Preserving and Exploiting Genetic Diversity in Evolutionary Programming Algorithms

Gang Chen, Chor Ping Low, *Member, IEEE*, and Zhonghua Yang, *Senior Member, IEEE*

***Abstract*— Evolution programming (EP) is an important category of evolutionary algorithms. It relies primarily on mutation operators to search for solutions of function optimization problems (FOPs). Recently a series of new mutation operators have been proposed in order to improve the performance of EP. One prominent example is the fast EP (FEP) algorithm which employs a mutation operator based on the Cauchy distribution instead of the commonly used Gaussian distribution. In this paper, we seek to improve the performance of EP via exploring another important factor of EP, namely, the *selection strategy*. Three selection rules R1–R3 have been presented to encourage both fitness diversity and solution diversity. Meanwhile, two solution exchange rules R4 and R5 have been introduced to further exploit the preserved genetic diversity. Simple theoretical analysis suggests that through the proper use of R1–R5, EP is more likely to find high-fitness solutions quickly. Our claim has been examined on 25 benchmark functions. Empirical evidence shows that our solution selection and exchange rules can significantly enhance the performance of EP.**

***Index Terms*— Evolutionary optimization, evolutionary programming (EP), selection strategy.**

I. Introduction

INSPIRED by Darwinian principles of evolution and natural selection, a significant amount of computational models and methods have been studied under the umbrella of evolutionary computation (EC) for solving large, complex, and dynamic problems, which are generally very difficult to solve using conventional approaches [1]. In this paper, we will focus primarily on one important method in EC, commonly known as evolutionary programming (EP) [2], which was originally proposed as a paradigm for artificial intelligence (AI) [3].

A. EP Algorithm

Starting from evolving finite-state machines, EP has since been gradually enhanced to handle almost any data structures and has been successfully applied to numerous combinatorial optimization problems [4], [5]. Similar to other EC methods, EP is a population-based algorithm. In its primitive form, EP is fairly simple and can be summarized in terms of two major steps: 1) mutate the candidate solutions in the current population; and 2) select individual solutions for the next generation from both the mutated and the current solutions [6]. For convenience, solutions in the current population are called *parent solutions*, and solutions obtained through mutation are called *offspring solutions*. Schematically, offspring is generated by adding nonzero random numbers that follow certain probability distributions to its parent.

Previous research suggests that EP may experience *immature convergence* in problems with many local optima. As mutation is the essential operator of EP, a series of research has been devoted to designing new mutation operators so as to diversify the search process. For example, Yao *et. al.* proposed to construct the mutation operator based on the Cauchy distribution instead of the commonly used Gaussian distribution [4]. They have conducted some theoretical analysis to explain the advantages of using the Cauchy distribution. The resulting algorithm is termed the *Fast* EP (FEP). Its effectiveness has been demonstrated in optimizing high-dimensional functions. Later research further introduced mutation operators based on the Lévy distribution, which actually generalizes the Cauchy distribution [7].

B. Selection Strategies for EP

In addition to employing probability distributions with *infinite* second moment (e.g., Cauchy distribution), it is possible to explore another important factor of EP, namely the *selection mechanisms*, in order to further improve the optimization performance. In conventional EP, *tournament selection* has been utilized to build next-generation populations [8]. The first step is to define a *winning function* based on the fitness of each individual solution [9]. Depending on the problems under study, fitness is either positively correlated or negatively correlated with the value to be optimized. For any pair of solutions, the solution with higher fitness will win the tournament. In conventional EP, a fixed round of tournaments will be performed for every parent and offspring solution [6], [7]. Only those solutions that have managed to win a large number of tournaments will survive to the next generation.

Common selection strategies that are applicable in EP include proportionate selection, truncate selection, and ranking-based selection [10], [11]. In linear proportionate selection, an individual solution will be selected in proportion to its fitness. In truncate selection, $\alpha\%$ of the fittest solutions will be chosen and usually multiplied by a factor $100/\alpha$ to maintain a fixed population size. And in ranking-based selection, individuals in the population are ordered according to their fitness. Based on the ordering, each solution is assigned a rank, which further determines its probability of being chosen for the next generation.

