its application to MLPS, we employ two basic metaheuristic algorithms, GA (Genetic Algorithm), and SA (Simulated Annealing) as comparison algorithms for case study.

In order to obtain the best performance of TCEA, parameters experiment has been taken to search the best parameters combination. The experiment is designed with the following guidance rules [21].

Table 2: The modified data of case study problem.

Level 1: the part level

JobNr	dum	mach	time	dum	mach	time	dum	mach	time	· · ·
11–01	32.1	DM	38	26	NS	0				
11–02	8	RM	5.3	7	PM	10	8	DM	80	· · ·
11–03	20	DM	44.8	11.2	PM	80	4.8	NS	0	· · ·
11–04	0	RM	3.3	0.4	PM	4.6	8	PM	23	· · ·
11–05	10.4	DM	44.2	28	PM	80	4.4	NS	0	· · ·
11–06	5	DM	14	0	RM	2	11.5	DM	8	· · ·
11–07	20	DM	56	0	RM	33.2	14	DM	32	· · ·
12–01	0	RM	29	0	DM	20	8	RM	22	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
12–05	7	RM	11	20	PM	52	1	DM	19	· · ·
13–01	27.5	RM	21.2	2	PM	11	18.4	NS	0	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
13–08	21	RM	33	30	PM	52	13	DM	19	· · ·
14–01	0	RM	9	66	NS	0				
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
14–09	10.4	DM	12.4	2	NS	0				
21–01	245.6	PM	416	10.4	DM	152				
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
21–09	0	RM	1152	0	OM	768	0	PM	768	· · ·
22–01	0	RM	61.5	0	DM	170	40	DM	720	· · ·
22–02	0	RM	19	0	DM	60	16	DM	248	· · ·
23–01	8.5	DM	5.2	19.5	DM	13	8	DM	32	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
23–08	8.5	DM	5.2	19.5	DM	13	8	DM	32	· · ·
31–01	4.6	RM	4	0	DM	6.4	3	NS	0	· · ·
31–02	19.3	DM	16	2.5	NS	0				
31–03	8	DM	4.4	8.5	NS	0				
32–01	20	OM	12	9.3	NS	0				
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
32–05	4	DM	1	1	NS	0				

Level 2: the subassembly level

JobNr	dum	mach	time	dum	mach	time	dum	mach	time	· · ·
11–0	0	RM	31	36.3	PM	12	16	NS	0	· · ·
12–0	0	RM	27	12	NS	0				
13–0	0	RM	36	0	OM	54	0	PM	21	· · ·
14–0	29	DM	29							
21–0	208	RM	136	0	DM	84	32	NS	0	· · ·
22–0	23	RM	54							
23–0	43	DM	44	4	NS	0				
31–0	32	OM	58	36	OM	60	8	OM	12	· · ·
32–0	10.4	RM	116	23	NS	0				

Level 3: the assembly level (including the final product)

JobNr	dum	mach	time	dum	mach	time	dum	mach	time	· · ·
1–0	96	RM	61	24	NS	0				
2–0	84	DM	20	32	DM	88	16	NS	0	· · ·
3–0	40	DM	52	8	NS	0				
0–0	32	OM	81	23	OM	68				

Table 3: Parameters setting for 3-level MLPS study case.

	pop_size	max_gens	P_c	P_m	n_L
TCEA	200	200×5	0.9	0.1	20
GA	200	1000	0.9	0.1	
	T_0	T_{final}	β	n_L	
SA	1000	0.1	0.999	50	

Table 4: Statistical optimization results of the 3-level MLPS with TCEA, GA and SA.

Algorithms	avg.Makespan	max.Makespan	min.Makespan	dev.Makespan
TCEA	5612.4	5632	5598	3.86
GA	5691.2	5704	5668	8.61
SA	5788.5	5814	5768	11.26

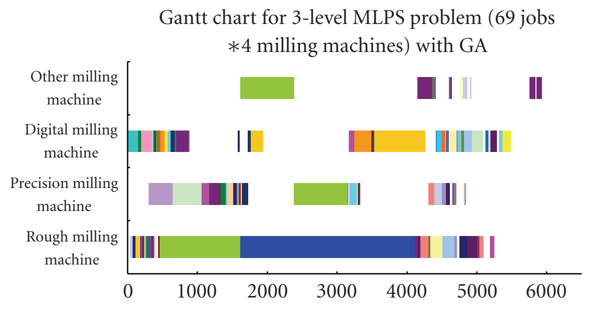


Figure 12: A typical solution of 3-level MLPS with GA.

(i) Population size (POP_SIZE) varies from 100 to 500, skip rule 100.

(ii) Evolutionary generation varies from 100 to 500, skip rule 100.

(iii) Crossover probability varies from 0.3 to 0.9, skip rule 0.1.

(iv) Mutation probability varies from 0.01 to 0.1, skip rule 0.01.

(v) Population escalation gap varies from 10 to 1, skip rule 1.

(vi) Elitist local search step varies from 10 to 50, skip rule 10.

After the parameters experiment, we set the parameters of TCEA as in Table 3. In order to compare the performance of TCEA with that of GA and SA in a fair circumstance, we make the similar parameters experiments for GA and SA, respectively, in which the total CPU time consumption is kept in the same level as TCEAs. After the experiments, we can set the parameters of TCEA, GA, and SA as in Table 3. Since the twin-screw coding strategy is a general encoding strategy designed for MLPS problems, we employ the coding strategy in all the three algorithms.

5.4. Results Analysis. Since all the algorithms that we study are metaheuristic algorithms, we run each algorithm for 20 independent times to collect their statistical results. With parameters set in Table 3, we get the optimization results as in Table 4 and figure 10, in which the average result of TCEA outperforms that of GA and SA:

The average makespan of product 0–0 obtained by our TCEA is 5612.4, and the results of GA and SA are 5691.2 and 5788.5, respectively; all these metaheuristic algorithms outperformed current technique in the factory (6580).

However, the computational cost of TCEA (about 270s) is a bit longer than that of GA (about 250s) and SA (about 190s) in the same experiment environment (all the experiments are taken in a CPU Pentium IV-3.2 G, 1 G Ram PC platform).

In general, the statistical results show the outstanding performance of our TCEA by contrast to that of GA and SA to cope with an MLPS problem.

Figures 11 and 12 show two typical solutions derived from TCEA and GA, respectively.

6. Conclusion

A twin-screw-coded evolutionary algorithm (TCEA), which is motivated by solving a typical multi-level production scheduling problem (MLPS), is put forward in this paper. The principle of the new algorithm is introduced; besides, a real 3-level satellite part's case study has revealed the superiority of TCEA by contrast to GA and SA, which further demonstrates the effectiveness and practicability of our integrated model and optimization technique in solving such complex production scheduling problems. As we know, MLPS is a complex NP-hard problem; this work just shows the preliminary result of our project. Further research has been taken, in which multiobjective MLPS problem modeling and optimization technique has been taken into consideration.

Acknowledgments

The authors would like to thank the Natural Science Foundation of China (Grant nos. 70771003, 60974051), the Natural Science Foundation of Hebei Province (Grant no. F2010001714), the Doctorial Faculty Research Foundation of North China Electric Power University (Grant no. 200822025), and Shanghai Special Foundation of

Outstanding Young Teachers in University (Grant no. ssc08023) for their financial supports with this research. The authors would like to thank Dr. X.W. Tan and J. Wang for their helpful suggestions to improve this work, and thank Ms. S.P. Wang for her help in case study data collection. The authors would also like to thank the anonymous reviewers and Dr. C.K. Ting for their constructive comments to improve the work.

References

[1] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[2] N. Boysen, M. Fliedner, and A. Scholl, "Sequencing mixed-model assembly lines: survey, classification and model critique," *European Journal of Operational Research*, vol. 192, no. 2, pp. 349–373, 2009.

[3] Y. Li, K. F. Man, K. S. Tang, S. Kwong, and W. H. Ip, "Genetic algorithm to production planning and scheduling problems for manufacturing systems," *Production Planning and Control*, vol. 11, no. 5, pp. 443–458, 2000.

[4] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: an updated review," *Advanced Engineering Informatics*, vol. 20, no. 4, pp. 415–431, 2006.

[5] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Computers and Operations Research*, vol. 30, no. 8, pp. 1151–1171, 2003.

[6] L. Li, J. Y. H. Fuh, Y. F. Zhang, and A. Y. C. Nee, "Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments," *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 6, pp. 568–578, 2005.

[7] F. Riane, A. Artiba, and S. Iassinovski, "An integrated production planning and scheduling system for hybrid flowshop organizations," *International Journal of Production Economics*, vol. 74, no. 1–3, pp. 33–48, 2001.

[8] L. Tang, J. Liu, A. Rong, and Z. Yang, "A review of planning and scheduling systems and methods for integrated steel production," *European Journal of Operational Research*, vol. 133, no. 1, pp. 1–20, 2001.

[9] N. Vandaele and L. De Boeck, "Advanced resource planning," *Robotics and Computer-Integrated Manufacturing*, vol. 19, no. 1-2, pp. 211–218, 2003.

[10] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers and Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[11] J. Deschamps and J. Bourrieres, "Multi-level data model for load allocation to distributed manufacturing resources," in *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 357–362, New York, NY, USA.

[12] M. Pinedo, *Scheduling-Theory, Algorithms, and Systems*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.

[13] D. E. Shobrys and D. C. White, "Planning, scheduling and control systems: why cannot they work together," *Computers and Chemical Engineering*, vol. 26, no. 2, pp. 149–160, 2002.

[14] C. Moon and M. Gen, "Genetic algorithm-based approach for design of independent manufacturing cells," *International Journal of Production Economics*, vol. 60, pp. 421–426, 1999.

[15] A. Drexl and A. Kimms, "Lot sizing and scheduling—survey and extensions," *European Journal of Operational Research*, vol. 99, no. 2, pp. 221–235, 1997.

[16] B. Karimi, S. M. T. Fatemi Ghomi, and J. M. Wilson, "The capacitated lot sizing problem: a review of models and algorithms," *Omega*, vol. 31, no. 5, pp. 365–378, 2003.

[17] M.-J. Yao and J.-X. Huang, "Solving the economic lot scheduling problem with deteriorating items using genetic algorithms," *Journal of Food Engineering*, vol. 70, no. 3, pp. 309–322, 2005.

[18] E. Alvarez, "Multi-plant production scheduling in SMEs," *Robotics and Computer-Integrated Manufacturing*, vol. 23, no. 6, pp. 608–613, 2007.

[19] B. R. Sarker and C. V. Balan, "Operations planning for a multi-stage kanban system," *European Journal of Operational Research*, vol. 112, no. 2, pp. 284–303, 1999.

[20] Y.-N. Yang, H. R. Parsaei, and H. R. Leep, "Prototype of a feature-based multiple-alternative process planning system with scheduling verification," *Computers and Industrial Engineering*, vol. 39, no. 1-2, pp. 109–124, 2001.

[21] T. P. Bagchi, *Multiobjective Scheduling by Genetic Algorithms*, Kluwer Academic, Boston, Mass, USA, 1999.

[22] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, NY, USA, 1996.

[23] T. Murata, *Genetic algorithms for multi-objective optimization*, Ph.D. thesis, Osaka Prefecture University, Osaka, Japan, 1997.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 505194, 10 pages
doi:10.1155/2010/505194

Research Article

A Distributed Bio-Inspired Method for Multisite Grid Mapping

I. De Falco,[1] A. Della Cioppa,[2] U. Scafuri,[1] and E. Tarantino[1]

[1] *Institute of High Performance Computing and Networking, National Research Council of Italy, Via P. Castellino 111, 80131 Naples, Italy*
[2] *Natural Computation Laboratory, DIIIE, University of Salerno, Via Ponte don Melillo 1, 84084 Fisciano (SA), Italy*

Correspondence should be addressed to I. De Falco, ivanoe.defalco@na.icar.cnr.it

Received 31 July 2009; Revised 8 January 2010; Accepted 20 March 2010

Academic Editor: Chuan-Kang Ting

Computational grids assemble multisite and multiowner resources and represent the most promising solutions for processing distributed computationally intensive applications, each composed by a collection of communicating tasks. The execution of an application on a grid presumes three successive steps: the localization of the available resources together with their characteristics and status; the mapping which selects the resources that, during the estimated running time, better support this execution and, at last, the scheduling of the tasks. These operations are very difficult both because the availability and workload of grid resources change dynamically and because, in many cases, multisite mapping must be adopted to exploit all the possible benefits. As the mapping problem in parallel systems, already known as NP-complete, becomes even harder in distributed heterogeneous environments as in grids, evolutionary techniques can be adopted to find near-optimal solutions. In this paper an effective and efficient multisite mapping, based on a distributed Differential Evolution algorithm, is proposed. The aim is to minimize the time required to complete the execution of the application, selecting from among all the potential ones the solution which reduces the use of the grid resources. The proposed mapper is tested on different scenarios.

1. Introduction

A grid [1] is a decentralized heterogeneous multisite system which aggregates geographically dispersed and multiowner resources (CPUs, storage system, network bandwidth, etc.). From user's perspective, a grid is a collaborative computationally intensive problem-solving environment in which users execute their distributed jobs. Each job, made up of a collection of separate cooperating and communicating tasks, can be processed on the available grid resources without user's knowledge on where they are or even who owns them.

It is noted that the execution times of distributed applications and the throughput of parallel multicomputer systems are heavily influenced by the task mapping and scheduling which, in case of large and disparate set of grid resources, become still more impractical even for experienced users. In fact, grid resources have a limited capacity and their characteristics vary dynamically as jobs change and randomly arrive. Since, in many cases, single-site resources could be inefficient for meeting job requirements, multisite mapping must be adopted to provide all the possible benefits. Obviously, this latter concern further complicates the mapping operation.

On the basis of these considerations, it is clear that an efficient mapping is possible only if it is supported by a fully automated grid task scheduler [2].

Naturally when a new job is submitted for execution on a grid, the dynamical availability and the pertaining workload of grid resources imply that, to select the appropriate resources, the grid task scheduler has to know number and status of the resources available in that moment. Hence such a scheduler, hereinafter referred to as Metascheduler, is not simply limited to the mapping operation, but must act in three successive phases: resource discovery, mapping or task/node allocation and job scheduling [3].

The resource discovery phase, which has to determine the amount, type, and status of the available resources, can obtain this information either by specific tables based on statistical estimations in a particular time span or gathered tracking periodically and forecasting dynamically resource conditions [4, 5]. For example, in Globus Toolkit [6], which is the middleware used for building grids, global information gathering is performed by the Grid Index Information Service which contacts the Grid Resource Information Service to acquire local information [7].

In the mapping phase, the Metascheduler has to select, in accordance with possible user requirements, the nodes which opportunely match the application needs with the available grid resources.

Finally, in the last phase the Metascheduler establishes the schedule timing of the tasks on the nodes. To have that all the tasks will be promptly coscheduled, our Metascheduler selects, in line with job requirements, resources conditions and knowledge of the different local scheduling policies, only the nodes, even belonging to different sites, which in that moment are able to coschedule the tasks assigned to them. This last assumption avoids to perform the job scheduling phase. It is noted that, if locally supported, an alternative to attain the coscheduling could be to make advance reservations. However, this approach, which requires that resource owners have a good planning on their own tasks, presents difficulties to be employed in a shared environment.

As concerns the resource discovery phase, the Metascheduler here implemented determines the available nodes considering historical information pertaining the workload as a function of time, and the characteristics of each node by using specific tables.

In this paper, the attention is focused only on the mapping phase. Since mapping algorithms for traditional parallel and distributed systems, which usually run on homogeneous and dedicated resources, for example, computer clusters, cannot work adequately in heterogeneous environments [1], other approaches have been proposed to cope with different issues of the problem [8–12].

Generally the allocation of jobs to resources is performed respecting one or more optimization criteria like minimal makespan, minimal cost of assigned resources, or maximal throughput and so on. Here, in contrast to the classical approach [13–15] which takes into account the grid user's point of view and aims at minimizing the completion time of the application task, we deal with the multisite mapping problem from the grid manager's point of view. Thus, our aim is to find the solution which minimizes execution time and communication delays, and optimizes resource utilization using at the minimum the grid resources it has to exploit at the most.

Unfortunately, the mapping problem, already known as NP-complete for parallel systems [16, 17], becomes even more difficult in a distributed heterogeneous environment as in grid systems. Moreover, in the future, grids will be characterized by an increasing number of sites and nodes per site, so as to meet the ever growing computational demands of large and diverse groups of tasks. Hence, it has seemed natural to devote attention to the development of mapping tools based on heuristic optimization techniques, as, for example, evolutionary algorithms. Several evolutionary-based techniques have been used to face the task allocation in a heterogeneous or grid environment [10, 13–15, 18–22].

Within this paper, a distributed version of Differential Evolution (DE) [23, 24] approach is proposed. This technique is attractive because it requires few control parameters, it is relatively easy to implement, effective and efficient in solving practical engineering problems. Unlike all the other existing evolutionary approaches which simply search for mapping the job onto just one site [21], we deal with a multisite approach.

Then, differently from other methods which face the problem of mapping in a heterogeneous environment for applications developed according to a specific paradigm, as, for example, the master/slave model in [25, 26], we do not make hypotheses about the application graph. Moreover, as a further distinctive issue with respect to other approaches in literature [12], we consider the nodes making up the sites as the lowest computational unit taking into account its actual load.

Paper structure is as follows: Section 2 illustrates our evolutionary approach to the mapping problem. Section 3 describes the distributed DE algorithm, while Section 4 reports on the test problems faced and outlines the results achieved. Lastly, Section 5 contains conclusions and future works.

2. Differential Evolution for Mapping

2.1. The Technique. Differential Evolution is a stochastic and reliable evolutionary optimization strategy which presents noticeable performance in optimizing a wide variety of multidimensional and multimodal objective functions in terms of final accuracy and robustness, and overcomes many of the already existing stochastic and direct search global optimization techniques [27–29]. In particular, given a minimization problem with q real parameters, DE faces it starting with a population of $\mathcal{M}$ randomly chosen solution vectors each made up by q real values. At each generation, new vectors are generated by a combination of vectors randomly chosen from the current population. The outcoming vectors are then mixed with a predetermined target vector. This operation is called recombination and produces the trial vector. Many different transformation schemes have been defined by the inventors to produce the candidate trial vector [23, 24]. To explicit the strategy they established a sensible naming-convention for each DE technique with a string like DE/*x*/*y*/*z*. In it, DE stands for Differential Evolution, x is a string which denotes the vector to be perturbed (*best* = the best individual in current population, *rand* = a randomly chosen one, *rand-to-best* = a random one, but the current best participates in the perturbation too), y is the number of difference vectors taken for perturbation of x (either 1 or 2), while z is the crossover method (*exp* = exponential, *bin* = binomial). We have chosen the *DE/rand/1/bin* strategy throughout our investigation. In this model, a random individual is perturbed by using one difference vector and by applying binomial crossover. More specifically, for the generic ith individual in the current population three integer numbers r_1, r_2, and r_3 in $\{1, \ldots, \mathcal{M}\}$ differing one another and different from i are randomly generated. Furthermore, another integer number s in the set $\{1, \ldots, q\}$ is randomly chosen. Then, starting from the ith individual a new trial one i' is generated whose generic jth component is given by

$$x_{i'_j} = x_{r_{3_j}} + F \cdot \left(x_{r_{1_j}} - x_{r_{2_j}} \right) \qquad (1)$$

provided that either a randomly generated real number ρ in $[0.0, 1.0]$ is lower than a value CR (parameter of the DE, in the same range as ρ) or the position j under account is exactly s. If neither is verified, then a simple copy takes place: $x_{i'_j} = x_{i_j}$. F is a real and constant factor which controls the magnitude of the differential variation $(x_{r_{1j}} - x_{r_{2j}})$, and is a parameter of the algorithm.

This new trial individual $x_{i'}$ is compared against the ith individual in the current population and is inserted in the next population if fitter. This basic scheme is repeated for a maximum number of generations g.

2.2. Definitions and Assumptions. In this work, we refer to a grid as a system constituted by one or more sites, each containing a set of nodes, while to a job as a set of distributed tasks, each with various requirements [8, 30–33]. In absence of virtual or dedicated links, sites generally communicate by means of internet infrastructure.