Manuscript received August 7, 2007; revised October 6, 2008; accepted November 29, 2008. Current version published June 10, 2009.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: chengang@ntu.edu.sg).

Digital Object Identifier 10.1109/TEVC.2008.2011742

All these selection strategies only favor high-fitness individuals and have their strength and weakness. On one hand, they allow high-fitness solutions to quickly prevail and occupy a significant proportion of the population. This may be helpful for problems with few local optima. On the other hand, tournament and similar selection strategies are also prone to *immature convergence*, where all solutions in the population are located closely around some local optima and no improvements can be obtained further. As highlighted in [12], maintaining the right *selection pressure* is critical for solving many optimization problems. Consequently, in order to improve the optimization performance, *genetic diversity* should be encouraged and preserved at a proper level.

There are no generally accepted definitions for genetic diversity. In the literature, a variety of methods, such as fitness sharing [13], crowding [14], and local mating [15], have been proposed in an attempt to preserve genetic diversity. These methods more or less involve a neighborhood relation, which might be problem and domain specific. Moreover, the population may still converge immaturely to local optima when the network of neighborhood relations exhibit certain topologies [16]. Recently, Hutter and Legg suggested to preserve genetic diversity in terms of preserving the fitness diversity [12]. The full range of fitness values is divided equally into several consecutive sub-ranges. Two solutions are considered similar to each other if their fitness values belong to the same sub-range. Meanwhile, solutions in all sub-ranges, even of the sub-range near the bottom (i.e., low fitness values), deserve an equal opportunity of survival (i.e., chosen for the next generation). Experiments showed that their selection strategy is effective for some NP hard problems.

C. Proposed Selection and Solution Exchange Mechanisms

Aiming at preserving and exploiting genetic diversity, a series of solution selection and exchange mechanisms will be presented in this paper. Unlike Hutter and Legg's fitness uniform selection scheme (FUSS) [12], genetic diversity is not achieved by selecting solutions that span the full fitness range. We notice that, in reality, there lacks strong theoretical justifications of such an approach for general optimization problems. Especially for EP, which relies primarily on mutation operations, it may be highly unlikely to obtain high-fitness solutions by solely mutating solutions with very low fitness values. In view of this, a *selection mechanism* based on the three rules below has been proposed by us to promote fitness diversity. Explanations of these rules will be covered in later sections.

R1: Individual solutions are ranked according to the number of tournaments they have managed to win.

R2: The relative probability of selecting solutions of different ranks follows a power-law distribution.

R3: Each individual solution is placed at a position i in the population. Among all solutions located at the same position, exactly one solution will be selected for the next generation.

To actually improve the performance of EP, genetic diversity as preserved through rules R1–R3 should be effectively exploited during the evolution process. Specifically, the "*co-influence*" of the search process performed locally at every position in the population must be enhanced. Otherwise, we end up with more or less a group of $(1 + 1)$ EA [17], [18] conducted in parallel as a population. Motivated by this idea, two rules R4–R5 below have been suggested in this paper to take advantage of genetic diversity. They together form a *solution exchange mechanism.*

R4: During each generation, at every position i in the population, an individual solution will be selected at random and a copy of it will be placed at position i.

R5: Instead of using R4, with a certain probability, a *combined solution* will be created at position i for subsequent selection.

Rules R4 and R5 introduced an additional solution at every position i in the population. According to rules R1–R3, this solution has some probability of replacing those solutions originally located at i in succeeding generations. At the same time as preserving genetic diversity, using R4 and R5 high-fitness solutions are expected to prevail in the population. This might help to increase the opportunity of identifying solutions with even higher fitness.

D. Contents of the Paper

This paper constitutes a study of the solution selection and exchange mechanisms in EP algorithms. In Section II, we will briefly review the basic operations of EP, especially the mutation operator of FEP and improved FEP (IFEP). A detailed description of the mechanisms proposed by us together with their application in EP will be covered in Section III. Simple analysis has been performed in an attempt to explain the effectiveness of the proposed mechanisms for function optimization problems. Related discussions have been arranged in Section IV.

For the convenience of our discussion, the FEP algorithm that adopts the solution selection and exchange mechanisms of this paper will be termed *modified EP* (MEP) in the sequel. A new set of benchmark functions, which was provided by the CEC2005 special session [19], has been explored in our experimental studies to evaluate the effectiveness of MEP. For a majority of benchmark functions, we found that MEP has prominently improved the performance of FEP to a level that is competitive to some of the recently introduced evolution algorithms, including neighborhood search differential evolution (NSDE) [20] and opposition-based differential evolution (ODE) [21]. The experiment results will be presented in Section V. Finally, Section VI concludes this paper.

II. Basic Operations of EP

This paper focuses on function optimization problems (FOPs). In its simplest form, an FOP includes two main elements X and f. X is any bounded subset of R^n and n is the dimension of the solution space. Fitness function $f : X \to R$ establishes a mapping from solutions $x \in X$ to real values. The problem is to find a solution x_{opt} such that $f\left(x_{opt}\right)$ reaches its optimum over X. No restrictions apply to function f except that it must be *bounded*.

EP employs a population-based search strategy in order to find solutions close enough to x_{opt}. A population Γ is comprised of μ distinct positions, where μ is the population size. Each individual in the population is defined as a two-element tuple (x_i, η_i), where $x_i \in X$, $\forall i \in \{1, \ldots, \mu\}$, stands for a candidate solution of the FOP, and i indicates the position of the individual in the population. Similar to x_i, η_i is an n-dimensional real-valued vector. Each element of η_i, $\eta_i[k]$, refers to the standard deviation of a probability distribution which is used by the mutation operator to alter the corresponding element of x_i, $x_i[k]$. The introduction of η_i is inspired by the research reported by Fogel [22] and Bäck and Schwefel [23]. They have shown that self-adaptive mutation in EP usually performs better than EP without self-adaptation. The equation below shows how the element of η_i will be updated

$$\eta_i'[k] = \eta_i[k] e^{N(0,1)\tau' + \tau N_k(0,1)}, \ k \in \{1, \ldots, n\} \tag{1}$$

where $N(0, 1)$ and $N_k(0, 1)$ represent two random real variables generated from independent Gaussian distributions with mean 0 and deviation 1. Parameters τ and τ' are defined as

$$\tau = \frac{1}{\sqrt{2\sqrt{\mu}}} \quad \text{and} \quad \tau' = \frac{1}{\sqrt{2\mu}}.$$

According to the description of FEP [6], especially IFEP, both the Gaussian and Cauchy distributions will be utilized to *mutate* individual solutions in the population. To be more specific, with respect to a solution (x_i, η_i), only one element of x_i will be selected at random for mutation. Suppose the chosen element is $x_i[k]$ after mutation. Its value becomes

$$x_i'[k] = x_i[k] + \eta_i[k] D_k^i(0, 1) \tag{2}$$

where $D_k^i(0, 1)$ stands for a sample from either a Gaussian or Cauchy distribution. Based on this mutation operation, solution (x_i', η_i') is therefore termed the offspring and solution (x_i, η_i) is termed its parent. The only difference between an offspring and its parent is the kth element (k is randomly selected). In this paper, an offspring is placed at the same position in the population as its parent.

The basic steps of the FEP algorithm have been summarized in Fig. 1. During each generation, FEP simply chooses μ solutions with highest $win(\cdot)$ to form a new population [6], where $win(x_i)$ indicates the total number of tournaments won by any solution x_i. This may lead to immature loss of genetic diversity. To prevent this problem, a series of solution selection and exchange rules will be proposed in the next section to preserve and exploit genetic diversity.

To conclude this section, it is noted that ranking a solution based on multiple rounds of tournaments (Ref. S6 in Fig. 1) is a common practice for EP algorithms [6], [7], [23]. This is different from the tournament-based selection usually performed by genetic algorithms [24], where only one round of tournaments may be run over a pair of candidate solutions and the winner is chosen and passes through [10].

S1: Generate an initial population of μ solutions (x_i, η_i), $i \in \{1, \ldots, \mu\}$, and evaluate their fitness $f(x_i)$.