In each site, single node and multinode systems are present. With single node we intend a standalone computational system provided by one or more processors and one or more links, while with multinode we refer to a parallel system. Moreover, we assume that the node is the elementary computation unit and that the proposed mapping is task/node. Each node executes the tasks arranged in two distinct queues: the local queue (L_q) for the locally submitted tasks and the remote queue (R_q) for those presented via grid. The tasks in R_q can be executed only if there are not ready tasks in L_q. While the tasks in L_q will be scheduled on the basis of the locally established policy, a First-Come-First-Served (FCFS) strategy with priority must be adopted for those in R_q. According to this scheduling policy, to perform the mapping process both the current local and grid workloads are taken into account.

To focus the mapping problem in the premised grid we need information on the number and on the status of both accessible and demanded resources. Consequently, we assume to have a grid application subdivided into P tasks (demanded resources) to be mapped on n nodes (accessible resources) with $n \in \{1, \ldots, N\}$, where P is fixed *a priori* and N is the number of grid nodes.

We have to know node capacities (the number of instructions computed per time unit), network bandwidth and load of each grid node in a given time span. In fact, the available power of each node varies over time due to the load by the original users in shared-resource computing. In particular, we need to know *a priori* the number of instructions α_i computed per time unit on node i. Furthermore, we assume to have cognition of the communication bandwidth β_{ij} between any couple of nodes i and j. It should be noted that β_{ij} is the generic element of an $N \times N$ symmetric matrix $\boldsymbol{\beta}$ with very high values on the main diagonal, that is, β_{ii} is the bandwidth between two tasks on the same node. We suppose that this information is contained in tables based on statistical estimations in a particular time span.

In general, grids address nondedicated resources since they have their own local workloads. This affects the availability of local performance. Thus we must consider these load conditions to evaluate the expected computation time. There exist several prediction models to face the challenge of nondedicated resources [34, 35]. For example, as attains the computational power, we suppose to know the average load $\ell_i(\Delta t)$ of the node i at a given time span Δt with $\ell_i(\Delta t) \in [0.0, 1.0]$, where 0.0 means a node completely discharged and 1.0 a node locally loaded at 100%. Hence $(1 - \ell_i(\Delta t)) \cdot \alpha_i$ represents the fraction of power at node i available for executing grid tasks.

As an example, if the resource is a computational node, the conditions collected could be the fraction of CPU which can be destined to the execution of the newly started processes, and the fraction of bandwidths which could be different in conformity with the remote hosts involved in the communication.

As regards the resources requested by the job, we assume to know for each task k the respective number of instructions γ_k to be executed and the number of communications ψ_{km} between the kth and the mth task for all $m \neq k$. Obviously, ψ_{km} is the generic element of a $P \times P$ symmetric matrix $\boldsymbol{\psi}$ with all null elements on the main diagonal.

All this information can be obtained either by a static program analysis, or by using smart compilers or by other emerging tools which automatically generate them. For example, the Globus Toolkit includes the Resource Specification Language which constitutes an XML format to define application requirements [7].

2.3. Encoding. In general, any mapping solution should be represented by a vector $\boldsymbol{\mu}$ of P integers in the set $\{1, \ldots, N\}$. To obtain $\boldsymbol{\mu}$, the real values provided by DE in the range $[1, N + 1[$ are truncated before evaluation. The truncated value $\lfloor \mu_i \rfloor$ denotes the node onto which the task i is mapped.

As long as the mapping is considered by characterizing the tasks by means of their computational needs γ_k only, this is an NP-complete optimization problem, in which the allocation of a task does not affect that of the other ones, unless one attempts to load more tasks on the same node. If, instead, also communications ψ_{km} are taken into account, the mapping problem becomes by far more complicate. In fact, the allocation of a task on a given node can cause that the optimal mapping needs that also other tasks must be allocated on the same node or in the same site, so as to decrease their communication times and thus their execution times, taking advantage of the higher communication bandwidths existing within any site compared to those between sites.

Such a problem is a typical example of *epistasis*, that is, a situation in which the value taken on by a variable influences those of other variables. This situation is also *deceptive*, since a solution $\boldsymbol{\mu}_1$ can be transformed into another with better fitness $\boldsymbol{\mu}_2$ only by passing through intermediate solutions, worse than both $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, which would be discarded. To overcome this problem we have introduced a new operator, named *site mutation*, applied with a probability p_m any time a new individual must be generated. When this mutation is to be carried out, a position in the current solution $\boldsymbol{\mu}$ is randomly chosen. Let us suppose its value refers to a node belonging to a site C_i. This value is equiprobabilistically modified into another one which is related to a node of another cluster, say C_j. Then, any other task assigned to C_i in the current solution is let randomly migrate to a node of C_j by inserting into the related position a random value within the bounds for C_j. If *site mutation* does not take place, the classical transformations typical of DE must be applied.

2.4. Fitness. The two major parties in grid computing, namely, resource consumers who submit various applications, and resources providers who share their resources, usually have different motivations when they join the grid. Currently, most of objective functions in grid computing are inherited from traditional parallel and distributed systems. As attains applications, grid users and providers of resources can have different demands to satisfy. As an example users could be interested in the total cost to run their application, while providers could pay more attention to the throughput of their resources in a particular time interval. Thus objective functions can meet different goals.

In our case, the fitness function calculates the summation of the execution times of the set of all the tasks on the basis of the specific mapping solution.

Use of Resources. Denoting $\tau_{ij}^{\mathrm{comp}}$ and $\tau_{ij}^{\mathrm{comm}}$, respectively, the computation and the communication times requested to execute the task i on the node j it is assigned to the generic element of the execution time matrix $\boldsymbol{\tau}$ is computed as

$$\tau_{ij} = \tau_{ij}^{\mathrm{comp}} + \tau_{ij}^{\mathrm{comm}}. \quad (2)$$

In other words, τ_{ij} is the total time needed to execute task i on node j and is evaluated on the basis of the computation power and of the bandwidth which remain available once deducted the local workload. Let τ_j^s be the summation on all the tasks assigned to the jth node for the current mapping. This value is the time spent by node j in executing computations and communications of all the tasks assigned to it by the proposed solution. Of course, it does not consider the time intervals in which these tasks are idle waiting for communicating, so that tasks dependency does not influence the results of the mapping proposed. Clearly, τ_j^s is equal to zero for all the nodes j not included in the vector $\boldsymbol{\mu}$, that is, all the nodes which do not have assigned tasks.

Considering that all the tasks are coscheduled, the time required to complete the application execution is given by the maximum value among all the τ_j^s. Then, the fitness function is

$$\Phi(\boldsymbol{\mu}) = \max_{j\in\{1,\dots,N\}} \left\{\tau_j^s\right\}. \quad (3)$$

The goal of the evolutionary algorithm is to search for the smallest fitness value among these maxima, that is, to find the mapping which uses at the minimum, in terms of time, the grid resource it has to exploit at the most.

If during the DE generation of new individuals the offspring has the same fitness value as its parent, then it is selected the individual for which

$$\Phi^*(\boldsymbol{\mu}) = \sum_{j=1}^{N} \tau_j^s \quad (4)$$

is smaller. This quantity represents the total amount of time dedicated by the grid to the execution of the job. Obviously, such a mechanism takes place also for the selection of the best individual in the population. This choice aims at meeting the requirements of resource providers, favouring mappings which exploit best the shared resources.

It should be noted that, though the fitness values of the proposed mapping are not related to the completion time of the application, Φ and Φ^* can be seen, respectively, as the lower and the upper bound of the job execution time.

The pseudocode of our DE for mapping is shown in the following Algorithm 1.

3. The Distributed Algorithm

Our Distributed DE (DDE) algorithm is based on the classical coarse-grained approach to Evolutionary Algorithms, widely known in literature [36]. It consists in a locally linked strategy, the *stepping stone-model* [37], in which each DE instance is connected to d instances only. If, for example, we arrange them as a folded torus, then each DE instance has exactly four neighbouring subpopulations as shown in Figure 1, where the generic DE algorithm is shown in black, and its neighbouring subpopulations are indicated in grey. The subpopulation under examination is, thus, "isolated" from all the other ones, shown in white, and it can communicate with them in an indirect way only, through the grey ones. Moreover every M_I generations (*Migration Interval*), neighbouring subpopulations are allowed to exchange individuals. The percentage of individuals each subpopulation sends to its neighbours is called *Migration Rate* (M_R).

A design decision is the quality of the elements to be sent; they might be the best ones or randomly chosen ones. Another decision must be taken about the received individuals; they might anyway replace the worst individuals in the population or substitute them only if better, or they might finally replace any individual (apart from the very best ones, of course). It is known from literature that the number of individuals sent should not be high, nor should the exchange frequency, otherwise the subsearch in a processor might be very disturbed by these continuously entering elements which could even be seen as noise [36]. This mechanism allows to achieve both *exploitation* and *exploration*, which are basic features for a good search. Exploration means to wander through the search space so as to consider also very different situations, looking for the most hopeful (favourable) areas to be intensively sampled. Exploitation means that one area is thoroughly examined, so that we can be confident in being able to state whether this area is promising. By making use of this approach, good solutions will spread within the network with successive diffusions, so more and more processors will try to sample that area (exploitation), and, on the other hand, there will exist at the same time clusters of processors which will investigate different subareas of the search space.

Within this general framework, we have implemented a distributed version for DE, which consists of a set of classical DE schemes, running in parallel, assigned to different

```
begin
randomly initialize population $X = (x_1, \ldots, x_{\mathcal{M}})$
evaluate fitness $\Phi$ for all the individuals $x_i$
while (maximal number of generations $g$ is not reached) do
    begin
     for $i = 1$ to $\mathcal{M}$ do
        begin
           choose a random real number $p_{sm} \in [0.0, 1.0]$
           if $(p_{sm} < p_m)$
            apply site mutation
        else
           begin
              choose three integers $r_1, r_2$ and $r_3 \in \{1, \ldots, \mathcal{M}\}$, with $r_1 \neq r_2 \neq r_3 \neq i$
              choose an integer number $s$ in $\{1, \ldots, q\}$
              for $j = 1$ to $q$ do
                 begin
                    choose a random real number $\rho \in [0.0, 1.0]$
                    if $((\rho < CR)$ OR $(j = s))$
                        $x_{i'_j} = x_{r_{3j}} + F \cdot (x_{r_{1j}} - x_{r_{2j}})$
                    else
                      $x_{i'_j} = x_{i_j}$
                 end
              if $\Phi(x_{i'}) \leq \Phi(x_i)$
                 insert $x_{i'}$ in the new population
              else
                insert $x_i$ in the new population
           end
        end
    end
end
```

Algorithm 1

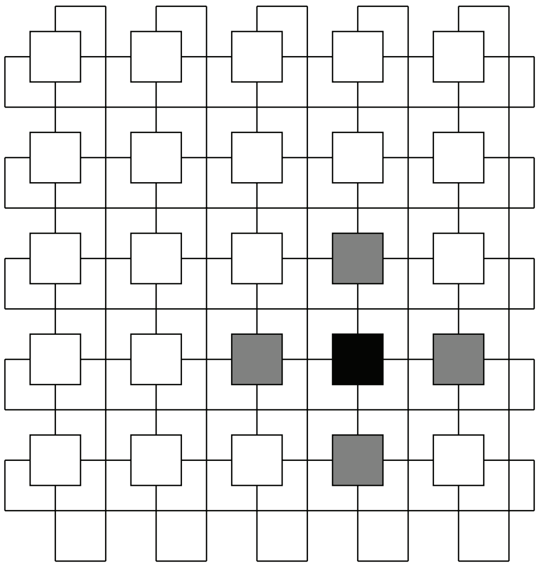


Figure 1: The folded torus topology.

processing elements arranged in a folded torus topology, plus a master. The master process acts as an interface to the user: it simply collects the current local best solutions of the "slave" processes and saves the best among them at each generation. Besides, this latter is compared with the overall best found so far and, if fitter, becomes the new overall best and is shown to the user.

4. Experiments and Findings

Before effecting any kind of experiment the structure of the available resources and the features of the machines belonging to each site must be known. Generally, sites of a grid architecture have different number of systems (parallel machines, clusters, supercomputers, dedicated systems, etc.) with various characteristics and performance. To perform a simulation, we assume to have a grid composed of $N = 58$ nodes subdivided into five sites denoted with A, B, C, D, and E with 16, 8, 8, 10, and 16 nodes, respectively. This grid structure is outlined in Figure 2 while an example of the site B, made up by four single nodes and a four-node cluster, is shown in Figure 3.

Hereinafter, we will denote the nodes by means of the numbers shown in Figure 2, so that, for example, 20 is the fourth node in site B, while 37 is the fifth node in site D.

Without loss of generality, we suppose that all the nodes belonging to the same site have the same power α expressed in terms of millions of instructions per second (MIPS) as shown in Table 1.

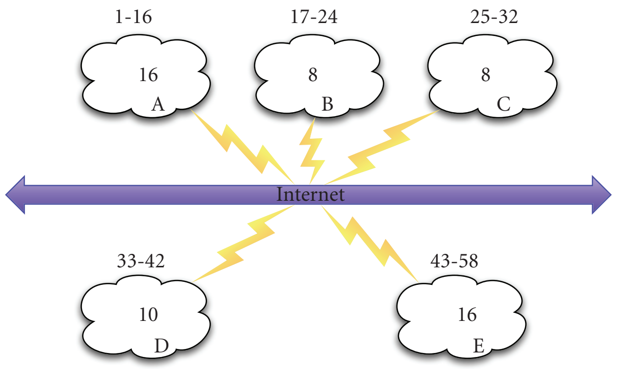


Figure 2: The grid architecture.

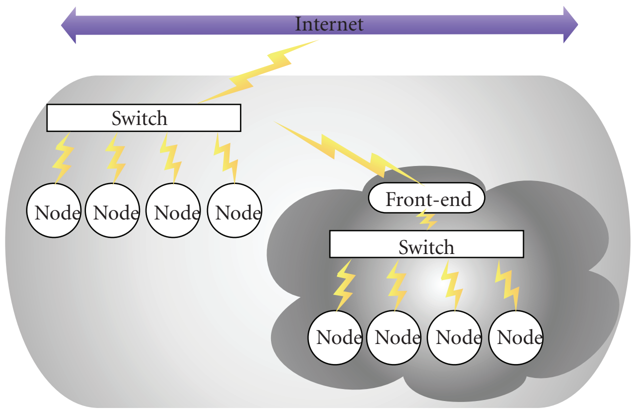


Figure 3: An example of site B.

Table 1: Power of nodes for each site expressed in MIPS.

Sites	A	B	C	D	E
α	500	900	2000	1700	700

For the sake of simplicity, we have hypothesized for each node three communication bands. The first is the bandwidth β_{ii} available when tasks are mapped on the same node (*intranode communication*), the second is the bandwidth β_{ij} between the nodes i and j belonging to the same site (*intrasite communication*), and the third is the bandwidth β_{ij} when the nodes i and j belong to different sites (*intersite communication*). Besides, we presume that all the β_{ii}s have the same very high value (10 Gbit/s) so as to yield the related communication time negligible with respect to intrasite and intersite communications.

For each site, the bandwidth of the output link is supposed equal to that of the input link. In our case, the intersite bandwidths are reported, with the addition of the intrasite bandwidths, in Table 2.

Moreover we assume to know the average load of available grid resources for the time span of interest.

A generally accepted set of heterogenous computing benchmarks does not exist and the detection of a representative set of such benchmarks remains a current and unresolved challenge. To evaluate the effectiveness of our DDE-based

Table 2: Intersite and intrasite bandwidths expressed in Mbit/s.

	A	B	C	D	E
A	10				
B	2	100			
C	6	3	1000		
D	5	10	7	800	
E	2	5	6	1	100

approach we have decided to investigate different application tasks with particular attention to both computation-bound and communication-bound tasks as the load of grid nodes varies.

After a very preliminary tuning phase, the parameters of each DDE have been set as follows: $\mathcal{M} = 30$, $g = 1000$, $CR = 0.3$, $F = 2.0$, $p_m = 0.2$, $M_I = 50$, and $M_R = 1$. This set of parameters is left unchanged for all the experiments carried on.

Our DDE can be profitably used for mapping of message passing applications. Here we have used the Message Passing Interface (MPI) [38] which is a widely used standard library which makes the development of grid applications more accessible to programmers with parallel computing skills. Actually, many MPI library implementations, as MPICH-G2 [39], MagPIe [40], MPI_Connect [41], MetaMPICH [42] and so on, allow the execution of MPI programs on groups of multiple machines potentially based on heterogeneous architectures. However, all these libraries require that users must explicitly specify the resources to be used and they may have enormous difficulties to select, at the best, the appropriate resources for their works in grid environments.

The DDE algorithm has been implemented in C language and all the experiments have been effected on a cluster of 17 (1 master and 16 slaves) 1.5 GHz Pentium 4 interconnected by a FastEthernet switch.

For each test problem 20 DDE executions have been carried out, so as to investigate the dependence of the results on the random seed. Each execution has required 13s for a total of 260*s* for each set of experiments. It should be noted that if the situation described at the end of Section 2.4 takes place when comparing the results of the different runs, the same tie-break mechanism is adopted.

Once defined the evolutionary parameters and the grid characteristics, different scenarios must be designed to demonstrate the effectiveness of the approach over a broad range of realistic conditions. To ascertain the degree of optimality, different tests are conceived to allow a simple comparison between a manual calculation and the solution provided by the mapping tool. Note that, for the sake of simplicity, in the experiments reported, we suppose that the local load of a node is constant during all the execution time of the application task allocated to it. Obviously, a variable load would require only a different calculation but it would not invalidate the approach proposed. In the following, we show the mapping results attained for these experiments.

The first experiment has regarded an application of $P = 12$ tasks with $\gamma_k = 90$ Giga Instructions (GI), $\psi_{km} = 0$ for

Table 3: Findings for each experiment.

Exp. no.	1	2	3	4	5	6	7	8
Φ	52.94	52.94	128.57	139.19	180.00	271.73	484.77	128.57
Φ^*	571.76	587.64	2571.42	2744.59	5387.14	8693.28	5817.29	1781.22
n_b	20	15	20	15	3	4	3	20
Φ_{av}	52.94	62.20	128.57	156.64	218.25	298.50	939.97	128.57
σ	0.00	16.46	0.00	31.01	16.48	13.73	255.29	0.00

all $k, m \in \{1, \ldots, P\}$, and $\ell_i(\Delta t) = 0$ for all the nodes. The mapping solution found by our DDE is:

$$\boldsymbol{\mu} = \{25, 26, 27, 28, 29, 30, 31, 32, 41, 42, 35, 36\}. \quad (5)$$

As expected, the mapping procedure has allocated all the tasks on the most powerful available nodes, eight belonging to the site C and four to site D.

In the second experiment, all the parameters remain unchanged except the load. In particular, we have supposed $\ell(\Delta t) = 0.7$ on the two nodes 31 and 32 and $\ell(\Delta t) = 0.5$ on the three nodes 40, 41, and 42. In this hypothesis, the mapping solution found is

$$\boldsymbol{\mu} = \{25, 34, 27, 28, 37, 30, 39, 38, 33, 29, 36, 26\}. \quad (6)$$

As it can be observed the solution again involves the most powerful nodes (six belonging to C and six to D), discarding correctly the loaded nodes in those sites.

In the third experiment, we have $P = 20$ with $\gamma_k = 90$ GI, $\psi_{km} = 0$ for all $k, m \in \{1, \ldots, P\}$ and $\ell(\Delta t) = 0.9$ for all the nodes of the sites B and D, while for the site C we assume $\ell_i(\Delta t) = 0.8$ for $i \in \{25, \ldots, 28\}$ and $\ell_i(\Delta t) = 0.6$ for $i \in \{29, \ldots, 32\}$. The mapping solution discovered by our DDE is

$$\boldsymbol{\mu} = \{43, 44, 45, 46, 47, 29, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 48, 30, 31, 32\}. \quad (7)$$

It is worth noticing that in this load conditions the mapping procedure has chosen once again the most powerful nodes: 4 of C with $\ell_i(\Delta t) = 0.6$ which are those with a minor load and 16 of E.

The same solution has been obtained in the fourth experiment where we have just introduced the communications $\psi_{km} = 10$ Mbit for all $k, m \in \{1, \ldots, P\}$.

In the fifth experiment, we have left unchanged both the load conditions and the number of instructions that each task k has to effect ($\gamma_k = 90$ GI). Simply we have considered $P = 36$ and removed all the communications among the tasks. The allocation is outlined in the following

$$\boldsymbol{\mu} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 47, 50, 44, 57, 46, 31, 29, 54, 49, 51, 55, 53, 30, 32, 45, 52, 43, 58, 56, 48\}. \quad (8)$$

This solution, according to the load conditions, has mapped 16 tasks on the 16 nodes of A, 16 on all the nodes of E, and 4 on the 4 nodes of C which present the lowest load (0.6).

In the sixth experiment, we have merely added a communication $\psi_{km} = 10$ Mbit for all $k, m \in \{1, \ldots, P\}$. The result is:

$$\boldsymbol{\mu} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 43, 45, 51, 28, 58, 49, 46, 52, 47, 48, 56, 27, 57, 50, 30, 29, 25, 32, 31, 26\}. \quad (9)$$

Such a solution provides 16 tasks on the 16 nodes of site A, 12 on the site E, and 8 on all the nodes of C. It can be noted that the mapping proposed has selected four nodes of C which are loaded at 0.8, and therefore less powerful than the other discharged nodes of E, to exploit the major bandwidth among nodes allocated on the site C with respect to the intersite bandwidth between C and E.

The influence of the communications is highly evidenced in the successive experiment where, leaving unchanged all the other conditions, the communication ψ_{km} has been set to 100 Mbit for all $k, m \in \{1, \ldots, P\}$. The mapping proposed has allocated all the 36 tasks on the 16 nodes of site E. In fact, the time requested to perform the communications becomes relevant compared to the computation time and thus it is advantageous to allocate more tasks on each node of site E rather than to subdivide them on nodes of different sites. The solution is

$$\boldsymbol{\mu} = \{53, 47, 43, 44, 47, 48, 45, 49, 50, 46, 46, 48, 49, 50, 43, 44, 53, 51, 51, 45, 52, 52, 54, 54, 56, 57, 57, 55, 55, 56, 58, 58, 43, 47, 53, 55\}. \quad (10)$$

As an example of the behavior shown by our tool, Figure 4 reports the evolution of the best run achieved for this last test. Namely, we depict the best, average and worst fitness values among those sent to the master by the 16 slaves at each generation. Since the initial generation the average, the best and the worst fitness values decrease over generations, and this continues until the end of the run. Every now and then several successive generations take place in which no improving solutions are found, and this results in best, average and worst values becoming more and more similar. Then, a new better solution is found and the three values become quite different. The described behavior implies that good solutions spread only locally among linked

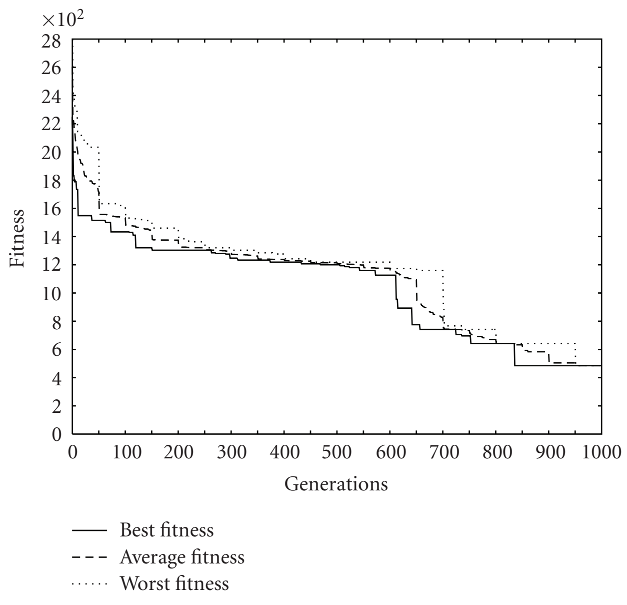


Figure 4: Behavior of fitness as a function of the number of generations for the best run of experiment 7.

subpopulations without causing premature convergence to the same suboptimal solution on all the slaves, which is a positive feature of the system.

The final experiment has attained a job with $P = 36$, $\gamma_k = 90$ GI for $k \in \{1, \ldots, 12\}$, $\gamma_k = 9$ GI for $k \in \{13, \ldots, 36\}$, $\psi_{km} = 0$ for all $k, m \in \{1, \ldots, 24\}$ and $\psi_{km} = 10$ Mbit for all $k, m \in \{25, \ldots, 36\}$, while the load conditions are the same of the previous experiment. The mapping found is

$$\boldsymbol{\mu} = \{43, 44, 45, 46, 47, 48, 31, 32, 51, 52, 53, 30, 55, 56, 57, 58, 29, 57, 58, 31, 58, 57, 58, 57, 29, 29, 29, 29, 29, 29, 29, 30, 29, 29, 29, 32\}. \tag{11}$$

From the mapping proposed, it can be observed that 17 tasks are placed on C and 19 are allocated on E. In particular, three of the tasks with $\gamma_k = 90$ GI have been mapped on three nodes of C with $\ell(\Delta t) = 0.6$ (nodes 30, 31 and 32) and the remaining 9 with the same computational requirements on 9 nodes of site E, while the fourth node of C with $\ell(\Delta t) = 0.6$ (node 29) has been used to allocate 10 tasks each with $\gamma_k = 9$ GI and $\psi_{km} = 10$ Mbit for all $(k, m) \in \{25, \ldots, 36\}$.

In Table 3, for each experiment (Exp. no) the best fitness values for Φ and Φ^* are outlined and, for all the 20 runs, the number of occurrences (n_b) of the best result, the average fitness values (Φ_{av}), and the standard deviations σ are shown.

The tests performed have evidenced a high degree of efficiency of the proposed model in terms of both goodness of the solutions provided and convergence times. In fact, efficient solutions have been quickly provided independently of work conditions (heterogenous nodes diverse in terms of number, type, and load) and kind of jobs (computation or communication bound).

5. Conclusions and Future Works

This paper faces the multisite mapping problem in a grid environment by means of Differential Evolution. In particular, the goal is the minimization of the degree of use of the grid resources by the proposed mapping. The results show that a Distributed Differential Evolution algorithm is a viable approach to the important problem of grid resource allocation. A comparison with other methods is impossible at the moment due to the lack of approaches dealing with this problem in the same operating conditions as ours. In fact, some of these algorithms, such as Min-min, Max-min, and XSuffrage [12], are related to independent tasks and their performances are affected in heterogenous environments. In case of dependent tasks, the classical approaches apply the popular model of Direct Acyclic Graph (DAG) differently from our approach in which no assumptions are made about the communications among the processes since we have hypothesized tasks coscheduling.

Future works will include an investigation of the different DE schemes, together with a wide tuning phase for parameter sets, to experiment their effectiveness in facing the problem under exam.

A dynamic measure of the load of grid nodes will be examined. Furthermore, we have supposed that the cost per MIPS and Mbit/s is the same for all the grid nodes. Since nodes with different features have different costs, in the future these costs will be added to the other parameters considered in the mapping strategy.

Finally, since Quality of Service (QoS) assumes an important role for many grid applications, we intend to enrich our tool so it will be able to manage multiple QoS requirements as those on performance, reliability, bandwidth, cost, response time, and so on.

References

[1] F. Berman, "High-performance schedulers," in *The Grid: Blueprint for a Future Computing Infrastructure*, I. Foster and C. Kesselman, Eds., pp. 279–307, Morgan Kaufmann, San Francisco, Calif, USA, 1998.

[2] G. Mateescu, "Quality of service on the grid via metascheduling with resource co-scheduling and co-reservation," *International Journal of High Performance Computing Applications*, vol. 17, no. 3, pp. 209–218, 2003.

[3] J. M. Schopf, "Ten actions when grid scheduling: the user as a grid scheduler," in *Grid Resource Management: State of the Art and Future Trends*, pp. 15–23, Kluwer Academic Publishers, Norwell, Mass, USA, 2004.

[4] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, pp. 365–375, IEEE Computer Society, Portland, Ore, USA, August 1997.

[5] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 181–194, San Francisco, Calif, USA, August 2001.

[6] I. Foster, "Globus toolkit version 4: software for service-oriented systems," in *Proceedings of IFIP International Conference on Network and Parallel Computing (NPC '05)*, vol. 3779 of *Lecture Notes in Computer Science*, pp. 2–13, Beijing, China, November-December 2005.

[7] L. Adzigogov, J. Soldatos, and L. Polymenakos, "EMPEROR: an OGSA grid meta-scheduler based on dynamic resource predictions," *Journal of Grid Computing*, vol. 3, no. 1-2, pp. 19–37, 2005.

[8] R. F. Freund, "Optimal selection theory for super concurrency," in *Supercomputing*, pp. 699–703, IEEE Computer Society, Reno, Nev, USA, 1989.

[9] M. M. Eshaghian and M. E. Shaaban, "Cluster-m programming paradigm," *International Journal of High Speed Computing*, vol. 6, no. 2, pp. 287–309, 1994.

[10] T. D. Braun, H. J. Siegel, N. Beck, et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[11] K.-H. Kim and S.-R. Han, "Mapping cooperating grid applications by affinity for resource characteristics," in *Proceedings of the 13th International Conference on AIS*, vol. 3397 of *Lecture Notes in Artificial Intelligence*, pp. 313–322, 2005.

[12] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing: state of the art and open problems," Tech. Rep. 2006-504, School of Computing, Queens University, Kingston, Canada, 2006.

[13] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," in *Proceedings of Heterogeneous Computing Workshop*, pp. 86–97, IEEE Computer Society, Honolulu, Hawaii, USA, 1996.

[14] P. Shroff, D. W. Watson, N. S. Flan, and R. F. Freund, "Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments," in *Proceedings of Heterogeneous Computing Workshop*, pp. 98–104, IEEE Computer Society, Honolulu, Hawaii, USA, 1996.

[15] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. MacIejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 8–22, 1997.

[16] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non identical processors," *Journal of Association for Computing Machinery*, vol. 24, no. 2, pp. 280–289, 1977.

[17] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, 1989.

[18] Y.-K. Kwok and I. Ahmad, "Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm," *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 58–77, 1997.

[19] A. Abraham, R. Buyya, and B. Nath, "Nature's heuristics for scheduling jobs on computational grids," in *Proceedings of the 8th International Conference on Adavanced Computing and Communication*, pp. 45–52, 2000.

[20] S. Kim and J. B. Weissman, "A genetic algorithm based approach for scheduling decomposable data grid applications," in *Proceedings of the International Conference on Parallel Processing (ICPP '04)*, pp. 406–413, Montreal, Canada, August 2004.

[21] A. Bose, B. Wickman, and C. Wood, "MARS: a metascheduler for distributed resources in campus grids," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID '04)*, pp. 110–118, IEEE Computer Society, Pittsburgh, Pa, USA, November 2004.

[22] S. Song, Y.-K. Kwok, and K. Hwang, "Security-driven heuristics and a fast genetic algorithm for trusted grid job scheduling," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, p. 65, Denver, Colo, USA, April 2005.

[23] K. Price and R. Storn, "Differential evolution," *Dr. Dobb's Journal*, vol. 22, no. 4, pp. 18–24, 1997.

[24] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[25] G. Shao, F. Berman, and R. Wolski, "Master/slave computing on the grid," in *Proceedings of the 9th Heterogeneous Computing Workshop*, pp. 3–16, IEEE Computer Society, Cancun, Mexico, 2000.

[26] N. Ranaldo and E. Zimeo, "An economy-driven mapping heuristic for hierarchical master-slave applications in grid systems," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS '06)*, Rhodes Island, Greece, 2006.

[27] S. Das, A. Abraham, and A. Konar, "Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives," in *Studies in Computational Intelligence*, Y. Liu, et al., Ed., vol. 116, pp. 1–38, Springer, Berlin, Germany, 2008.

[28] A. Nobakhti and H. Wang, "A simple self-adaptive differential evolution algorithm with application on the ALSTOM gasifier," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 350–370, 2008.

[29] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[30] R. F. Freund and H. J. Siegel, "Heterogeneous processing," *IEEE Computer*, vol. 26, no. 6, pp. 13–17, 1993.

[31] A. Khokhar, V. K. Prasanna, M. Shaaban, and C. L. Wang, "Heterogeneous computing: challenges and opportunities," *IEEE Computer*, vol. 26, no. 6, pp. 18–27, 1993.

[32] H. J. Siegel, J. K. Antonio, R. C. Metzger, M. Tan, and Y. A. Li, "Heterogeneous computing," in *Parallel and Distributed Computing Handbook*, A. Y. Zomaya, Ed., pp. 725–761, McGraw-Hill, New York, NY, USA, 1996.

[33] V. S. Sunderam, "Design issues in heterogeneous network computing," in *Proceedings of the Workshop on Heterogeneous Processing*, pp. 101–112, IEEE Computer Society, Beverly Hills, Calif, USA, 1992.

[34] R. Wolski, N. T. Spring, and J. Hayes, "Network weather service: a distributed resource performance forecasting service for metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5, pp. 757–768, 1999.

[35] L. Gong, X.-H. Sun, and E. F. Watson, "Performance modeling and prediction of nondedicated network computing," *IEEE Transactions on Computers*, vol. 51, no. 9, pp. 1041–1055, 2002.

[36] E. Cantú-Paz, "A summary of research on parallel genetic algorithms," Tech. Rep. 95007, University of Illinois, Urbana-Champaign, Ill, USA, July 1995.

[37] H. Mühlenbein, "Evolution in time and space—the parallel genetic algorithm," in *Foundation of Genetic Algorithms*, pp. 316–337, Morgan Kaufmann, San Francisco, Calif, USA, 1992.

[38] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference, Vol. 1—The MPI Core*, MIT Press, Cambridge, Mass, USA, 1998.

[39] N. T. Karonis, B. Toonen, and I. Foster, "MPICH-G2: a grid-enabled implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing*, vol. 63, no. 5, pp. 551–563, 2003.

[40] T. Kielmann, H. E. Bal, J. Maassen, et al., "Programming environments for high-performance grid computing: the Albatross project," *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1113–1125, 2002.

[41] G. E. Fagg, K. S. London, and J. J. Dongarra, "MPI connect: managing heterogeneous MPI applications interoperation and process control," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, vol. 1497 of *Lecture Notes in Computer Science*, pp. 93–96, Springer, New York, NY, USA, 1998.

[42] B. Bierbaum, C. Clauss, T. Eickermann, et al., "Reliable orchestration of distributed MPI-applications in a UNICORE-based grid with MetaMPICH and MetaScheduling," in *Proceedings of the 13th European PVM/MPI User's Group Meeting*, vol. 4192 of *Lecture Notes in Computer Science*, pp. 174–183, Bonn, Germany, September 2006.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 185063, 11 pages
doi:10.1155/2010/185063

Review Article

A Review of Constraint-Handling Techniques for Evolution Strategies

Oliver Kramer

International Computer Science Institute, Berkeley, CA 94704, USA

Correspondence should be addressed to Oliver Kramer, okramer@icsi.berkeley.edu

Received 24 September 2009; Accepted 6 January 2010

Academic Editor: Chuan-Kang Ting

Evolution strategies are successful global optimization methods. In many practical numerical problems constraints are not explicitly given. Evolution strategies have to incorporate techniques to optimize in restricted solution spaces. Famous constraint-handling techniques are penalty and multiobjective approaches. Past work has shown that in particular an ill-conditioned alignment between the coordinate system of Gaussian mutation and the constraint boundaries leads to premature convergence. Covariance matrix adaptation evolution strategies offer a solution to this alignment problem. Last, metamodeling of the constraint boundary leads to significant savings of constraint function calls and to a speedup by repairing infeasible solutions. This work gives a brief overview over constraint-handling methods for evolution strategies by demonstrating the approaches experimentally on two exemplary constrained problems.

1. Introduction

Many continuous optimization problems in practical applications are subject to constraints [1]. Constraints can make an easy problem hard and hard problems even harder. Surprisingly, in the past only little research efforts have been devoted to the development of efficient and effective constraint-handling techniques—in contrast to the energy invested in the development of new methods for unconstrained optimization. This observation also holds true in the field of evolutionary computation. This paper is devoted to constraint-handling techniques that have been developed, in particular for evolution strategies. It summarizes our line of research of the last years in the field of constraint-handling and premature step-size reduction [2–7]. In real-valued solution spaces a constrained problem can be hard to solve due to a coordinate system alignment problem that leads to premature fitness stagnation. We review not only various general approaches like penalty functions, but also specialized approaches that have been developed to solve coordinate alignment problems, by summarizing each constraint-handling method, stating experimental results on two exemplary test functions and discussing advantages and disadvantages.

The remainder of this section gives a brief introduction to evolution strategies, constrained problems, and a taxonomy of constraint-handling techniques. Section 2 introduces three examples from the famous family of penalty functions. A bioinspired multiobjective approach is reviewed in Section 3. The methods that concentrate on coordinate system alignment are presented in Section 4, while Section 5 is devoted to metamodeling of the constraint boundary.

1.1. Evolution Strategies. Evolution strategies (ES) are a family of strong stochastic methods for global optimization. Developed by Rechenberg [8] and Schwefel [9], they have become famous for global numerical optimization, that is, nonconvex optimization in $\mathbb{R}^N$. In each iteration λ offspring solutions are produced and the μ best are selected as parents for the following generation. An important basis of ES is the self-adaptive Gaussian mutation operator that we briefly repeat in this context. An individual $\mathbf{a}$ of a $(\mu\overset{+}{,}\lambda)$-ES with the N-dimensional objective variable vector $\mathbf{x} \in \mathbb{R}^N$ is mutated in the following way:

$$\begin{gathered}\mathbf{x}' := \mathbf{x} + \mathbf{z}, \\ \mathbf{z} := (\sigma_1 \mathcal{N}_1(0,1), \dots, \sigma_N \mathcal{N}_N(0,1)),\end{gathered} \tag{1}$$

while $\mathcal{N}_i(0,1)$ delivers a Gaussian distributed number. The strategy parameter vector undergoes mutation—a typical variation of σ—with log-normal mutation:

$$\sigma' := e^{(\tau_0 \mathcal{N}_0(0,1))} \cdot \left(\sigma_1 e^{(\tau_1 \mathcal{N}_1(0,1))}, \dots, \sigma_N e^{(\tau_1 \mathcal{N}_N(0,1))}\right), \qquad (2)$$

as crossover operator arithmetic recombination is applied in most cases. For a detailed introduction to ES we recommend the introduction by Beyer and Schwefel [10] or the introductory chapter to ES in Eiben's book [11].

1.2. Constrained Problems. In the field of evolutionary computation the constraints typically are not considered available in their explicit formal form. Rather, the constraints are assumed to be black boxes: a vector $\mathbf{x}$ fed to the black box just returns a numerical or boolean value. If there is a numerical response, then the information about a positive value can be used to assess the distance to feasible solutions. A number of constraint-handling methods exploit this information. In general, the constrained continuous nonlinear programming problem is defined as follows: find a solution $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$ in the N-dimensional solution space $\mathbb{R}^N$ that minimizes the objective function $f(\mathbf{x})$, in symbols as:

$$\begin{aligned} & f(\mathbf{x}) \longrightarrow \min!, && \mathbf{x} \in \mathbb{R}^N && \text{subject to} \\ & \text{inequalities} && g_i(\mathbf{x}) \le 0, && i = 1, \dots, n_1, \\ & \text{equalities} && h_j(\mathbf{x}) = 0, && j = 1, \dots, n_2 . \end{aligned} \qquad (3)$$

A feasible solution $\mathbf{x} \in \mathbb{R}^N$ satisfies all n_1 inequality and n_2 equality constraints. A feasible solution that minimizes $f(\cdot)$ is termed as an optimal solution. If $g_i(\mathbf{x}^*) = 0$ for some inequality constraint at an optimal solution $\mathbf{x}^*$, then the constraint is said to be active. We assume that the evaluations of the constraint functions are computationally expensive and that the return values are boolean and provide the information of whether the solution is feasible or not. In order to be able to develop more advanced constraint-handling techniques, for example, repair or feasibility check approaches, metamodels of the constraint function can be built with certain assumptions, that is, to be linear, quadratic, and so forth.