S2: For every parent solution (x_i, η_i) in the population, create an offspring (x_i', η_i') based on (2), where $D_k^i(0, 1)$ is sampled from a standard Gaussian distribution.

S3: For every parent solution (x_i, η_i) in the population, create an offspring (x_i'', η_i'') based on (2), where $D_k^i(0, 1)$ is sampled from a standard Cauchy distribution.

S4: Update every η_i, η_i', and η_i'' according to (1).

S5: Evaluate the fitness of offspring solutions (x_i', η_i') and (x_i'', η_i'').

S6: Perform pairwise tournaments over the *union* of parents (i.e., (x_i, η_i)) and offspring (i.e., (x_i', η_i') and (x_i'', η_i'')). For each solution in the union, q opponents are chosen uniformly at random from the union. For every tournament, if the fitness of the solution is no smaller than the fitness of the opponent, it is awarded a *win*. The total number of wins awarded to a solution (x_i, η_i) is denoted by $win(x_i, \eta_i)$.

S7: Choose μ individual solutions from the union of parents and offspring at step S6. Chosen solutions will form the population of the next generation.

S8: Stop if the number of generations exceeds a predetermined number.

Fig. 1. Basic steps of the FEP algorithm.

III. Solution Selection and Exchange Mechanisms

In this paper, the performance of EP (or FEP) is to be improved via two techniques: 1) maintaining genetic diversity in order to prevent *immature convergence*; and 2) enhancing the interactions among candidate solutions in order to exploit genetic diversity. The following two sections will address the two techniques, respectively.

A. Selection Mechanism to Preserve Genetic Diversity

Genetic diversity is interpreted in this paper in terms of both *fitness diversity* and *solution diversity*. Fitness diversity is a general term that differentiates solutions according to their fitness. Alternatively, solution diversity characterizes the structure resemblance of solutions in the population. Both types of genetic diversity will be preserved in this section.

1) Preserving Fitness Diversity: In order to achieve fitness diversity, it is necessary to *deliberately* choose some solutions with relatively low fitness for the next generation. In line with the basic selection strategy of EP and FEP, solutions will be actually ranked in this paper through a series of pairwise tournaments (step S6 in Fig. 1).

In FEP, q rounds of tournaments will be arranged for each solution. Therefore, a total of $q + 1$ different rankings can be assigned to a solution. Specifically, according to the number of winning competitions, the rank of a solution (x_i, η_i) is defined to be $win(x_i, \eta_i)$, $0 \le win(x_i, \eta_i) \le q$. With respect to any given rank γ, the number of solutions assuming that rank is expected to be identical. C_γ is utilized to denote the group of solutions with rank γ. Based on the discussions in [12], q is usually set to $\sqrt{\mu}$. Using these settings, we are able to present our first two selection rules as follows.

R1: Each solution (x_i, η_i) available at S7 in Fig. 1 is associated with a *rank* that equals $win(x_i, \eta_i)$, $0 \le win(x_i, \eta_i) \le q$.

R2: If $(x_i, \eta_i) \in C_\gamma$, the probability of selecting (x_i, η_i) will be proportional to $(\gamma + 1)^\alpha$, where α, $\alpha \ge 0$, is the *exponent parameter* of a power-law distribution that governs the solution selection process.

A power-law distribution is utilized by R2 to control the tendency of selecting solutions that belong to different groups C_γ. This selection tendency can be adjusted by changing the *exponent parameter* α. At one extreme when $\alpha = 0$, all solutions enjoy equal selection probability. As α is gradually enlarged, solutions in high-ranking groups, such as C_q, will be selected more and more frequently. As a general guideline, we found that it would be a good strategy to slightly skew the selection preferences towards high-fitness solutions without significantly destroying fitness diversity (e.g., $\alpha = 1$).