The two following test functions excellently demonstrate the phenomenon of premature fitness stagnation that will be discussed in the following sections and that is a challenge for most constraint-handling techniques. The two functions will be used for the discussion of the methods reviewed in the current paper. Problem 2.40—taken from Schwefel's artificial test problems [10]— exhibits a linear objective function and an optimum with five active linear constraints. The problem is to minimize

$$f_{2.40}(\mathbf{x}) = -\sum_{i=1}^{5} x_i, \qquad (4)$$

subject to

$$g_{2.40}(\mathbf{x}) = \begin{cases} x_j \ge 0, & \text{for } j = 1, \dots, 5, \\ -\sum_{i=1}^{5} (9+i)x_i + 50000 \ge 0, & \text{for } j = 6 \end{cases} \qquad (5)$$

with minimum $\mathbf{x}^* = (5000, 0, 0, 0, 0)^T$ and $f(\mathbf{x}^*) = -5000$.

The second problem is called tangent problem (TR). It is based on the sphere model subject to one linear constraint:

$$f_{\text{TR}}(\mathbf{x}) = \sum_{i=1}^{N} x_i \quad \text{with } g_{\text{TR}}(\mathbf{x}) = \sum x_i - N > 0 \qquad (6)$$

with $\mathbf{x}^* = (1, \dots, 1)^T$ and $f(\mathbf{x}^*) = N$. The success rates on TR get worse when approximating the optimum. In this paper we will focus on the TR problem with $N = 2$ dimensions, denoted as TR2.

1.3. A Brief Taxonomy of Constraint-Handling Methods. A variety of constraint-handling methods for evolutionary algorithms have been developed in the last decades. Most of them can be classified into five main types of concepts.

(i) *Penalty functions* decrease the fitness of infeasible solutions by taking the number of infeasible constraints or the distance to feasibility into account [12–16]. The history of penalty functions began with the sequential unconstrained minimization technique by Fiacco and McCormick [13] in which the constrained problem is solved by a sequence of unconstrained optimizations. The penalty factors are stepwise intensified. In similar approaches penalty factors can be defined statically [14] or depending on the number of satisfied constraints [16]. They can dynamically depend on the number of generations as Joines and Houck propose [15]:

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + (C \cdot t)^{\alpha} \cdot G(\mathbf{x}), \qquad (7)$$

at generation t, parameters C and α are user defined; typical settings are $C = 0.5$, $\alpha = 1$, or 2. $G(\mathbf{x})$ is a measure for the constraint violation. A frequent definition is $G(\mathbf{x}) = \sum_{i=1}^{n_1} \max[0, g_i(\mathbf{x})]^{\beta} + \sum_{j=1}^{n_2} |h_j(\mathbf{x})|^{\gamma}$ with factors $\beta \ge 1$ and $\gamma \ge 1$. Penalties can be adapted according to an external cooling scheme [15] or by adaptive heuristics [12]. In the death penalty approach [5] infeasible solutions are rejected and new solutions are created until enough feasible ones exist. In the segregated genetic algorithm by Riche et al. [17] two penalty functions, a weak one and an intense one, are calculated in order to surround the optimum. In the coevolutionary penalty function approach by Coello Coello [18] the penalty factors of an inner evolutionary algorithm are adapted by an outer evolutionary algorithm. Some methods are based on the assumption that any feasible solution is better than any infeasible one [19, 20]. Examples are the metric penalty functions by Hoffmeister and Sprave [21]. Feasible solutions are compared using the objective function while infeasible solutions are compared considering the satisfaction of constraints. Similar is the approach by

Oyman et al. [22]. Their fitness function depends on the parent and children population at every generation and, therefore, becomes a dynamic approach.
In his approach called stochastic-ranking Runarsson [23] he uses metamodels to predict both fitness functions values and penalties based on constraint violations. From this point of view the approach is related to methods that are based on metamodeling the constraint boundary.

(ii) *Repair algorithms* either replace infeasible solutions or only use the repaired solutions for evaluation of their infeasible pendants [24, 25]. This class of algorithms can also be seen as local search methods that reduce the constraint violation. The repair algorithm generates a feasible solution from an infeasible one. In the Baldwinian case, the fitness of the repaired solution replaces the fitness of the original solution. In the Lamarckian case, the feasible solution overwrites the infeasible one. In general, defining a repair algorithm can be as complex as solving the problem itself.

(iii) *Decoder functions* map genotypes to phenotypes which are guaranteed to be feasible. Decoders build up a relationship between the constrained solution space and an artificial solution space easier to handle [25–27]. They map a genotype into a feasible phenotype. By this means even quite different genotypes may be mapped onto the same phenotype. Eiben and Smith [11] define decoders as a class of mappings from the genotype space $\mathcal{S}'$ to the feasible regions $\mathcal{F}$ of the solution space $\mathcal{S}$ with the following properties: every $z \in \mathcal{S}'$ must map to a single solution $s \in \mathcal{F}$, every solution $s \in \mathcal{F}$ must have at least one representation $s' \in \mathcal{S}'$, and every $s \in \mathcal{F}$ must have the same number of representations in $\mathcal{S}'$ (this need not be one).

(iv) *Feasibility preserving representations and operators* force candidate solutions to be feasible [28, 29]. A famous example is the GENOCOP algorithm [27] that reduces the problem to convex search spaces and linear constraints. A predator-prey approach to handle constraints is proposed by Paredis [28] using two separate populations. Schoenauer and Michalewicz [29] propose special operators that are designed to search regions in the vicinity of active constraints. A comprehensive overview to decoder-based constraint-handling techniques is given by Coello [25] and also by Michalewicz and Fogel [27].

(v) *Multiobjective optimization* techniques are based on the idea of handling each constraint as an objective [30–35]. Under this assumption many multiobjective optimization methods can be applied. Such approaches were used by Parmee and Purchase [34], Jimenez and Verdegay [32], Coello Coello [31], and Surry et al. [36]. In the behavioral memory-method by Schoenauer and Xanthakis [35] the EA concentrates on minimizing the constraint violation of each constraint in a certain order and optimizing the objective function in the last step.

Of course, constraint-handling methods exist that do not fit into the taxonomy. Montes and Coello Coello [37] introduced a technique based on a multimembered ES with a feasibility comparison mechanism. The ϵ-constrained differential evolution approach by Takahama and Sakai [38] combines the usage of an ϵ for equality constraints with differential evolution. The dynamic multiswarm particle optimizer by Liang and Suganthan [39] makes use of a set of subswarms concentrating on different constraints. It is combined with sequential quadratic programming as a local search method. The approach of Mezura-Montes et al. [40] combines differential evolution, different mutation operators to increase the probability of producing better offspring, three selection criteria, and a diversity mechanism. Mezura-Montes [41] approach gives a survey of constraint-handling methods for evolutionary algorithms.

In the following section we will compare various approaches from different fields and compare them, in particular with regard to the mentioned premature step-size problem. The next section shows this problem experimentally.

Table 1: Experimental results of the death penalty method.

Death penalty	best	mean	dev	ffe	cfe
TR2	$4.1 \cdot 10^{-7}$	$3.1 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$	11,720	20,447
2.40	51.9	227.6	65.2	50,624	96,817

2. Penalty Methods

Evolutionary search is guided by the quality of its candidate solutions. Consequently, an obvious solution to constraint-handling is to deteriorate the fitness of infeasible methods [11, 25]. Here we review three penalty functions exemplarily. Death penalty is the simplest way, but wastes comparably many constraint function calls. Paragraph 2.2 is a typical penalty technique where the solutions are penalized with regard to the progress of the search. The death penalty step control approach that prevents premature step-size reduction is reviewed in Section 2.3.

2.1. Death Penalty. First of all, we will analyze the behavior of death penalty, that is, simply discarding infeasible offspring solutions [42, 43]. This is the first time we can observe premature fitness stagnation. Table 1 shows the corresponding results of a (15,100)-ES with the following settings on problems 2.40 and TR2. We use the mutation introduced in Section 1.1 with settings $\tau_0 = (\sqrt{2N})^{-1}$ and $\tau_1 = (\sqrt{2\sqrt{N}})^{-1}$ and arithmetic recombination with $\rho = 2$ randomly chosen parents. All experiments in this article make use of the same experimental settings unless stated explicitly. The termination condition is fitness stagnation: the algorithms terminate if the fitness win from generation t to generation $t+1$ falls below $\theta = 10^{-12}$. In this case the magnitude of the step sizes is too small to effect further improvements. Parameters best, mean, and dev describe the achieved fitness (difference between the optimal fitness and the fitness of the best solution $|f(\mathbf{x}^*) - f(\mathbf{x}^{\text{best}})|$) of 25 experimental runs while ffe counts the average number of fitness function evaluations and cfe of constraint function evaluations, respectively. The results show that death penalty is not able to approximate the optimum of the problem satisfactorily. The relatively high-standard deviations dev show that the algorithms produce unsatisfactorily different results.

Table 2: Experimental results of the dynamic penalty function by Joines and Houck [15] on problems TR2 and 2.40.

	best	mean	dev	ffe	cfe
TR2	$1.2 \cdot 10^{-6}$	$1.2 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	13,100	13,100
2.40	219.4	440.8	85.0	31,878	31,878

We can summarize the behavior of death penalty mentioning the advantage that *death penalty* is easy to implement. The disadvantages are that *death penalty* is inefficient as many infeasible tries are wasted, and it suffers from premature convergence. The following methods aim at preventing premature convergence.

2.2. Dynamic Penalty Functions. The question arises whether dynamic penalty functions also suffer from premature convergence. To answer this question we tested the penalty function by Joines and Houck [15] that is based on adding a penalty on infeasible solutions

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + (C \cdot t)^{\alpha} \cdot \sum_{i=1}^{n_1} G_i^{\beta} \tag{8}$$

with parameters C, α, β and the constraint violation $G_i(\mathbf{x}) = \sum_{i=1}^{n_1} \max[0, g_i(\mathbf{x})]^{\beta}$. The penalty depends on the number of iterations t and decreases in the course of time. Table 2 shows the experimental analysis of the penalty function on 2.40 and TR2 with $\alpha = 1.0$ and $\beta = 1.0$. Again, the algorithm is based on a (15,100)-ES with the same settings like in the last paragraph 2.1. The algorithm stops earlier, but the results are even worse and show that premature fitness stagnation occurs, too. The reason is quite obvious: the success rate in the vicinity of the infeasible search space remains small because of the penalty—no matter whether caused by discarding or penalizing. Consequently, we can summarize as follows: *dynamic penalty functions* are easy to implement, and no feasible starting point is required. But the disadvantages are that *dynamic penalty functions* suffer from premature convergence. Related work on penalty functions can be found in [12–16].

2.3. Death Penalty Step Control. The most obvious modification to prevent premature step-size reduction is the introduction of a minimum step-size ϵ for the mutation strengths σ_i with $1 \leq i \leq N$:

$$\sigma_i \geq \epsilon. \tag{9}$$

This is exactly what the death penalty step control evolution strategy (DSES) is aiming at [5]. Nevertheless, a lower bound on the step sizes will also prevent an unlimited approximation of the optimum when reaching the range of ϵ. Consequently, the DSES makes use of a control mechanism to reduce ϵ during convergence to the optimal solution. The reduction process depends on the number of infeasible mutations produced when reaching the area of the optimum at the boundary of the feasible solution space. The reduction process of ϵ depends on the number z of rejected infeasible solutions: in every ϖ infeasible trial, ϵ is reduced by a factor $0 < \vartheta < 1$ according to the equation

$$\epsilon' := \epsilon \cdot \vartheta. \tag{10}$$

The DSES is denoted by $[\varpi; \vartheta]$-DSES. Again, we show the behavior of the constraint-handling method on problem TR2 and 2.40; see Table 3. The method is able to approximate the optimum of problem 2.40 with comparably few fitness function evaluations, but a waste of constraint function evaluations. Intuitively, the five active linear constraints of problem 2.40 cause many infeasible samples, so does the step-sizes reduction mechanism. On *harder* problems like TR2 the low success rates still prevent an arbitrarily exact approximation of the optimal solution. The success of the DSES depends on a proper *reduction speed*, that is, proper parameter settings for ϵ and ϑ. Too fast reduction results in premature convergence; too slow reduction is inefficient. Further experiments on other test functions confirm this picture.

Again, we summarize the following results: *death penalty step control* is easy to implement, and shows an improvement of the approximation of optima with active constraints. But the disadvantages are that *death penalty step control* consumes many constraint function evaluations, its success depends on proper parameter settings, and on some problems it may still suffer from low success rates. A more detailed experimental analysis of the DSES can be found in [4, 5].

3. A Multiobjective Bioinspired Approach

A familiar variant to handle constraints is to treat each constraint—or an aggregated sum of all constraints—and the objective function as separate objectives in a multiobjective formulation. Similar approaches have been introduced in the past [30–35]. Here we review a similar constraint-handling technique that treats the fulfillment of constraints and the optimization of the objective function as separate objectives that are optimized using a population specific selection scheme. The bioinspired concept offers an answer to the problem of low success rates: our two-sex evolution strategy (Kramer and Schwefel [5]) allows candidate solutions to cross the constraint boundary. The mechanism to enforce the approach of the optimum stems from nature. Individuals of different sex are selected by different criteria and nature allows pairing only between individuals of different sex. Transferring this principle to constraint-handling means: Every individual of the two sexes evolution strategy (TSES) is assigned to a feature called *sex*. Similar to nature, individuals with different sexes are selected according to different criteria. Individuals with sex *o* are selected by the objective function. Individuals with sex *c* are selected by the fulfillment of constraints. The intermediary recombination operator plays a key role. Recombination is only allowed between parents of different sex. A few modifications are necessary to prevent an explosion of the step size, that is, a two-step selection operator for individuals of sex *c* similar to the operator by Hoffmeister and Sprave [21]. For a list of TSES variants and modifications we refer to [5]. The populations

Table 3: Experimental results of the death penalty step control evolution strategy.

DSES	Type	best	Mean	dev	ffe	cfe
TR2	$[15; 0.5]$	$3.7 \cdot 10^{-9}$	$8.5 \cdot 10^{-6}$	$2.5 \cdot 10^{-6}$	1,253,394	2,315,574
2.40	$[100; 0.7]$	$1.9 \cdot 10^{-11}$	$2.7 \cdot 10^{-10}$	$7.9 \cdot 10^{-10}$	89,832	1,118,490

Table 4: Experimental results of the two-sex evolution strategy on TR2 and 2.40.

TSES	Type	κ	best	mean	dev	ffe/cfe
TR2	(8+8,10+90)	200	$5.4 \cdot 10^{-8}$	$2.9 \cdot 10^{-7}$	$4.7 \cdot 10^{-8}$	521,523
2.40	(8+8,13+87)	50	0.0	0.0	$3.7 \cdot 10^{-11}$	498,594

are noted as $(\mu_o + \mu_c, \lambda_o + \lambda_c)$—the index determines the sex, that is, o for objective function and c for constraints.

Table 4 shows the experimental results of the TSES on problems TR2 and 2.40. While death penalty completely fails on problem 2.40, the (8 + 8, 13 + 87)-TSES reaches the optimum in every run. Now, a better approximation of the *harder* problem TR2 is possible. Nevertheless, the approximation quality may still be improved and an analysis on further test problems—that can be found in [4]—shows that the TSES is successful on many constrained problems, but not on all. Fortunately, the TSES is quite robust to the chosen population ratios.

We can summarize that the *two-sex evolution strategy* improves the approximation of optima with active constraints, allows infeasible starting points, saves constraint function evaluations, for example, in comparison to the DSES, and is quite robust to parameter changes. But the disadvantages are that the *two-sex evolution strategy* still consumes many fitness function evaluations; on some problems it may still suffer from low success rates, for example, on TR2.

4. Coordinate Alignment Techniques

In real-valued optimization the coordinate system plays an important role. If the coordinate system of the mutation operators, for example, of Gaussian mutation, is not aligned to the coordinate system of the objective function—and this is frequently the case in black-box optimization—undesirable effects may occur like premature step-size reduction.

4.1. Premature Step-Size Reduction. The phenomenon of premature step-size reduction at the constraint boundary has been analyzed in [2]—in particular for the condition that the optimum lies on the constraint boundary or even in a vertex of the feasible search space. In such cases the evolutionary algorithm frequently suffers from low success probabilities near the constraint boundaries. Under simple conditions, that is, a linear objective function, linear constraints, and a comparably simple mutation operator, the occurrence of premature convergence due to a premature decrease of step sizes was proven. Figure 1 illustrates the reason for premature step size reduction. We assume the simplified case in which mutations are produced on the boundary of the circles.

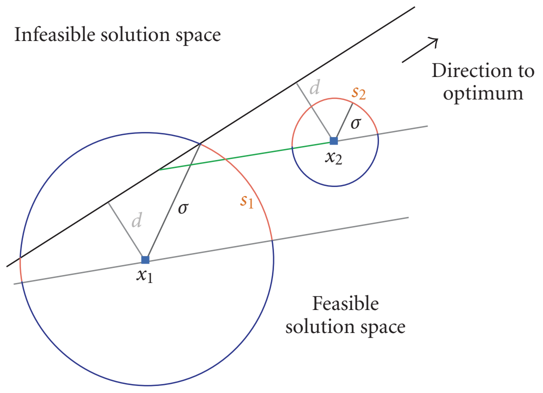


Figure 1: Illustration of the success probabilities at the constraint boundary. In this simplified model we assume that mutations are produced uniformly on the boundary of the circles. Both solutions x_1 and x_2 lie close to the constraint boundary with distance d. In case of small step sizes $\sigma < d$, the success probability $(s_i/(2\pi\sigma))$ is higher than in the case of $\sigma > d$.

in case of large mutation strengths (x_1) with $\sigma > d$ the region of success, that is, the marked part s_1 of the circles, is smaller in comparison to the whole circle than in the case that the circle is not cut by the constraint boundary (x_2). Consequently, the probability to produce successful mutations is higher for small step sizes and these mutations are favored during optimization. This is a coordinate system alignment problem: In case of N independent step sizes and coordinate rotation the mutation circle can adapt to a mutation ellipsoid whose region of success is not restricted by the constraint boundary.

Arnold and Brauer [44] analyzed the behavior at the boundary of linear constraints and models the distance between the search point and the constraint plane with a Markov chain. Furthermore, they discuss the working of step length adaptation mechanisms based on success probabilities.

4.2. Biased Mutation. The shape of the standard mutation ellipsoid is Gaussian. The best modification to improve the success rate situation would be a more flexible mutation distribution function. Later, we will see that a rotation of the mutation ellipsoid is a reasonable undertaking. But is a deformation also an adequate solution to low success rates? Biased mutation aims at biasing the mean of the Gaussian distribution into beneficial directions self-adaptively [7]. A self-adaptive bias coefficient vector ξ determines the direction of this bias and augments the degree of freedom of the mutation operator. This additional degree of freedom improves the success rate of reproducing superior offspring.

The mutation operator adapts the bias direction within the interval -1 (for left) and 1 (for right) in each of the N dimensions:

$$\xi = (\xi_1, \ldots, \xi_N) \quad \text{with} -1 \leq \xi_i \leq 1. \tag{11}$$

This relative direction must be translated into an absolute bias vector. For this sake the step sizes σ_i can be used. For every $i \in 1, \ldots, N$ the bias vector $\mathbf{b} = (b_1, \ldots, b_N)$ is defined by

$$b_i = \xi_i \cdot \sigma_i. \tag{12}$$

Since the absolute value of bias coefficient ξ_i is less than or equal to 1, the bias will be bound to the step sizes σ_i. This restriction prevents the search from being biased too far away from the parent. Hence, the biased mutation works as follows:

$$\begin{aligned}\mathbf{x}' &= \mathbf{x} + (\sigma_1 \mathcal{N}_1(0,1) + b_1, \ldots, \sigma_N \mathcal{N}_N(0,1) + b_N) \\ &= \mathbf{x} + (\sigma_1 \mathcal{N}_1(\xi_1, 1), \ldots, \sigma_N \mathcal{N}_N(\xi_N, 1)).\end{aligned} \tag{13}$$

To allow self-adaptation, the bias coefficients are mutated in the following *meta-EP* way:

$$\xi_i' = \xi_i + \gamma \cdot \mathcal{N}(0,1), \quad i = 1, \ldots, N, \tag{14}$$

with parameter γ determining the mutation strength on the bias. The biased mutation operator (BMO) biases the mean of mutation and enables the ES to reproduce offspring outside the standard mutation ellipsoid. To direct the search, the biased mutation enables the center of the ellipsoid to move within the bounds of the regular step sizes σ. An *adaptive* variant of the originally *self-adaptive* biased mutation is the descent mutation operator. It estimates the descent direction of two population's centers χ_t and χ_{t+1} of successive generations. Let χ_t be the center of the population at generation t:

$$\chi_t = \sum_{i=1}^{\mu} \mathbf{x}_i. \tag{15}$$

The normalized descent direction ξ of two successive population centers χ_t and χ_{t+1} is

$$\xi = \frac{\chi_{t+1} - \chi_t}{\left|\chi_{t+1} - \chi_t\right|}. \tag{16}$$

Similar to the BMO, the descent mutation operator (DMO) now becomes

$$\mathbf{x}' = \mathbf{x} + (\sigma_1 \mathcal{N}_1(\xi_1, 1), \ldots, \sigma_N \mathcal{N}_N(\xi_N, 1)). \tag{17}$$

The DMO is reasonable as long as the assumption of locality is true: the estimated direction of the global optimum can be derived from local information, that is, the descent direction of two successive populations' centers. Again, we analyze both biased mutation operators on the test problems 2.40 and TR2 and show the results in Table 5. For the sake of adaptation of the bias an increase of offspring individuals to $\lambda = 300$ is necessary. The bias mutation parameter is set to the standard setting $\gamma = 0.1$. Our experiments show that the BMO and the DMO are both able to improve the results on problem 2.40. The experiments reveal that the mutation distribution deformation improves the success rate—intuitively by shifting the center of the mutation ellipsoid so that the latter is not cut off by the infeasible solution space. But the results show that the *harder* problem TR2 is still not easy to approximate.

Table 5: Experimental results of the biased mutation variants BMO and DMO.

BMO	best	mean	dev	ffe	cfe
TR2	$1.6 \cdot 10^{-6}$	$9.0 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$	26,832	25,479
2.40	$8.2 \cdot 10^{-12}$	$2.2 \cdot 10^{-7}$	$2.4 \cdot 10^{-8}$	459,774	508,387
DMO	best	mean	dev	ffe	cfe
TR2	$8.8 \cdot 10^{-9}$	$4.6 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$	31,506	29,196
2.40	$1.6 \cdot 10^{-11}$	$1.2 \cdot 10^{-9}$	$2.8 \cdot 10^{-10}$	358,954	359,545

We can conclude that *biased mutation* improves the approximation of optima with active constraints. *Descent biased mutation* is comparatively efficient, in particular more efficient than the BMO. But the disadvantages are that *biased mutation* consumes many fitness and constraint function evaluations, and on some problems it may still suffer from low success rates.

4.3. Mutation Ellipsoid Rotation. Correlated mutation by Schwefel [45] rotates the axes of the hyperellipsoid to adapt to local properties of the fitness landscape. Three ways are possible to rotate the mutation ellipsoid with the help of $N_\alpha = N(N-1)/2$ possible rotation angles:

(1) a self-adaptive rotation—in this case the N_α rotation angles become strategy parameters and the algorithm has to tune itself,

(2) a rotation with the help of a coevolutionary approach,

(3) with a metamodel of the constraint boundary that delivers the orientation of the constraint boundary.

Table 6 shows the experimental results of self-adaptive correlated mutation (SA-ES), a metaevolutionary approach ((3,15(3,15))-MA-ES) [5], and correlated mutation using the metamodel estimator (MM-ES) with 10 and 30 binary search steps. Correlated mutations make use of N_α additional strategy parameters, that is, angles for the rotation of the hyperellipsoid. The self-adaptation process of the SA-ES fails to adapt the angles automatically. The parameter space of N step sizes and N_α angles is too large to adapt successfully by means of self-adaptation. The MA-ES is a nested ES, that is, an outer ES evolves the angles of an inner ES that optimizes the problem itself. Of course, this approach is rather inefficient—as one fitness evaluation of the outer ES causes a whole run of the inner ES on the original problem—but the results demonstrate that the rotation of the hyperellipsoid has a strong impact on

Table 6: A comparison of correlated mutation, metaevolution, and the metamodel-based ellipsoid rotation on TR2.

	SA-ES	MA-ES	MM-ES (10)	MM-ES (30)
Best	$1.6 \cdot 10^{-8}$	0	$2.9 \cdot 10^{-11}$	0.0
Mean	$2.4 \cdot 10^{-4}$	0	$1.6 \cdot 10^{-6}$	0.0
Dev	$3.5 \cdot 10^{-4}$	$3.1 \cdot 10^{-16}$	$5.9 \cdot 10^{-6}$	0.0
Ffe	22,445	927,372	18,736	11,998
Cfe	39,921	1,394,023	32,960	20,183

Table 7: Experimental analysis of the CMA-ES with death penalty.

CMA-ES (DP)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$5.8 \cdot 10^{-16}$	6,754	12,019
2.40	0.0	0.0	$1.3 \cdot 10^{-13}$	19,019	71,241

the approximation capabilities on problem TR2. The MM-ES approach is capable of estimating the proper rotation angle and controlling the mutation ellipsoid to approximate the optimum. We use the linear metamodel that will be introduced in Section 5. The N_α rotation angles can be computed estimating the normal vector $\mathbf{n}_h$ of the estimated hyperplane h and the axes of the mutation ellipsoid. This is an easy undertaking in two dimensions. A comparison between the MM-ES approach with 10 and with 30 binary search steps shows that it is advantageous to invest search for a precise metamodel estimation: a higher accuracy of the metamodel delivers better approximation results.

Obviously, the coordinate system alignment problem is solved with the mutation ellipsoid rotation. But the self-adaptive rotation does not lead to satisfying results, while the metaevolutionary approach is inefficient. In the following paragraph we will investigate whether the covariance matrix adaptation techniques, which are designed to align coordinate systems, are able to adapt their covariance matrix to constrained problems automatically without a metamodel.

4.4. Covariance Matrix Techniques. Past research on constraint-handling missed to concentrate on covariance matrix adaptation techniques. It is an astonishing fact that no sophisticated constraint-handling techniques for these algorithms have been introduced so far. Nevertheless, we will now analyze whether the coordinate system alignment problem can be solved with covariance matrix adaptation using death penalty. The idea of covariance matrix adaptation techniques is to adapt the distribution of the mutation operator such that the probability to reproduce steps that led to the actual population increases. This idea is similar to the estimation of distributions approaches. The covariance matrix adaptation evolution strategy (CMA-ES) was introduced by Hansen [46] and Ostermeier [47]. The results of the CMA-ES on problems TR2 and 2.40 can be found in Table 7. Amazingly, the CMA-ES is able to cope with the low success rates around the optimum of the TR problem. We observed that the average number of infeasible solutions during the approximation is 44%. This indicates that a reasonable adaptation of the mutation ellipsoid takes place. An analysis of the angle between the main axis of the mutation ellipsoid and the constraint function shows that it converges to zero, the same do the step sizes during approximation of the optimum. Hence, the coordinate system alignment is successful.

We can conclude that the CMA-ES is able to align the coordinate system automatically without a metamodel. Recent results have shown that an acceleration can be achieved if the covariance matrix is rotated with the help of a metamodel exactly at the time when the constraint boundary is reached [3].

5. Metamodeling of Constraints

In black-box scenarios the constraint boundaries are not explicitly given. Metamodeling of constraints allows advanced constraint-handling methods. Metamodels can be used for various purposes, for example, for checking the feasibility and for repair of infeasible mutations, and—like we have seen in the previous section—for control of mutation ellipsoids and covariance matrices. Metamodeling of objective functions has developed to a successful standard in evolutionary optimization [48–50].

5.1. Linear Constraint Estimation. For constraint metamodeling various classification and regression methods can be applied. For the case of linear constraints a metamodel that is based on sampling N infeasible points and binary search on the segments to the last feasible point has been developed [3]. The approach works as follows: first, the center point of the model estimator is determined. When the first infeasible offspring individual q_1 is produced, the feasible parent $\mathbf{x}_f$ is the center of the corresponding metamodel estimator and the distance becomes radius r of the model estimator. Then, random points are generated on the surface of a hypersphere. Point $\mathbf{x}_f$ is the center of a hypersphere with radius r, such that the constraint boundary is cut. In N dimensions $N-1$ additional infeasible points q_i, $1 \leq i \leq N-1$ have to be produced. The model estimator produces the infeasible points by sampling on the surface of a hypersphere with radius r until a sufficient number of infeasible points are produced. The points on the surface are sampled randomly with uniform distribution using the method of Marsaglia [51]. In the first step the algorithm produces $N-1$ Gaussian distributed points and scales the numbers to length 1. Further scaling and shifting yields N randomly distributed point on the hypersphere surface.

In a next step the binary search procedure is applied to identify N points $s_1, \ldots, s_N$ *on* the constraint boundary: the line between the feasible point $\mathbf{x}_f$ and the ith infeasible point q_i cuts the real constraint hyperplane h^* in point s_i^*. We approximate s_i^* with binary search on this segment. The center s_i of the last interval defined by the last points of the binary search is an estimation of point s_i^* on h_0. Figure 2 illustrates the situation. With regard to the estimated angle error ϕ, the real hyperplane lies between h_1^* and h_2^*.

In the last step we calculate the normal vector $\mathbf{n}_0$ of h_0 using the N points on the constraint boundary. We assume that the points s_i, $1 \leq i \leq N$, represent linearly independent vectors as the endpoints of the lines they lie on have been generated in a random procedure. A successive Gram-Schmidt orthogonalization of the $(i+1)$th vector on the ith previously produced vectors delivers the normal vector $\mathbf{n}_0$ of h_0. Note that we estimate the normal vector $\mathbf{n}_0$ of the linear constraint model h_0 only one time, that is, when the first infeasible solutions have been detected. Later update steps only concern the local support point p_t of the hyperplane (hence, in iteration t the linear model h_t is specified by normal vector $\mathbf{n}_0$ and current support point p_t). At the beginning, any of the points s_i may be the support point p_0. For later update steps two cases have to be distinguished. Let d_{t_0} be the distance between the mutation ellipsoid center c_{t_0} and the constraint boundary h_{t_0} at time t_0 and let k be the number of binary search steps to achieve the angle accuracy of $\delta < 0.25^\circ$.

(1) The search (i.e., the center of the mutation ellipsoid) c_t approaches h_t: if distance d_t between h_t and c_{t_0} becomes smaller than $d_{t_0}/2^k$, a reestimation of the support point p_t is reasonable.

(2) The search c_t moves parallel to h_t: an exceeding of distance

$$c_{t_0} - c_t = \sqrt{\frac{1}{\tan(\phi)^2} + 4 \cdot d_{t_0}} \qquad (18)$$

with $\phi = 0.25 \cdot (0.57)^{3k}$ causes a reestimation of h_t.

We use $4k$ binary steps on the line between the current infeasible solutions and c_t to find the new support point p_t.

For nonlinear constraints other regression or classification techniques may be taken into account like support vector regression or support vector machines [52].

5.2. Feasibility Check. A metamodel can be used to check the feasibility of new solutions in order to reduce constraint function evaluations [3]. For this purpose an exact estimation of the constraint boundary is necessary. Potentially feasible solutions are checked for feasibility with a real evaluation of the constraint function. Two errors for the feasibility prediction of individual $\mathbf{x}_t$ are possible.

(1) The model predicts that $\mathbf{x}_t$ is feasible, but it is not. Points of this category are examined for feasibility. This will cause an unnecessary constraint function evaluation.

(2) The model predicts that $\mathbf{x}_t$ is infeasible, but it is feasible. The individual will be discarded, but may be a very good approximation of the optimum.

Exemplarily, we take the linear constraint metamodel of the previous paragraph into account and test the feasibility check approach. A *safety margin* δ can reduce the number of errors of type 2. We set δ to the distance d of the mutation ellipsoid center c and the estimated constraint boundary h_t. Hence, the distance between c and the shifted constraint boundary h'_t becomes $2d$. A regular update of the constraint boundary support point p_t is necessary; see previous Section 5.1. Table 8 shows the results of the CMA-ES with feasibility check using the constraint metamodel.

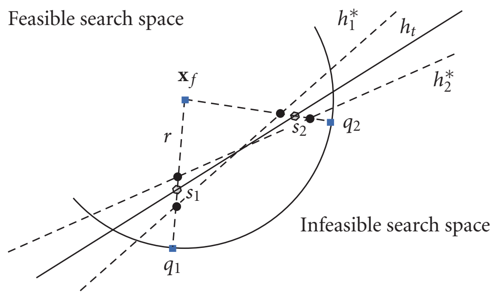


Figure 2: Procedure to estimate the constraint boundary h_0 in two dimensions: the method performs binary search on the segments between a feasible point $\mathbf{x}_f$ and each two infeasible points q_1, q_2 to estimate two points s_1, s_2 on the metamodel.

Table 8: Results of the CMA-ES with feasibility check based on the linear metamodel.

CMA-ES (check)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$6.9 \cdot 10^{-16}$	6,780	7,781
2.40	0.0	0.0	$1.8 \cdot 10^{-13}$	19,386	34,254

Table 9: Results of the CMA-ES with repair mechanism based on the linear metamodel.

CMA-ES (repair)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$5.5 \cdot 10^{-16}$	3,432	5,326
2.40	0.0	0.0	$9.1 \cdot 10^{-14}$	16,067	75,705

We can observe a significant saving of fitness and constraint evaluations with a high approximation capability.

5.3. Solution Repair. The repair approach projects infeasible mutations onto the constraint boundary h_t. We assume the angle error ϕ that can be estimated by the number of binary search steps k. In the solution repair approach the projection vector is elongated by length δ. Figure 3 illustrates the situation. Let p_t be the support point of the hyperplane h_t at time t and let x_i be the infeasible solution. It holds that $a^2 + b^2 = d^2$ and $\delta/b = \tan\phi$. We get

$$\delta = \sqrt{a^2 - d^2} \cdot \tan\phi. \qquad (19)$$

The elongation of the projection into the potentially feasible region guarantees feasibility of the repaired individuals. Nevertheless, it might prevent fast convergence, in particular in regions far away from the hyperplane support point p_t as δ grows with increasing length of d. The center of the hyperplane is updated every 10 generations. The results of the CMA-ES repair algorithm can be found in Table 9. We observe a significant decrease of fitness function evaluations, in particular on problem TR2. The search concentrates on the boundary of the infeasible search space, in particular on the feasible site.

6. Summary

Many constraint-handling methods exist for evolution strategies, at the head penalty functions. Due to low success

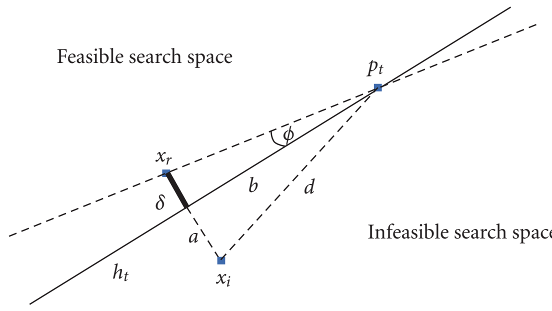


Figure 3: The elongation of the projection of infeasible solution x_i onto the estimated constraint boundary h_t by length δ ensures that the repaired point x_r is feasible.

Table 10: Results of the CMA-ES with covariance matrix rotation, feasibility check, and repair mechanism [3].

CMA-ES (all)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$5.1 \cdot 10^{-16}$	3,249	3,650
2.40	0.0	0.0	$9.1 \cdot 10^{-14}$	11,216	30,069

rates at the constraint boundary, ES without coordinate alignment techniques often fail to find the optima in the vertex of the feasible solution space. The death penalty step control approach and the multiobjective biologically inspired two-sex ES prevent a premature step-size reduction on some problems, but its success depends on proper parameter settings. Low success rates at the constraint boundary can be increased with coordinate system alignment techniques. A first step into this direction is biased mutation techniques, that is, biased mutation and descent biased mutation. Much better results can be achieved with metamodel-based mutation ellipsoid rotation. This rotation cannot be achieved self-adaptively, but automatically with covariance matrix adaptation mechanisms. The latter shows excellent results, even on hard problems like TR2. Further improvements of the CMA-ES can be achieved with metamodeling: constraint boundary surrogates can be used for prediction of feasibility of mutations and for repair of infeasible solutions. At last, Table 10 summarizes the best results that could be achieved on the two problems combining the CMA-ES with covariance matrix rotation, feasibility check, and repair of infeasible solutions.

Metamodeling of constraints will probably become more and more important for future research. Nonlinear models will increase the accuracy of feasibility prediction. Advanced regression methods will improve the accuracy of repaired infeasible solutions. Further constraint-handling methods are imaginable like adaptation of mutation probability distributions and covariance matrices—also with non-linear metamodels.

References

[1] C. Floudas and P. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer, Berlin, Germany, 1990.

[2] O. Kramer, "Premature convergence in constrained continuous search spaces," in *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN '08)*, pp. 62–71, Springer, Dortmund, Germany, 2008.

[3] O. Kramer, A. Barthelmes, and G. Rudolph, "Surrogate constraint functions for CMA evolution strategies," in *Proceedings of the 32nd German Annual Conference on Artificial Intelligence (KI '09)*, pp. 169–176, Paderborn, Germany, September 2009.

[4] O. Kramer, S. Brugger, and D. Lazovic, "Sex and death: towards biologically inspired heuristics for constraint handling," in *Proceedings of the 9th Conference on Genetic and Evolutionary Computation (GECCO '07)*, pp. 666–673, ACM Press, London, UK, July 2007.

[5] O. Kramer and H.-P. Schwefel, "On three new approaches to handle constraints within evolution strategies," *Natural Computing*, vol. 5, no. 4, pp. 363–385, 2006.

[6] O. Kramer, C.-K. Ting, and H. Kleine Büning, "A mutation operator for evolution strategies to handle constrained problems," in *Proceedings of the 7th Conference on Genetic and Evolutionary Computation (GECCO '05)*, pp. 917–918, Washington, DC, USA, June 2005.

[7] O. Kramer, C.-K. Ting, and H. Kleine Büning, "A new mutation operator for evolution strategies for constrained problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 2600–2606, Edinburgh, UK, September 2005.

[8] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, Germany, 1973.

[9] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen Mittel der Evolutionsstrategie*, Birkhäuser, Basel, Switzerland, 1977.

[10] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[11] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.

[12] J. C. Bean and A. B. Hadj-Alouane, "A dual genetic algorithmfor bounded integer programs," Tech. Rep., University of Michigan, Kalamazoo, Mich, USA, 1992.

[13] A. Fiacco and G. McCormick, "The sequential unconstrained minimization technique for nonlinear programming—a primal-dual method," *Management Science*, vol. 10, pp. 360–366, 1964.

[14] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[15] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, D. B. Fogel, Ed., pp. 579–584, IEEE Press, Orlando, Fla, USA, June 1994.

[16] A. Kuri-Morales and C. V. Quezada, "A universal eclectic genetic algorithm for constrained optimization," in *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT '98)*, pp. 518–522, Mainz, Aachen, Germany, September 1998.

[17] R. G. L. Riche, C. Knopf-Lenoir, and R. T. Haftka, "A segregated genetic algorithm for constrained structural optimization," in *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA '95)*, L. J. Eshelman, Ed., pp. 558–565, University of Pittsburgh, Morgan Kaufmann, San Francisco, Calif, USA, July 1995.

[18] C. A. Coello Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[19] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, pp. 971–978, 2001.

[20] D. Powell and M. M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 424–431, University of Illinois at Urbana-Champaign, Morgan Kaufmann, San Francisco, Calif, USA, July 1993.

[21] F. Hoffmeister and J. Sprave, "Problem-independent handling of constraints by use of metric penalty functions," in *Proceedings of the 5th Conference on Evolutionary Programming (EP '96)*, L. J. Fogel, P. J. Angeline, and T. Bäck, Eds., pp. 289–294, MIT Press, Cambridge, UK, February 1996.