In an attempt to encourage fitness diversity, the power-law distribution ensures that high-ranking solutions will never dominate the selection process. In particular, as k is at most $\sqrt{\mu}$, when α takes a small value less than or equal to 1, during μ rounds of selections, the number of solutions selected from group C_0 is expected to be

$$\mu \times \frac{Ps(C_0)}{\sum\limits_{C_\gamma, \gamma \in \{0,\ldots,\sqrt{\mu}\}} Ps(C_\gamma)} = \mu \times \frac{1}{\sum\limits_{\gamma=0}^{\sqrt{\mu}} (\gamma + 1)^\alpha}. \quad (3)$$

Equation (3) is derived with the assumption that the size of any solution group C_γ will exceed the number of solutions selected from that group. $Ps(C_\gamma)$ in (3) denotes the probability of selecting a solution from group C_γ. Equation (3) is greater than 1 when μ is big enough, such as $\mu > 20$. It suggests that the resulting population after the selection process is expected to include solutions from all $q+1$ groups $C_0, \ldots, C_q$. Fitness diversity is therefore encouraged. As far as the authors know, this paper is among the first attempts to integrate power-law distribution and tournament selection in the direction of preserving fitness diversity.

2) Preserving Solution Diversity: In addition to preserving fitness diversity, we also seek to maintain solution diversity. Essentially, solution diversity reveals a *closeness relationship* among the solutions in the population. For example, solutions (x_i, η_i) and (x_j, η_j) may be *close* to each other if every element of x_i, $x_i[k]$, only differs from its counterpart in x_j, $x_j[k]$, by a small value ϵ. Instead of fixing the upper bound of ϵ, which may depend on the problems being solved, this paper will follow a different approach.

According to the mutation operation described in Section II, an offspring (x'_i, η'_i) is identical to its parent (x_i, η_i) except for one element $x'_i[k]$. (x'_i, η'_i) is therefore considered *close* to its parent (x_i, η_i). Knowing that solutions (x'_i, η'_i) and (x_i, η_i) actually occupy the same position in the population, in order to preserve solution diversity, rule R3 below will be further utilized during the selection process.

R3: Among all solutions (x_i, η_i) at the same position i in the population, exactly one parent solution will be chosen for the next generation.

Based on R3, the population for the succeeding generation will actually be constructed via selecting one solution from each of the μ positions in the population. The actual selection decision should also obey R1 and R2. Intuitively, solutions located at the same position usually resemble a small region inside the overall solution space. After imposing rule R3, only one *representative* solution will be chosen with respect to each of these regions, thereby encouraging EP to explore new regions of the solution space. Meanwhile, as only a few solutions are available at each position in the population, the probability of selecting solutions with relatively low fitness will be increased, resulting in increased fitness diversity.

B. Solution Exchange Mechanism to Exploit Genetic Diversity

Due to R3, the local search at one position is by and large immune to the changes happening at other positions in the population. Although tournaments allow a solution (x_i, η_i) to compete freely with any existing solutions, from generation to generation, (x_i, η_i) will only be replaced by solutions located at the same position i. For this reason, despite of preserved genetic diversity at the population level, the overall performance of EP may not exhibit noticeable improvement at all. In view of this, it is necessary to enhance the interactions among solutions at all positions in the population. One straightforward way of achieving this is through exchange rule R4 below.

R4: At every position i in the population, select uniformly at random a parent solution (x_j, η_j) from the population and place a copy of it (x'''_i, η'''_i) at position i.

R4 introduces an additional solution (x'''_i, η'''_i) at every position i in the population. In association with R3, there are hence a total of four candidates to be selected at position i: 1) the parent solution (x_i, η_i), 2) an offspring (x'_i, η'_i) obtained at step S2 in Fig. 1, 3) an offspring (x''_i, η''_i) obtained at step S3 in Fig. 1, and 4) a randomly chosen copy (x'''_i, η'''_i).

As copied solutions may eventually replace those solutions originally located at position i in the population (at selection step S7 in Fig. 1), solutions at one location are able to affect the solutions being selected at other positions. This may lead to a more diversified search as well as rapid spread and exploration of high-fitness solutions. Besides simply cloning solutions, opportunities to explore new regions of the solution space are available through exchange rule R5 below.

R5: At every position i in the population, with a certain probability of Pc, a combined solution (x^*_i, η^*_i) will be created according to (4).

$$x^*_i[k] = x_{i_1}[k] + 0.5 \times \left(x_{i_2}[k] - x_{i_3}[k]\right), 1 \le k \le n \quad (4)$$

where i_1, i_2, and i_3 are three randomly selected and mutually different positions in the population Γ. Equation (4) is derived from the differential evolution (DE) algorithm, which has been proven to be very effective in both benchmark functions [20] and real-world applications [25]. The value 0.5 in (4) is taken from a recommended range of values based on extensive experimental studies of DE in the literature [26], [27]. Notice that, due to a probability of Pc, R5 may only be used occasionally and MEP still relies primarily on mutation operations, similar to FEP.

C. Summary

The solution selection and exchange rules R1–R5 constitute a significant modification of the tournament-based selection mechanism as adopted by conventional EP and FEP. During each generation, the process of generating and selecting parent solutions for the next generation is summarized in Fig. 2 as a flowchart.

Identical to FEP, the parent solution (x_i, η_i) at every position i of the population Γ will be exploited by our flowchart to generate two mutated offspring (x_i', η_i') and (x_i'', η_i'') according to steps S2 and S3 in Fig. 1. Meanwhile, depending on a probability of Pc, either one combined solution (x_i^*, η_i^*) (ref. R5) or one copy of a parent solution at a randomly chosen position (x_i''', η_i''') (ref. R4) will be created and placed at position i. After that, pairwise tournaments will be performed to determine the rank of those solutions currently available at position i (ref. R1). The parent solution for the next generation is finally selected subject to the power-law distribution (ref. R2), which will be applied repetitively at all positions of the population (ref. R1). Based on Fig. 2, two important observations are now in place.

O1: If $Pc = 0$, then the union of solutions available for building up the next-generation population is the same as FEP.

O2: If $Pc = 0$, two solutions (x_i', η_i') and (x_i'', η_i'') will be evaluated at every position i in the population. Hence the total number of function evaluations performed during each generation equals to $2 \times \mu$.

Observation O1 indicates that when $Pc = 0$, MEP actually demands exactly the same number of function evaluations as FEP. However, when Pc is greater than 0, the computational cost will be slightly increased. In particular, for every generation, the number of times of using R5 is expected to be $Pc \times \mu$. Following observation O2, the percentage of extra function evaluations [for solutions (x_i^*, η_i^*)] incurred due to R5 is

$$\frac{Pc \times \mu}{2 \times \mu} = \frac{Pc}{2}. \tag{5}$$

Equation (5) suggests that MEP will not considerably increase the computational cost with relatively small Pc. For example, when $Pc = 0.5$, the percentage of extra function evaluations is 25%, which may be judged as a small increase in many situations. Nevertheless, depending on functions, the effectiveness of R5 may not be able to justify the extra cost involved all the time.

Experiment results in Section V suggest that Pc should assume a larger value (i.e., equal to or above 0.5) especially when handling hybrid composition functions which are