[22] A. I. Oyman, K. Deb, and H.-G. Beyer, "An alternative constraint handling method for evolution strategies," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 1, pp. 612–619, IEEE Service Center, Piscataway, NJ, USA, July 1999.

[23] T. P. Runarsson, "Approximate evolution strategy using stochastic ranking," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 2760–2767, IEEE, Vancouver, Canada, July 2006.

[24] S. V. Belur, "CORE: constrained optimization by randomevolution," in *Proceedings of the Late Breaking Papers at the Genetic Programming Conference*, J. R. Koza, Ed., pp. 280–286, Stanford University, Stanford, Calif, USA, July 1997.

[25] C. A. Coello Coello, "Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[26] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[27] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, Springer, Berlin, Germany, 2000.

[28] J. Paredis, "Co-evolutionary constraint satisfaction," in *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature (PPSN '94)*, pp. 46–55, Springer, Jerusalem, Israel, October 1994.

[29] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility," in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature (PPSN '96)*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., pp. 245–254, Berlin, Germany, September 1996.

[30] C. A. Coello Coello, "Constraint handling through a multiobjective optimization technique," in *Proceedings of the Genetic and Evolutionary Computation Conference*, A. S. Wu, Ed., pp. 117–118, Orlando, Fla, USA, July 1999.

[31] C. A. Coello Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Engineering Optimization*, vol. 32, no. 3, pp. 275–308, 2000.

[32] F. Jimenez and J. L. Verdegay, "Evolutionary techniques for constrained optimization problem," in *Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT '99)*, H.-J. Zimmermann, Ed., Mainz, Aachen, Germany, September 1999.

[33] E. Mezura-Montes and C. A. Coello Coello, "Constrained optimization via multiobjective evolutionary algorithms," *Multi-Objective Problem Solving from Nature: From Concepts to Applications*, pp. 53–75, 2008.

[34] I. C. Parmee and G. Purchase, "The development of a directed genetic search technique for heavily constrained design spaces," in *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control (PEDC '94)*, I. C. Parmee, Ed., pp. 97–102, University of Plymouth, Plymouth, UK, September 1994.

[35] M. Schoenauer and S. Xanthakis, "Constrained GA optimization," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 573–580, Morgan Kaufman, San Francisco, Calif, USA, July 1993.

[36] P. D. Surry, N. J. Radcliffe, and I. D. Boyd, "Amulti-objective approach to constrained optimisation of gas supply networks: the COMOGA Method," in *Proceedings of the Evolutionary Computing, AISB Workshop*, T. C. Fogarty, Ed., Lecture Notes in Computer Science, pp. 166–180, Springer, Sheffield, UK, April 1995.

[37] E. M. Montes and C. A. Coello Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, 2005.

[38] T. Takahama and S. Sakai, "Constrained optimization by the e constrained differential evolution with gradient-based mutation and feasible elites," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, G. G. Yen, S. M. Lucas, G. Fogel, et al., Eds., pp. 1–8, IEEE Press, Vancouver, Canada, July 2006.

[39] J. Liang and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *Proceedings of the IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, et al., Eds., pp. 9–16, IEEE Press, Vancouver, Canada, July 2006.

[40] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, et al., Eds., pp. 25–32, Vancouver, Canada, July 2006.

[41] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Computation*, vol. 198 of *Studies in Computational Intelligence*, Springer, Berlin, Germany, 2009.

[42] H.-P. Schwefel, *Evolutionsstrategie und numerische optimierung*, Ph.D. thesis, TU Berlin, Berlin, Germany, 1975.

[43] H.-P. Schwefel, *Evolution and Optimum Seeking. Sixth-Generation Computer Technology*, Wiley Interscience, New York, NY, USA, 1995.

[44] D. V. Arnold and D. Brauer, "On the behaviour of the (1+1)-es for a simple constrained problem," in *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature (PPSN '08)*, pp. 1–10, Dortmund, Germany, September 2008.

[45] H.-P. Schwefel, "Adaptive mechanismen in der biologischen evolution und ihr einfluss auf die evolutionsgeschwindigkeit," in *Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik*, TU Berlin, Berlin, Germany, July 1974.

[46] N. Hansen, "The cma evolution strategy: a tutorial," Tech. Rep., TU Berlin, ETH Zürich, Germany, 2005.

[47] A. Ostermeier, A. Gawelczyk, and N. Hansen, "A derandomized approach to self adaptation of evolution strategies," *Evolutionary Computation*, vol. 2, no. 4, pp. 369–380, 1994.

[48] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, "Metamodel-assisted evolution strategies," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN '02)*, pp. 361–370, Granada, Spain, September 2002.

[49] S. Kern, N. Hansen, and P. Koumoutsakos, "Local metamodels for optimization using evolution strategies," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06)*, pp. 939–948, Reykjavik, Iceland, 2006.

[50] H. Ulmer, F. Streichert, and A. Zell, *Optimization by Gaussian Processes Assisted Evolution Strategies*, Springer, Heidelberg, Germany, 2003.

[51] G. Marsaglia, "Choosing a point from the surface of a sphere," *The Annals of Mathematical Statistics*, vol. 43, pp. 645–646, 1972.

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, Berlin, Germany, 2009.

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 413179, 12 pages
doi:10.1155/2010/413179

Review Article

A Review of Gait Optimization Based on Evolutionary Computation

Daoxiong Gong, Jie Yan, and Guoyu Zuo

School of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China

Correspondence should be addressed to Daoxiong Gong, gongdx@bjut.edu.cn

Received 1 September 2009; Revised 21 March 2010; Accepted 22 April 2010

Academic Editor: Oliver Kramer

Gait generation is very important as it directly affects the quality of locomotion of legged robots. As this is an optimization problem with constraints, it readily lends itself to Evolutionary Computation methods and solutions. This paper reviews the techniques used in evolution-based gait optimization, including why Evolutionary Computation techniques should be used, how fitness functions should be composed, and the selection of genetic operators and control parameters. This paper also addresses further possible improvements in the efficiency and quality of evolutionary gait optimization, some problems that have not yet been resolved and the perspectives for related future research.

1. Introduction

Compared to wheeled robots, legged robots usually possess superior mobility in uneven and unstructured environments. This is because they can use discrete footholds to overcome obstacles, climb stairs, and so forth, instead of relying on a continuous support surface.

A gait is a cyclic, periodic motion of the joints of a legged robot, requiring the sequencing or coordination of the legs to obtain reliable locomotion. In other words, gait is the temporal and spatial relationship between all the moving parts of a legged robot [1]. Gait optimization is very important for legged robots, because it determines the optimal position, velocity and acceleration for each Degree of Freedom (DOF) at any moment in time, and the gait pattern will directly affect the robot's dynamic stabilization, harmony, energy dissipation and so on. Gait optimization determines a legged robot's quality of movement.

2. Why Evolutionary Computation Is Suitable for Gait Optimization

2.1. Gait Generation Is a Multiconstrained, Multiobjective Optimization Problem. Gait generation, which incorporates mobility and stability, is a very challenging task for legged robots, because their system of locomotion has multiple DOFs and a variable mechanical structure during locomotion. As a result a large number of parameters have to be established. For example, 54 motion parameters have to be considered for the walk gait of the Sony AIBO robot [2]. To obtain a natural and efficient gait for a legged robot, two kinds of strategies for sequencing or coordination of the leg movements can be followed.

The first strategy assumes that the gaits of humans or animals are optimal, as otherwise they would not have been able to survive the competition and natural selection proposed by Darwin's Theory of Evolution. This assumption has been proved accurate [3]. The constrained optimization hypothesis suggests that gait parameters are selected to optimize (minimize) the objective function of the cost of transport (metabolic cost/distance) within the limitations of imposed constraints [4]. A lot of research has shown that humans and animals move in a way that minimizes the metabolic cost of locomotion and validates the idea that the gait synthesis of legged robot is a constrained optimization problem [5–12].

Robots simulate human or animal behavior [13]. Therefore, it is quite natural to use biological locomotion data to control the gait of robots. For example, Human Motion Captured Data has been adopted to drive a humanoid robot [14]. However, some research indicates that biological locomotion data cannot be used directly for a legged robot due to kinematic and dynamic inconsistencies between humans/animals and the legged robot. This implies the need for kinematic corrections when calculating joint angle trajectories [14].

The second strategy formulates the gait generation problem of the legged robot as an optimization problem with constraints. It generates the optimal gait cycle by minimizing some performance indexes, for example, velocity of motion, stability criteria, actuating forces, energy consumption, and so forth. The gait generation problem of legged robots often has several objectives, and some of these objectives may be contradictory to each other (for example, speed and stability). Thus the gait generation problem can be stated as a multi-constrained and multi-objective optimization problem [15].

These two gait generation strategies may reach the same goal by different routes because both of them actually solve the gait synthesis problem as a multi-constrained multi-objective optimization problem. Once a database of precomputed optimal gaits has been created, the robot can cover the entire interval of precomputed optimal gaits by interpolation and thus realize smooth real time locomotion.

2.2. Evolutionary Computation Is Suitable for the Gait Optimization Problem. The dynamic equations of legged robot locomotion are high order highly coupled and nonlinear, and gait optimization for legged robots requires searching a set of parameters in a highly irregular, multidimensional space. As a result, the standard gradient search-based optimization methods are not useful for legged systems with high DOF [2, 16, 17].

Evolutionary Computation (EC), including the Genetic Algorithm (GA), Genetic Programming (GP), Evolutionary Programming (EP), and Evolutionary Strategy (ES), is a natural choice for the gait optimization of legged robots.

First, EC uses optimization methods based on Darwin's Natural Evolution Theory. According to this theory, the locomotion mechanisms of life forms resulted from natural selection and the interaction between individuals and the natural environment. This makes the use of EC a natural choice, as it is biologically inspired and can generate biologically plausible solutions [18].

Second, from the computational point of view, EC also fits well with the gait optimization of legged robots [2, 18, 19], because of the following:

(a) Gait optimization problems can have multiple criteria, multiple constraints, as well as multiple design variables, and EC has been shown effective for these kinds of large-dimension, multi-objective, multi-constraint optimization problems.

(b) EC has been seen to be robust for search and optimization problems and has been used to solve difficult problems with objective functions where local information such as continuity, differentiability, and so on is not available, even though it is very important for gait optimization, as the objective functions of gait optimization may be very complex and it is very difficult to obtain this local information.

(c) Because of the complexity and high DOF of the mechanical structure, it is difficult to obtain a precise dynamic model of a legged robot [20]. EC will be efficient as this method is resistant to noise in the evaluation function and offers a model-free approach to optimization, only requiring feedback from the environment to improve performance when online evolution is deployed with a real robot.

(d) EC has strong global search capability and is also insensitive to the initial population. Therefore EC decreases the risk of being trapped in a local minimum for finding a true optimum solution.

(e) EC can easily be parallelized. Since gait optimization of legged robots is often a large-scale problem and the objective function and constraints are often complex, the process of evolutionary optimization may be very time-consuming because of the high computational cost of EC due to iterative evaluations of candidate solutions. Therefore it is advantageous to use parallel implementations of EC to gain efficiency and improve the solution quality of EC-based gait optimization.

3. How to Evolve the Optimal Gait

3.1. The Multiform EC Models Adopted in Gait Optimization. Gait optimization based on EC is actually a combination of EC procedures and gait optimization problems. A general block diagram of EC-based gait optimization is given in Figure 1. This offers a first glance at the application of EC technique for gait optimization of legged robots.

A lot of EC models have been adopted to solve gait optimization problems. The gaits most often studied include the gaits of biped, quadruped, and hexapod robots engaged in walking, running, negotiating sloping surfaces, and going up and down stairs [21–26].

The Genetic Algorithm (GA) is the gait optimization tool which is most often used, and some modifications can be introduced to fit the specific problems of gait optimization [25, 27, 28]. For example, interpolating and extrapolating operators [29], two-point crossover, Gaussian mutation, overlapping populations [2], and Elitism strategy have been adopted. The explicit fitness sharing mechanism [30] has also been adopted to prevent premature convergence to suboptimal extremes. This speciation technique divides the population into a fixed number of species, where each species contains individuals that are similar to each other, and can force similar members of the same species to "share" one representative score, thereby penalizing species with a large number of individuals and allowing new species to form even if they do not perform as well as other, larger, species [30].

Adaptability that can adaptively change the probabilities of crossover and mutation is introduced in GA to balance global and local exploitation and exploration towards the

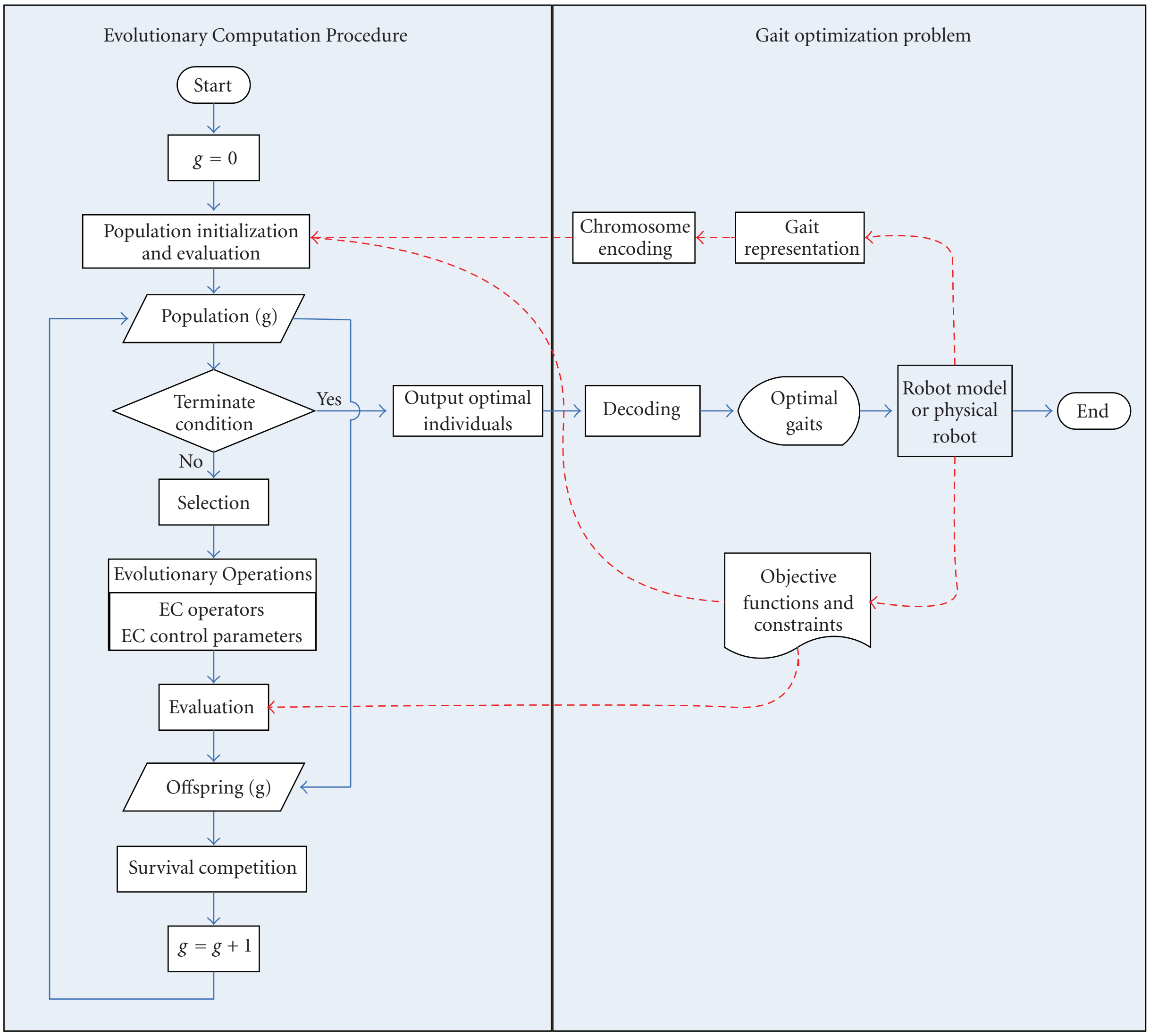


Figure 1: A general block diagram for EC-based gait optimization.

progress of evolutionary optimization. For instance, Adaptive GA is used to optimize the gait of a humanoid robot ascending and descending a staircase by searching optimal trajectory parameters in blending polynomials [31].

Adaptive mechanisms may also be applied to control mutation rate [2]. This method places radiation (the level of radiation decreases over time) into the middle of a region where a large group of individuals is clustered within the same locality to dramatically increase the mutation rate in this area, causing all the individuals to mutate in the next generation and to disperse to other areas of the space. It is reported that this mechanism can be useful in controlling the learning behavior of GA and makes GA more robust with respect to noise in parameter evaluations preventing premature convergence to suboptimal extremes.

Genetic Programming (GP) [32] and Grammatical Evolution (GE) are often used to evolve the gait of robots.

Simulation results obtained using GP on an AIBO quadruped in the Webots environment are reported much better than those obtained using simple GA-based approaches [17]. In this approach, the gait is defined using joint angle trajectories instead of locus of paw to reduce the search space of optimization. An elite archive mechanism (EAM) is used to prevent premature convergence and improve the search capability of GP. EAM can preserve elite individuals at an early stage and flow them into in later stage. In this way, genetic material from elite individuals at an early stage is used to refresh an evolutionary convergent state and to create a role for preserving diversity as long as possible.

GE is one of the most popular forms of grammar-based GP. The advantage of GE lies in that it allows the user to conveniently specify and modify the grammar, whilst ignoring the task of designing specific genetic search operators. Thus GE can be used to optimize pre-existing motion data or generate novel motions. Using a Fourier gait representation to encode the chromosome and the dynamic similarity principles as a constraint, GE is employed to optimize the gait retargeting problem in animation. It successfully modified one animal's gait cycle data into a different animal's gait cycle data in computer simulations of animal locomotion [33]. The same method can also be used to optimize the gait of a walking horse from a veterinary publication into a physics-based horse model [34].

ES is also employed to solve the gait optimization problem [35, 36], and some encouraging results have been obtained.

Using a hand-tuned gait as a seed, the bipedal gait is directly evolved on a physical robot by an ES approach with parametric mutation and structural mutation. After hundreds of evaluations significant improvements were obtained for a functioning but nonoptimized bipedal gait that improved the walking speed by around 65% compared to the hand-tuned gait [37].

A hybrid approach of space-time optimization and covariance matrix adaptation evolution strategy (CMA-ES) has been proposed to generate gaits and morphologies for legged animal locomotion [38]. It effectively generated dynamic locomotion gait of bipeds, a quadruped, as well as an imaginary five-legged creature by simulation. The gaits and morphologies produced are reported lifelike and exhibit many qualitative traits seen in real animals. This hybrid approach may combine the efficiency in high-dimensional spaces and the ability to handle general constraints of space-time optimization with the ability to handle nondifferentiable variables and to avoid many local minima from CMA.

Apart from traditional EC and its variations, some relatively new types of EC have also been applied to gait optimization research.

Estimation of Distribution Algorithms (EDAs) are evolutionary algorithms based on probabilistic models that replace the operators of mutation and crossover used in GAs. The main advantage of EDA lies in the fact that the knowledge about the problem acquired previously can be used to set the initial probability model, and the global statistical information about the search space can be extracted directly by EDA to modify the probability model with promising solutions. This can reduce the search space and obtain good solutions in a shorter time interval. For this reason EDA has been used to study the gait optimization problem [15, 39, 40]. For example, EDA has been applied to optimize the gait of the AIBO robot. A fitness function based on direct evaluation of the robots was adopted, and significant improvement of the previous gait was achieved over a short training period [41].

In some cases of gait optimization, the performance of a gait cannot be directly measured or calculated based on certain functions. In this case human preferences, intuition, emotions, and other psychological aspects can be introduced into the target system. Interactive evolutionary computation (IEC) is a form of evolutionary computation where the fitness function can be replaced by the user. A prominent advantage of IEC is that it can reflect user preference and allow optimization of the solution with a minimum of required knowledge in the problem domain [42].

Staged Evolution, which evolves the result in a number of stages, has also been proposed for gait optimization [20, 42, 43]. This approach employs a strategy of divide and conquer. By introducing a staged set of manageable challenges, it decreases the search space and thus improves the convergence rate of EC and obtains rapid evolution of behavior towards a given goal.

The multiobjective multiconstraint problem is often solved by combining the multiple objectives and constraints into a single scalar objective problem using weighting coefficients. To do this, some problem-specific information is needed, and the relative importance of the objectives and constraints should be decided. In the complex problem of gait optimization, it is difficult to know this information in advance. In addition, there is no rational basis for determining adequate weights for these competitive or conflicting criteria, and the objective function that will be formed may lose significance due to the combination of non-commensurable objectives [44]. Therefore, more and more gait optimization problems are parameterized and optimized using tailored Multiple Objective EC procedures [45], for example, the Strength Pareto Evolutionary Algorithm [46] and Nondominated Sorting Genetic Algorithm with Fitness Sharing method [44], and the obtained Pareto-optimal gaits, which is a set of nondominated or noninferior gaits that satisfies different objective functions. These methods have shown good performance [44, 46–48].

3.2. Gait Representation and Chromosome Encoding. The gait of a legged robot may be represented in three-dimensional space [30, 49] or in joint space [17, 20, 50].

In order to control a legged robot's movement, it is necessary to generate the trajectories of all the joints. Therefore, gait is usually represented by a sequence of key poses (states) extracted from one complete gait cycle [51], and phases between these key poses are approximated by a polynomial function, for example, 3rd, 4th, or 5th order polynomials. These polynomial functions are adopted because they can insure that the joint trajectories are smoothly connected with first-order and second-order derivative continuity. First order derivative continuity guarantees the smoothness of velocity, while the second order guarantees the smoothness requirements of acceleration or the torque in the joints [20]. As a result, the gait of robot will look natural.

If only the foot placement point of these key poses is specified, once the foot trajectories are generated, inverse kinematics should then be used to convert the locus of foot into the joint angles required to generate the foot placement curves for a particular gait [17, 20, 30, 47, 52–54].

To make the robot optimally move from its current position/stance to a goal position/stance, other parameters apart from those of the leg joint trajectories should also be considered [47, 55], for example, parameters describing the position and orientation of the body, how the robot's weight shifts during walking, whether or how much the arms swing, and so forth.

The joint angles in these states, the coefficients of the polynomials, and some of the other parameters mentioned above are the design variables to be optimized by EC [18, 39, 56, 57]. These design variables, when treated as genes and arranged in an array, make up a chromosome of EC [16].

A variety of chromosome encoding methods, including the gray code representation [43], real number coding [57], mixed encoding of floating point number, and binary number [1], have been adopted, but the most often used encoding is the real coded method. This is due to difficulties associated with binary representation when dealing with a continuous search space with large dimension [44].

3.3. Composing the Fitness Functions. EC are a family of objective function driven optimization algorithms. The objective/fitness functions represent the problem environment and decide how well the individual solves the problem. Therefore, the construction of a fitness function is very important for the correct functioning of EC, and researchers should define these objective functions appropriately according to the task to be accomplished so that each individual's actual behavior can be evaluated correctly and efficiently.

A lot of criteria can be used to construct the fitness functions of EC for gait optimization, and at present most studies mainly emphasize only one or a part of the following aspects [1].

(a) *Maximum Velocity*. The gait should help the robot attain maximum velocity, so the robot's speed of locomotion is a basic performance index [47, 51, 55, 58].

(b) *Minimum Consumed Energy*. The criterion most often used for gait optimization is the minimum consumed energy (MCE) [1, 20, 39, 40, 44, 46, 56, 57, 59–66] as an energy-efficient locomotion pattern results in a more natural walking motion. In fact, the MCE gait of a biped robot is similar to that of human. Another advantage of MCE criteria is that the consumed energy should be reduced so as to maximize battery operation time.

(c) *Minimum Torque Change*. The criterion of minimum torque change (MTC) is based on smoothness at the torque level [44, 46, 57, 59, 60]. This may result in a more stable motion due to a smoother change in link acceleration.

(d) *Stability*. In order to make the robot move in an environment and avoid falling down, it has to have a stable gait. Stability is the most important constraint and is most often used in gait optimization. Stability can be static or dynamic. Static stability can be verified via the center-of-gravity index [18, 59], while the dynamic stability is often verified via the zero-moment-point (ZMP) [1, 14, 16, 20, 39, 40, 46, 47, 57, 60, 67], which has an important role in gait optimization, especially for the biped gait. If the robot has to focus on how to restore balance rather than constantly trying to maintain dynamic balance, the foot placement estimator (FPE) can be adopted [68].

(e) *Geometric Constraint*. This ensures the feasibility of robot gaits from the point of physical structure [1, 14, 40, 58, 67, 69]. When the robot is passing through obstacles or climbing stairs, the gait should not lead to a collision between the robot and its environment. When the robot walks, the swing limb has to be lifted off the ground at the beginning of the step cycle and has to be landed back at the end of it.

(f) *Smooth Transition Constraint*. To have a continuous periodic motion, the initial posture and velocities should be identical to those at the end of the step [44, 56, 62, 69]. For a humanoid robot, the horizontal displacements of the hip during the single and double support phases must also be continuous. When a walking robot's swing limb makes contact with the ground (heel strike), the effect of impact should be minimized so that it does not influence the motion stability of the robot [1, 14, 40, 57, 58].

More criteria can be added to achieve other practical requirements in gait generation and optimization, and the constraints can be formulated as equalities and inequalities. These criteria will serve as objective functions for evolutionary-based gait optimization.

There are two ways to evolve gait for a robot, namely on-line evolution and off-line evolution. On-line evolution evolves gait directly on a real robot [32, 43, 47, 49, 51, 70, 71], while the off-line method evolves gait on a simulator [18, 59, 72]. In the case of off-line evolution, solutions are evaluated using the objective functions mentioned above. In the case of on-line evolution, the fitness may not be directly calculated, instead it will be determined based on measurements, that is, the solutions have to be tested by letting the robot actually walk with the parameters encoded by the chromosome, and the fitness for each individual is evaluated using the robot's sensors (digital camera, infrared sensor, and gyro-sensor) or directly evaluated by the user [30, 42].

3.4. The Genetic Operators and Control Parameters of EC. In order to make EC work properly, a set of control parameters and some genetic operators, for example, selection, crossover, and mutation should be predefined.

(a) *Selection*. Selection is performed so that better individuals are chosen for breeding and surviving. There are quite a lot of methods for selecting individuals for genetic operations. The mostly commonly used method is the roulette wheel procedure [16, 47], which assigns a higher probability of selection to an individual if its fitness is determined to be better. Tournament selection is another commonly used selection procedure. Binary tournament randomly selects two individuals from the population at each time and chooses the fitter one [42]. Other tournament algorithms simultaneously select individuals as parents and other specific individuals to be replaced [32, 49]. The parent(s) is the individual(s) with higher fitness, and the individual with the lowest fitness is replaced by the offspring of the parent(s). Elitism strategy is commonly adopted. This guarantees that the fittest individuals will always be retained into the next generation.

(b) *Crossover*. Crossover combines the genes of two individuals into a new one. A parameter P_c is used to control how often the crossover operator can be applied, and it can be encouraged by increasing the probability. Usually, $P_c \in [0.6 \sim 0.9]$. All commonly used crossover operators, for example, Simple Crossover, Two-point crossover, Multipoint crossover, Arithmetic crossover, Heuristic Crossover and Uniform crossover, can be used for gait optimization [2, 16, 18, 20, 30, 32, 42–44, 56, 67, 69, 73]. Other methods for crossover may also be used. For example, the interpolation and the extrapolation operators [29], as well as the quaternion recombination techniques of Guaranteed-Uniform-Crossover, and Guaranteed-Average, Guaranteed-Big-Creep, Guaranteed-Little-Creep have been employed [47].

(c) *Mutation*. Mutation introduces perturbation to the genes of an individual and thereby creating a new one. Parameter P_m affects the number of individuals mutated, as well as the number of mutated genes per chromosome. Mutation is performed with a very low probability, usually, $P_c \in [0.005 \sim 0.1]$. All commonly used mutation operators, for example, Uniform Mutation, Nonuniform Mutation, Boundary Mutation, and Gaussian Mutation, can be used for gait optimization [2, 16, 18, 20, 29, 32, 42–44, 67, 69, 73].

(d) *Population Size*. The number of genetic strings maintained at one time may vary from 10 to 800 according to the literature [2, 16, 18, 20, 32, 42–44, 47, 51, 56, 67, 69, 71, 73]. A larger population increases the evaluation time for each epoch, while a smaller population size may not provide enough variation, causing the algorithm to converge to local extremes more often than necessary. Therefore, the population size should be carefully decided according to the size or difficulty of the problem, and a compromise between efficiency of computation and diversity of solution should be made. This is why Chernova suggested that a population size of 30 is a good choice [2]; however Eperješi reported that a population of the same size lost most of the genetic material quite quickly and converged to local minima [42].

(e) *Initial Population*. The initial population can be randomly created in two ways. One generates the initial population by mutation using a hand-tuned gait as a seed [29, 51], and the other generates the initial population with a uniform distribution over the given search range [32, 49, 50, 71].

(f) *Maximum Generation Criterion*. EC will iteratively apply the genetic operations until a certain termination criterion is met. Typical termination criteria employed in EC include: the realization of a predefined total number of iterations, having reached a maximum number of iterations without improvement, and even more complex mechanisms based on estimating the probability of being at a local optimum. Maximum generation is the most often adopted termination criterion in gait optimization. The value of the maximum generation may vary between 30 to 5000 iterations [16, 18, 20, 44, 47, 51, 56, 58, 67, 69, 73]. This should also be carefully determined according to the complexity or difficulty of the problem. Due to the stochastic nature of EC, each evolutionary optimization experiment may have to be repeated a number of times to ascertain the global optima [18].

3.5. The Effect and Efficiency of Evolutionary Gait Optimization. Almost all research, whether it is simulation or physical experiment, has shown that EC achieves good results. The advantages of evolutionary gait optimization include the following.

(i) As the joint torques and link accelerations change smoothly, the final motion of the robot is very smooth, and the impact of the foot with ground is minimal. Therefore, the best-evolved gaits determined by this method outperform the best manually developed gaits in their ability to move straighter and faster, while at the same time being more flexible and reliable [2, 18, 32, 44, 49–51, 57, 71, 74].

(ii) The energy consumption of the optimal motion was dramatically reduced [44, 56, 57]. The optimal gait pattern was natural and very similar to that of humans [44, 59, 73], and battery actuated robots can thus prolong their operation time.

(iii) EC can autonomously and more efficiently search for high performance gait parameters for various surface conditions and different robot platforms compared to using a manual approach [2, 50]. Furthermore, the EC-based approach was able to match the best previously known AIBO gait within a matter of hours even starting from a random population [2]. Multi-objective evolution was able to find optimal humanoid robot gaits with completely different characteristics efficiently in one simulation run [44].

4. Comparing EC with Other Global Optimization Approaches

Besides evolution-based optimization techniques, other global optimization approaches that adopt a non-evolutionary metaphor have also been employed in gait optimization. These also search for the global optimum of the cost function without using the differential information of a given cost function.

Particle Swarm Optimization (PSO) can be used to optimize the stable and straight movement patterns (gaits) of a humanoid robot with the control signals of the joint angles produced by a Truncated Fourier Series (TFS) [75, 76]. It is reported that PSO optimized TFS significantly faster and better than GA to generate straighter and faster humanoid locomotion because PSO bypassed a local minimum that GA was caught in [76]. The authors therefore concluded that PSO is better than GA as a learning method for the gait optimization problem in a non-deterministic environment.

We argue that GA may not necessarily be inferior to PSO in gait optimization, even in a non-deterministic environment such as the one in this experiment. This is because the PSO employed in this experiment was Adaptive PSO, which has a dynamically adjustable nonlinear parameter of inertia weight w to control the balance between global and local exploration. A larger inertia weight facilitates a global search, while a smaller inertia weight facilitates a local search [77]. The GA employed in this experiment is just a canonical paradigm with roulette wheel selection and a fixed rate of crossover and mutation. This may be the reason why PSO can speed up the search and perform better than GA in this experiment.

EC of course can employ the same mechanism to improve its efficiency. For example, Adaptive GA can adaptively change the probabilities of crossover and mutation during the process of evolution. In ES, the step size or mutation strength is often governed by self-adaptation (evolution window), and the individual step sizes for each coordinate or correlations between coordinates are either governed by self-adaptation or also by covariance matrix adaptation (CMA-ES) [38].

Adaptive PSO is used to optimize the fastest forward gaits of the quadruped robot AIBO with the whole learning process running automatically on the physical robot [78]. Starting with randomly generated parameters instead of hand-tuned parameters, several high-performance sets of gait parameters are obtained, and these gaits were reported as being among the fastest forward gaits ever developed for the same robot platform.

Parallel PSO was applied to large-scale human movement problems, and experimental results show that PSO was outperformed by the gradient-based algorithm [19]. It is reported that a single run with a gradient-based nonlinear least squares algorithm produced a significantly better solution than did 10 runs using global PSO. Thus the authors do not recommend using the PSO algorithm for solving large-scale human movement optimization problems possessing constraints or competing terms in the cost function.

The results of this experiment may be a fortunate exception. The objective functions of large-scale gait optimization problems with hundreds of design variables will no doubt be massively multimodal and the landscape must be very rugged. Therefore a gradient-based algorithm will certainly be trapped in a local minimum, and the global search ability of EC is absolutely necessary for decreasing this risk. We agree with the suggestion of the authors that a global local hybrid algorithm may be necessary for PSO and other global optimizers to solve large-scale human movement problems efficiently.

As far as we have seen from the literature, Ant Colony Optimization (ACO) has not yet been used in the field of gait optimization though this too is a famous metaheuristic of Swarm Intelligence (SI) similar to PSO and has been widely used to solve a lot of kinds of optimization problems.

The univariate dynamic encoding algorithm for searches (uDEAS) has also been applied to the gait optimization problem of a biped model walking up and down a staircase. The simulation results show that uDEAS outperforms adaptive GA with a 17 s versus 126 s run time on average and a slightly smaller minimum for best cost values [79]. The authors attribute this result to the effectiveness of describing trajectories with the blending polynomial of uDEAS.

The problem representation method and the genotype encoding method directly determine the size and the characteristic of the search space and as a result directly affect the efficiency of EC optimization. Therefore, we suggest that researchers should pay attention to both the problem representation method and the genotype encoding method when studying the EC-based gait optimization problem. For example, TFS is reported to be a good gait representation approach that can generate suitable angular trajectories for controlling bipedal locomotion. This is because it does not require inverse kinematics, and stable gaits with different step lengths and stride frequencies can be readily generated by changing the value of only one parameter in the TFS [76].

Though some comparison of performance between EC and other non-evolutionary global optimization approaches has been reported, no systematic comparative study has been carried out. Such a systematic comparative study may be not necessary or not feasible because we often search for a set of satisfactory solutions instead of an absolutely global optimal solution. Both the robot platform and the objective functions of gait optimization will be different in each case, and thus it is difficult to find a benchmark robot and a set of benchmark objective functions to optimize.

Both EC and SI approaches are population-based iterative algorithms, even though they adopt different metaphors. Thus they share the same advantages and disadvantages in gait optimization, for example, a similar global and high-dimensional search capability, multi-objective optimization capability, as well as a lot of control parameters which require tuning. One thing that can be said for sure is that EC, and SI approaches are proven good tools for gait optimization for legged robots, and further research should be done to improve their performance in the field of gait optimization.

5. Conclusions and Perspectives for Future Research

The most important conclusion that can be drawn from the literature mentioned above is that EC is indeed capable of achieving good performance in gait optimization and that this direction of research is very encouraging. However, it is obvious that we have not yet taken full advantage of EC, and some questions still need to be resolved for EC-based gait optimization.

To solve these problems, we suggest that future research should focus on the following issues.

5.1. Studying EC Approaches That Are Gait Optimization Niches. One of the most obvious problems in EC-based gait optimization is that of computational efficiency. Evolutionary methods generally require the maintenance of a population of candidate solutions and an iteration of a large number of generations. Thus it can be very time-consuming to evaluate every candidate gait especially when the experiment is carried out on a physical robot over several days while requiring constant manual supervision [32].

Though many researchers have optimized the gait of legged robots using EC, they mainly just transplanted the EC from other optimization applications with a few modifications. In order to solve gait optimization problems properly, it is necessary to study EC-based algorithms that are especially suitable for solving gait optimization problems, that is, study EC that can evolve the gait of robots with a high level of efficiency and quality.

Some researchers have noticed the necessity of studying the gait optimization niching EC paradigm, operators, and parameters and have done some work in this area. For example, using interpolation and extrapolating operators instead of a crossover has been found to reduce the size of the population [29], which is an essential element for robotic applications because the time required for optimization may be considerably reduced. The ideas of parameter tuning and the use of a global-local hybrid algorithm for the global optimizers to solve large-scale gait optimization problems have also been suggested in [19, 38]. Some fragmentary information has also been established on the characteristics of the design space of the gait optimization problem, for example, the penalty terms of the constraints could result in the minimum of the objective function being located within a narrow "channel", and the shape of the design space could made it difficult to locate the global minimum without the use of gradient information, and so forth [19]. This may be helpful for designing EC approaches.

The problem representation method and the genotype encoding method should also be studied as they can lead to different efficiency of EC. For instance, two different gait definition methods, a finite state machine based on the joint angles of the robot legs, and an Elman's recurrent neural network were studied [80], and the performance of the neural controller was reported superior (more stable, better displacement) for a simulated legged robot navigating on an irregular surface. TFS has also been reported to be a good gait representation approach and has been mentioned above [76].

Yet another method that could dramatically improve the gait optimization efficiency of EC has been proposed [81]. This method harnesses the general purpose computing on graphics processing units (GPGPU) to produce hardware-accelerated simulations without significant loss in fidelity, and it has achieved results that are orders of magnitude faster than software-only simulation (a 10–50-fold increase in speed has been reported).

We believe that EC-based gait optimization research should continue in this direction. The characteristics of the search space of gait optimization should be studied. This is determined by objective functions and constraints. Different applications of legged robot can have different objective functions and constraints, and different combinations of objective functions and constraints can have different search space characteristics. Based on the characteristic information of a search space, some benchmark test functions should be designed and classified to reflect to a certain extent the characteristics of the gait optimization objective functions. These benchmark test functions will facilitate the study of the corresponding EC paradigms, the genetic operators and control parameter sets of EC, providing researchers of legged robot gait optimization with a suite of high-performance optimization tools that can work well "out of the box" [19]. In particular, we suggest that both the EC hybrid with local search and the GAs that can learn linkage may play an important role in gait optimization. The former can integrate the global search ability of EC with the local optimization ability of local search operators, and the latter can evolve optimal code sequencing by learning the linkage among the genes and therefore enhancing the search ability of EC in tackling complicated, large scale problems of gait optimization.

5.2. Investigating the Objective Functions of EC. The objective function plays a critical role in EC. It serves as the environment for judging whether a solution represented by a chromosome in the simulated evolution is good or bad. It thus directs the search direction. The fitness functions in the research mentioned above were all different from one another. These were dependent on each researcher's insight into the nature of the problem, the different nature of each robot model or platform, and the performance requirements of each robot.

When constructing the objective functions for gait optimization, one has to compromise between the quality of solution and the velocity of evolution. A complex objective function including more performance indexes will, of course, lead to good solutions, but it can also greatly increase the size of the search space, and hence the computation cost of evaluation as well as the duration of the evolution. A lot of experience and techniques are needed to compose the objective functions (and the penalty weight set) properly. Therefore, it would be beneficial to construct a set of general guidelines for selecting the evaluation functions, especially for a typical application scenario or for a special robot platform [43].

5.3. Bridging the Reality Gap. Another important problem in EC-based gait optimization is that the gait evolved by simulation usually does not yield the same behavior once it is transferred to a real robot, since simulation always includes some simplifications when modeling the real world. This is called *the reality gap* problem in the field of evolutionary robotics. It is a branch of artificial intelligence concerned with the automatic generation of autonomous robots [74] and of course includes the gait generation of legged robots. Though evolving gait directly in real robots is an attractive goal with certain advantages, it is also very time-consuming requiring heavy user-intervention.

A possible scheme for tackling the problem of the reality gap in gait optimization nowadays is to integrate off-line and on-line evolution with respect to the reality gap.