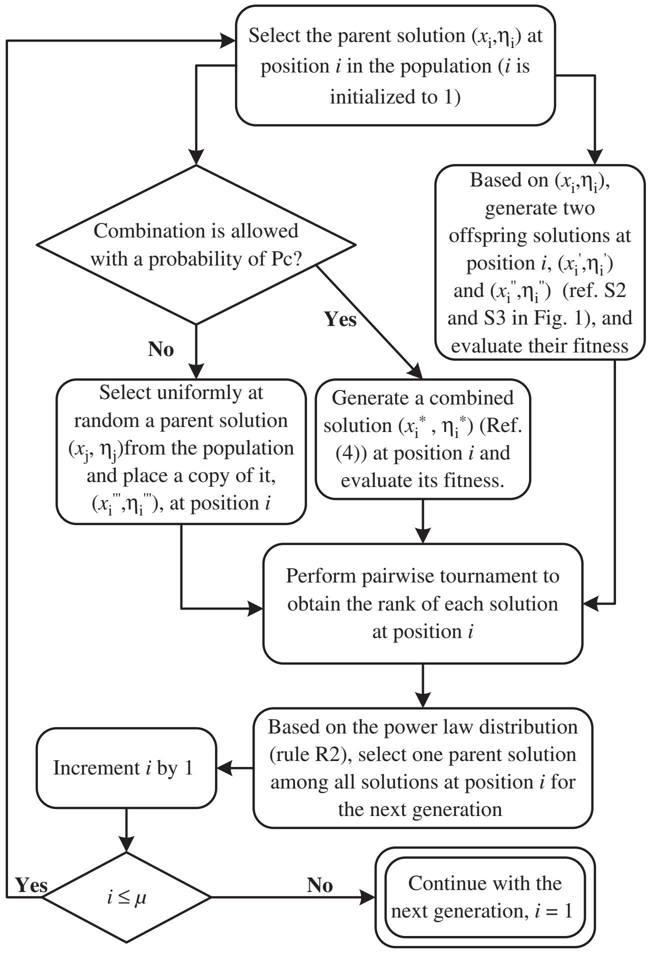
Select the parent solution (x_i, η_i) at position i in the population (i is initialized to 1)
Combination is allowed with a probability of Pc?
No
Yes
Based on (x_i, η_i), generate two offspring solutions at position i, (x_i', η_i') and (x_i'', η_i'') (ref. S2 and S3 in Fig. 1), and evaluate their fitness
Select uniformly at random a parent solution $(x_j, \eta_j)$from the population and place a copy of it, (x_i''', η_i'''), at position i
Generate a combined solution (x_i^*, η_i^*) (Ref. (4)) at position i and evaluate its fitness.
Perform pairwise tournament to obtain the rank of each solution at position i
Based on the power law distribution (rule R2), select one parent solution among all solutions at position i for the next generation
Increment i by 1
$i \le \mu$
Yes
No
Continue with the next generation, $i = 1$

Fig. 2. Solution selection and exchange process driven by rules R1–R5.

highly complex by nature, whereas for ordinary multimodal problems, Pc is expected to take a smaller value (i.e., equal to or below 0.5) for the purpose of reducing computational costs. In order to properly balance the cost and performance, a systematic approach should be developed to dynamically determine the appropriate value of Pc throughout the search process of MEP. As this paper concentrates on preserving genetic diversity, potential methods for adjusting Pc will not be elaborated, which will be treated as our future work.

It is worthwhile to note that by dropping rules R3–R5 and by increasing the value of the exponent parameter α, one is able to reconstruct the simple selection strategy of FEP, namely, only high fitness solutions will be chosen for subsequent generations. In fact, a family of selection strategies can be obtained through selectively combining the five rules R1–R5 and adjusting the parameters α and Pc. With this understanding, it is possible to view MEP as a consistent generalization of FEP.

IV. Simple Analysis of Solution Selection and Exchange Mechanisms

In this section, a simple analysis will be presented as a justification of the solution selection and exchange mechanisms proposed in Section III. In particular, the effectiveness of the solution exchange rule R4 will be demonstrated from two perspectives: 1) R4 helps to improve the mutation-based search process performed locally at every position of the population, and 2) R4 facilitates the rapid spread and exploration of high-fitness solutions, which may in turn increase the opportunity of identifying solutions with even higher fitness.

The analysis of the local search process is built on the absorbing Markov chain model of evolutionary algorithms [18]. Let us focus on a solution (x_i, η_i) at any location i of the population. Due to R3, the change of x_i across generations can be actually modeled as a discrete-time absorbing Markov chain. For the sake of our analysis, the state space X of this Markov chain is considered to be *discrete*, which is a practical assumption since computer systems essentially process discrete values. Let $g = 0, 1, \ldots$ indicate the gth generation of EP. x_i^g further denotes the solution at position i of the population during the gth generation. The change of x_i between any two consecutive generations can be captured through a *canonical transition matrix* [28]

$$P = \begin{pmatrix} I & 0 \\ R & T \end{pmatrix}. \tag{6}$$

P is an $\nu \times \nu$ matrix where ν is the cardinality of the state space X. Each state in X is assigned a unique row of matrix P. For convenience, the state associated with the mth row of matrix P will be denoted as $P[m]$. The element at the mth row and the nth column of matrix P, $P[m, n]$, stands for the probability for state $P[m]$ to *transfer* to state $P[n]$ after one step of evolution. In another word, $P[m, n]$ measures the probability for any solution $x_i^g = P[m]$ at position i to become $P[n]$ during the $(g + 1)$th generation.