Staged evolution may be a good method for this purpose [32, 42, 43, 82, 83]. This method evolves the gait by simulation first so that the gait may get a preliminary evaluation faster, and then the simulation results are transferred to the physical robot where the process of evolution is continued on the real robot. In this way, both the efficiency of EC and the quality of the solution may be significantly improved. A general multistage process, which minimizes the reality gap between real and simulated robots with respect to the behavior of actuators and their interaction with the environment, has been reported to be transferable to different kinds of legged robots [83].

Another method worth considering is *the back to reality algorithm* [74]. This method employs a coevolutionary construction that allows continuous robot behavior adaptation interleaved with simulator adaptation. It was able to solve the gait optimization problem in real robots using fewer evaluations than most of the currently existing approaches. It was used for the gait optimization of AIBO and is claimed to be the first work in which the simulated gait been successful and constantly transferred to reality [74].

In conclusion, a lot of work still remains to be done. Evolutionary gait optimization for legged robots is a promising field of research, and future encouraging and interesting results can be expected.

Acknowledgments

This work was supported by Talents Training Foundation of Beijing Municipal Government under Grant 20071D0501500196. The authors gratefully thank the anonymous reviewers for their valuable comments on the earlier draft of this paper and Mark Jonikas of Beijing University of Technology for extensive editing assistance.

References

[1] L. Chen, P. Yang, Z. Liu, H. Chen, and X. Guo, "Gait optimization of biped robot based on mix-encoding genetic algorithm," in *Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications (ICIEA '07)*, pp. 1623–1626, May 2007.

[2] S. Chernova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pp. 2562–2567, October 2004.

[3] M. Srinivasan and A. Ruina, "Computer optimization of a minimal biped model discovers walking and running," *Nature*, vol. 439, no. 7072, pp. 72–75, 2006.

[4] J. E. A. Bertram and A. Ruina, "Multiple walking speed-frequency relations are predicted by constrained optimization," *Journal of Theoretical Biology*, vol. 209, no. 4, pp. 445–453, 2001.

[5] R. M. Alexander, "Optimization and gaits in the locomotion of vertebrates," *Physiological Reviews*, vol. 69, no. 4, pp. 1199–1227, 1989.

[6] R. M. Alexander, "Optimum walking techniques for quadrupeds and bipeds," *Journal of Zoology*, vol. 192, pp. 97–117, 1980.

[7] R. M. Alexander, "Energetics and optimization of human walking and running: the 2000 Raymond Pearl Memorial Lecture," *American Journal of Human Biology*, vol. 14, no. 5, pp. 641–648, 2002.

[8] R. M. Alexander, "Design by numbers," *Nature*, vol. 412, no. 6847, p. 591, 2001.

[9] J. E. A. Bertram, "Constrained optimization in human walking: cost minimization and gait plasticity," *The Journal of Experimental Biology*, vol. 208, no. 6, pp. 979–991, 2005.

[10] A. K. Gutmann, B. Jacobi, M. T. Butcher, and J. E. A. Bertram, "Constrained optimization in human running," *The Journal of Experimental Biology*, vol. 209, no. 4, pp. 622–632, 2006.

[11] D. F. Hoyt and C. R. Taylor, "Gait and the energetics of locomotion in horses," *Nature*, vol. 292, no. 5820, pp. 239–240, 1981.

[12] F. Saibene, "The mechanisms for minimizing energy expenditure in human locomotion," *European Journal of Clinical Nutrition*, vol. 44, no. 1, pp. 65–71, 1990.

[13] J. Yang, B. Hong, S. Piao, and Q. Huang, "An efficient strategy of penalty kick and goal keep based on evolutionary walking gait for biped soccer robot," *Information Technology Journal*, vol. 6, no. 8, pp. 1120–1129, 2007.

[14] C. Zhou, P. K. Yue, J. Ni, and S.-B. Chan, "Dynamically stable gait planning for a humanoid robot to climb sloping surface," in *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, pp. 341–346, Singapore, December 2004.

[15] L. Hu and C. Zhou, "EDA-Based optimization and learning methods for biped gait generation," in *Robotic Welding, Intelligence and Automation*, T.-J. Tarn, et al., Ed., vol. 362 of *Lecture Notes in Control and Information Sciences*, pp. 541–549, Springer, Berlin, Germany, 2007.

[16] M. Shrivastava, A. Dutta, and A. Saxena, "Trajectory generation using GA for an 8 DOF biped robot with deformation at the sole of the foot," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 49, no. 1, pp. 67–84, 2007.

[17] K. Seo and S. Hyun, "Genetic programming based automatic gait generation for quadruped robots," in *Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference (GECCO '08)*, pp. 293–294, July 2008.

[18] M. R. Heinen and F. S. Osório, "Gait control generation for physically based simulated robots using genetic algorithms," in *Proceedings of the 2nd International Joint Conference on Advances in Artificial Intelligence—IBERAMIA-SBIA*, J. S. Sichman, et al., Ed., vol. 4140 of *Lecture Notes in Computer Science*, pp. 562–571, Ribeirão Preto, Brazil, October 2006.

[19] B.-I. Koh, J. A. Reinbolt, A. D. George, R. T. Haftka, and B. J. Fregly, "Limitations of parallel global optimization for large-scale human movement problems," *Medical Engineering and Physics*, vol. 31, no. 5, pp. 515–521, 2009.

[20] Z. Tang, C. Zhou, and Z. Sun, "Humanoid walking gait optimization using GA-based neural network," in *Proceedings of the 1st International Conference on Natural Computation (ICNC '05)*, L. Wang, K. Chen, and Y. S. Ong, Eds., vol. 3611 of *Lecture Notes in Computer Science*, pp. 252–261, August 2005.

[21] T. Arakawa and T. Fukuda, "Natural motion trajectory generation of biped locomotion robot using genetic algorithm

through energy optimization," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1495–1500, October 1996.

[22] C. Paul and J. C. Bongard, "The road less travelled: morphology in the optimization of biped robot locomotion," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '01)*, vol. 1, pp. 226–232, Maui, Hawaii, USA, October 2001.

[23] J. Pettersson, H. Sandholt, and M. Wahde, "A flexible evolutionary method for the generation and implementation of behaviors for humanoid robots," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*, pp. 279–286, November 2001.

[24] K. Wolff, J. Pettersson, A. Heralić, and M. Wahde, "Structural evolution of central pattern generators for bipedal walking in 3D simulation," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC '06)*, vol. 1, pp. 227–234, Taipei, Taiwan, October 2007.

[25] P. R. Vundavilli and D. K. Pratihar, "Soft computing-based gait planners for a dynamically balanced biped robot negotiating sloping surfaces," *Applied Soft Computing Journal*, vol. 9, no. 1, pp. 191–208, 2009.

[26] M. F. Silva, R. S. Barbosa, and J. A.T. Machado, "Development of a genetic algorithm for the optimization of hexapod robot parameters," in *Proceedings of the IASTED International Conference on Applied Simulation and Modelling (ASM '09)*, pp. 77–82, Palma de Mallorca, Spain, 2009.

[27] G. Dip, V. Prahlad, and P. D. Kien, "Genetic algorithm-based optimal bipedal walking gait synthesis considering tradeoff between stability margin and speed," *Robotica*, vol. 27, no. 3, pp. 355–365, 2009.

[28] M. R. Heinen and F. S. Osório, "Applying genetic algorithms to control gait of physically based simulated robots," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1823–1830, July 2006.

[29] T. Röfer, "Evolutionary gait-optimization using a fitness function based on proprioception," in *RoboCup Symposium*, pp. 310–322, Springer, Lisbon, Portugal.

[30] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI '04)*, pp. 611–616, San Francisco, Calif, USA, July 2004.

[31] E. Kim, M. Kim, and J.-W. Kim, "Optimal trajectory generation for walking up and down a staircase with a biped robot using genetic algorithm (GA)," in *FIRA RoboWorld Congress*, J.-H. Kim, et al., Ed., vol. 5744 of *Lecture Notes in Computer Science*, pp. 103–111, August 2009.

[32] K. Wolff and P. Nordin, "Evolutionary learning from first principles of biped walking on a simulated humanoid robot," http://fy.chalmers.se/~wolff/WN_gecco03.pdf.

[33] J. E. Murphy, H. Carr, and M. O'Neill, "Grammatical evolution for gait retargeting," in *Eurographics UK Symposium on Theory and Practice of Computer Graphics (TPCG '08)*, pp. 159–162, Manchester, UK, June 2008.

[34] J. E. Murphy, M. O'Neill, and H. Carr, "Exploring grammatical evolution for horse gait optimisation," in *Proceedings of the 12th European Conference on Genetic Programming (EuroGP '09)*, vol. 5481 of *Lecture Notes in Computer Science*, pp. 183–194, Springer, 2009.

[35] M. Hebbel, R. Kosse, and W. Nistico, "Modeling and learning walking gaits of biped robots," in *Proceedings of the Workshop on Humanoid Soccer Robots of the IEEE-RAS International Conference on Humanoid Robots*, pp. 40–48, Genoa, Italy, 2006.

[36] M. Hebbel, W. Nistico, and D. Fisseler, "Learning in a high dimensional space: fast omnidirectional quadrupedal locomotion," in *Proceedings of the 10th RoboCup International Symposium*, G. Lakemeyer, et al., Ed., vol. 4434 of *Lecture Notes in Computer Science*, pp. 314–321, Bremen, Germany, June 2007.

[37] K. Wolff, D. Sandberg, and M. Wahde, "Evolutionary optimization of a bipedal gait in a physical robot," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 440–445, June 2008.

[38] K. Wampler and Z. Popovi, "Optimal gait and form for animal locomotion," *ACM Transactions on Graphics*, vol. 28, no. 3, article 60, 8 pages, 2009.

[39] L. Hu, C. Zhou, and Z. Sun, "Biped gait optimization using spline function based probability model," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 830–835, Orlando, Fla, USA, May 2006.

[40] C. Zhou, L. Hu, C. A. A. Acosta, and P. K. Yue, "Humanoid soccer gait generation and optimization using probability distribution models," in *Proceedings of the Workshop on Humanoid Soccer Robots of the IEEE-RAS International Conference on Humanoid Robots*, pp. 49–55, Genoa, Italy, December, 2006.

[41] J. I. Alonso-Barba, J. A. Gámez, J. M. Puerta, and I. García-Varea, "Gait optimization in AIBO robots using an estimation of distribution algorithm," in *Proceedings of the 8th International Conference on Hybrid Intelligent Systems (HIS '08)*, pp. 150–155, September 2008.

[42] J. Eperješi, "Gait optimization of AIBO robot based on interactive evolutionary computation," in *Proceedings of the 6th International Symposium on Applied Machine Intelligence and Informatics (SAMI '08)*, pp. 236–240, January 2008.

[43] M. A. Lewis, A. H. Fagg, and G. A. Bekey, "Genetic algorithms for gait synthesis in a hexapod robot," in *Recent Trends in Mobile Robots*, V. Zheng, Ed., pp. 317–331, World Scientific, River Edge, NJ, USA, 1994.

[44] G. Capi, Y. Nasu, M. Yamano, and K. Mitobe, "Multicriteria optimal humanoid robot motion generation," in *Humanoid Robots*, A. C. de Pina Filho, Ed., pp. 157–170, Advanced Robotic Systems International and I-Tech, Vienna, Austria, 2007.

[45] S. Bi, H. Min, Q. Liu, and X. Zheng, "Multi-objective optimization for a humanoid robot climbing stairs based on genetic algorithms," in *Proceedings of the IEEE International Conference on Information and Automation (ICIA '09)*, pp. 66–71, Zhuhai, China, 2009.

[46] G. Capi, M. Yokota, and K. Mitobe, "A new humanoid robot gait generation based on multiobjective optimization," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '05)*, vol. 1, pp. 450–454, Monterey, Calif, USA, July 2005.

[47] D. Golubovic and H. Hu, "Parameter optimisation of an evolutionary algorithm for on-line gait generation of quadruped robots," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT '03)*, vol. 1, pp. 221–226, December 2003.

[48] Z. Peng, Q. Huang, X. Zhao, T. Xiao, and K. Li, "Online trajectory generation based on off-line trajectory for biped humanoid," in *Proceedings of IEEE International Conference on Robotics and Biomimetics (ROBIO '04)*, pp. 752–756, Shenyang, China, August 2004.

[49] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, "Autonomous evolution of gaits with the

Sony quadruped robot," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1297–1304, Morgan Kaufmann, San Mateo, Calif, USA, 1999.

[50] Q. Wang, C. Rong, G. Xie, and L. Wang, "Collaborative localization and gait optimization of SharPKUngfu team," in *Robotic Soccer*, P. Lima, Ed., pp. 549–574, Itech Education and Publishing, Vienna, Austria, 2007.

[51] K. Wolff, D. Sandberg, and M. Wahde, "Evolutionary optimization of a bipedal gait in a physical robot," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 440–445, June 2008.

[52] G. S. Hornby, S. Takamura, T. Yamamoto, and M. Fujita, "Autonomous evolution of dynamic gaits with two quadruped robots," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 402–410, 2005.

[53] W. Chen, "Odometry Calibration and Gait Optimization," Tech. Rep., School of Computer Science and Engineering, The University of New South Wales, 2005, http://www.cse.unsw.edu.au/~robocup/2005site/reports05/RobocupThesis2005_weiming.pdf.

[54] T. Mericli, H. L. Akin, C. Mericli, K. Kaplan, and B. Celik, "The Cerberus'05 Team Report," http://robot.cmpe.boun.edu.tr/~cerberus/wiki/uploads/Downloads/Cerberus2005-TR.pdf.

[55] C. Niehaus, T. Roefer, and T. Laue, "Gait optimization on a humanoid robot using particle swarm optimization," http://www.informatik.uni-bremen.de/kogrob/papers/Humanoids-Niehaus-etal-07.pdf.

[56] J. H. Park and M. Choi, "Generation of an optimal gait trajectory for biped robots using a genetic algorithm," *JSME International Journal, Series C*, vol. 47, no. 2, pp. 715–721, 2004.

[57] G. Capi, S. Kaneko, K. Mitobe, L. Barolli, and Y. Nasu, "Optimal trajectory generation for a prismatic joint biped robot using genetic algorithms," *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 119–128, 2002.

[58] W. I. Sellers, L. A. Dennis, W.-J. Wang, and R. H. Crompton, "Evaluating alternative gait strategies using evolutionary robotics," *Journal of Anatomy*, vol. 204, no. 5, pp. 343–351, 2004.

[59] S. Fan and M. Sun, "Gait parameters optimization and real-time trajectory planning for humanoid robots," in *Proceedings of the 3rd International Conference on Intelligent Computing (ICIC '07)*, D.-S. Huang, L. Heutte, and M. Loog, Eds., vol. 4682 of *Lecture Notes in Computer Science*, pp. 35–46, 2007.

[60] G. Capi, Y. Nasu, L. Barolli, K. Mitobe, M. Yamano, and K. Takeda, "A new gait optimization approach based on genetic algorithm for walking biped robots and a neural network implementation," *Information Processing Society of Japan (IPSJ) Journal*, vol. 43, no. 4, pp. 1039–1049, 2002.

[61] T. Arakawa and T. Fukuda, "Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP layers," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '97)*, pp. 211–216, April 1997.

[62] G. Capi, Y. Nasu, L. Barolli, K. Mitobe, and M. Yamano, "Real time generation of humanoid robot optimal gait for going upstairs using intelligent algorithms," *Industrial Robot*, vol. 28, no. 6, pp. 489–497, 2001.

[63] G. Capi, S. Kaneko, K. Mitobe, L. Barolli, and Y. Nasu, "Optimal trajectory generation for a prismatic joint biped robot using genetic algorithms," *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 119–128, 2002.

[64] Y. Hasegawa, T. Arakawa, and T. Fukuda, "Trajectory generation for biped locomotion robot," *Mechatronics*, vol. 10, no. 1-2, pp. 67–89, 2000.

[65] G. Capi and M. Yokota, "Optimal multi-criteria humanoid robot gait synthesis—an evolutionary approach," *International Journal of Innovative Computing, Information and Control*, vol. 2, no. 6, pp. 1249–1258, 2006.

[66] M.-Y. Cheng and C.-S. Lin, "Genetic algorithm for control design of biped locomotion," *Journal of Robotic Systems*, vol. 14, no. 5, pp. 365–373, 1997.

[67] X. Ke and Z. Gong, "Non-time reference gait planning and optimization for a stable bipedal walking," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '08)*, pp. 1400–1405, July 2008.

[68] D. L. Wight, E. G. Kubica, and D. W. L. Wang, "Introduction of the foot placement estimator: a dynamic measure of balance for bipedal robotics," *Journal of Computational and Nonlinear Dynamics*, vol. 3, no. 1, 2008.

[69] K. S. Jeon, O. Kwon, and J. H. Park, "Optimal trajectory generation for a biped robot walking a staircase based on genetic algorithms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pp. 2837–2842, October 2004.

[70] G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita, "Evolving robust gaits with AIBO," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, pp. 3040–3045, April 2000.

[71] K. Wolff and P. Nordin, "Evolution of efficient gait with an autonomous biped robot using visual feedback," http://fy.chalmers.se/~wolff/Papers/WN_gecco01.pdf.

[72] M. Eaton and T. J. Davitt, "Automatic evolution of bipedal locomotion in a simulated humanoid robot with many degree of freedom," in *Proceedings of the 11th International Symposium on Artificial Life and Robotics*, pp. 23–25, Beppu, Japan, 2006.

[73] S. Ha, Y. Han, and H. Hahn, "Adaptive gait pattern generation of biped robot based on human's gait pattern analysis," *International Journal of Mechanical Systems Science and Engineering*, vol. 1, no. 2, pp. 80–85, 2007.

[74] J. C. Zagal and J. Ruiz-Del-Solar, "Combining simulation and reality in evolutionary robotics," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 50, no. 1, pp. 19–39, 2007.

[75] N. Shafii, A. Khorsandian, A. Abdolmaleki, and B. Jozi, "An optimized gait generator based on Fourier series towards fast and robust biped locomotion involving arms swing," in *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL '09)*, pp. 2018–2023, Shenyang, China, August 2009.

[76] N. Shafii, S. Aslani, O. M. Nezami, and S. Shiry, "Evolution of biped walking using truncated Fourier series and particle swarm optimization," in *Proceedings of the 13th RoboCup International Symposium*, vol. 5949 of *Lecture Notes in Computer Science*, pp. 344–354, Graz, Austria, 2010.

[77] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm," *Chaos, Solitons & Fractals*, vol. 37, no. 3, pp. 698–705, 2008.

[78] C. Rong, Q. Wang, Y. Huang, G. Xie, and L. Wang, "Autonomous evolution of high-speed quadruped gaits using particle swarm optimization," in *Proceedings of the 12th Annual RoboCup International Symposium (RoboCup '08)*, L. Iocchi, et al., Ed., vol. 5399 of *Lecture Notes in Computer Science*, pp. 259–270, Suzhou, China, July 2009.

[79] E.-S. Kim, J.-H. Kim, and J.-W. Kim, "Generation of optimal trajectories for ascending and descending a stair of a humanoid based on uDEAS," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 660–665, Jeju, Korea, August 2009.

[80] M. R. Heinen and F. S. Osório, "Evolving morphologies and gaits of physically realistic simulated robots," in *Proceedings of 24th Annual ACM Symposium on Applied Computing (SAC '09)*, pp. 1161–1165, Honolulu, Hawaii, USA, March 2009.

[81] J. Rieffel, F. Saunders, S. Nadimpalli, et al., "Evolving soft robotic locomotion in PhysX," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pp. 2499–2504, 2009.

[82] L. Yang, C.-M. Chew, T. Zielinska, and A.-N. Poo, "A uniform biped gait generator with offline optimization and online adjustable parameters," *Robotica*, vol. 25, no. 5, pp. 549–565, 2007.

[83] T. Laue and M. Hebbel, "Automatic parameter optimization for a dynamic robot simulation," in *Proceedings of the 12th Annual RoboCup International Symposium (RoboCup '08)*, vol. 5399 of *Lecture Notes in Computer Science*, pp. 121–132, Suzhou, China, July 2009.