To obtain a transition matrix as in (6), the states in X need to be properly organized such that all absorbing states will be assigned a row number between 1 and $\nu - dim(T)$, where $dim(T)$ gives the dimension of the square sub-matrix T. Depending upon the problems of interest, absorbing states in Markov chain models may bear diverse meanings. In this section, we are mainly interested in examining the *expected* number of steps d_i for x_i^{g+d} to first obtain a fitness $f(x_i^{g+d})$ higher than the fitness of x_i^g. Absorbing states in this analysis therefore refer to all states in X with fitness higher than $f(x_i^g)$. Because we are generally not interested in the details of absorbing states as well as the transitions between absorbing states, it is convenient for us to replace all absorbing states in the Markov chain model with a single *aggregated* absorbing state $P[1]$. Matrix P consequently becomes

$$P = \begin{pmatrix} 1 & 0 \\ R & T \end{pmatrix}. \tag{7}$$

Sub-matrix R is now a column vector with the same dimension as the square matrix T. Each element of R, $R[m]$, determines the probability for state $P[m + 1]$ to transfer to the absorbing state $P[1]$ during one step of evolution. Let $d = \{d_1, \ldots, d_{dim(T)}\}$ be the vector that lists the expected number of generations d_m ($m \in \{1, \ldots, dim(T)\}$) for the Markov chain starting from state $P[m + 1]$ to first *hit* the absorbing state. According to [18], [28], d can be actually determined through Theorem 1 as follows.

Theorem 1: If the absorbing Markov chain under study can be characterized through the transition matrix P in (7), then

$$d = (I - T)^{-1} \cdot \mathbf{1} \tag{8}$$

where $\mathbf{1}$ denotes the all-one vector with dimension $dim(T)$.

Let d correspond to the Markov chain obtained without using rule R4 and d' correspond to the Markov chain obtained after using R4. To support our comparison between d and d', a vector norm as defined below will be utilized

$$\|d\|_1 = \frac{1}{dim(T)} \cdot \sum_{i=1}^{dim(T)} d_i.$$

$\|d\|_1$ represents the *average-case* analysis. If $\|d\|_1$ is greater than $\|d'\|_1$, we know that on average the local search carried out with the help of R4 is expected to be faster than the local search without using R4. This line of thinking gives rise to Theorem 2 as follows.

Theorem 2: Assuming that solutions copied from other positions in the population due to rule R4 will be selected only if they have higher fitness values, then

$$\|d\|_1 \geq \|d'\|_1.$$

Proof: Let P' and P denote, respectively, the transition matrices of the Markov chain models obtained with and without using rule R4. Also let the sub-matrix T belong to P and the sub-matrix T' belong to P'. Two important properties apply to both T and T'.

P1: Every element of T and T' is non-negative.

P2: Every element of T is not less than the corresponding element of T'.

Property P2 can be verified based on the following two understandings.

U1: For all m, $m \in \{1, \ldots, dim(T)\}$, $R[m] \leq R'[m]$. This is because for solutions x_i^q at any position i, in addition to mutations, rule R4 will introduce a copied solution x_j^q from another position j in the population. With certain probability which is greater than 0, $f(x_j^q)$ will be higher than $f(x_i^q)$. Therefore, it is more likely for nonabsorbing states $P'[m+1]$ of the Markov chain with R4 to hit the absorbing state $P'[1]$.

U2: For all r between 1 and $dim(T)$, $T[m, r]/T'[m, r]$ is only a function of m, provided that $T[m, r] \neq 0$ and $T'[m, r] \neq 0$. Notice that $P[m + 1]$ and $P'[m + 1]$ actually stand for the same state (i.e., the same solution). The only difference between P and P' at any state $P[m + 1]$ is a cloned solution. According to our assumption, this cloned solution will only be selected if it has the highest fitness. In that case, state $P'[m + 1]$ will be transferred to the hitting state, therefore not affecting the state transitions as represented by T'. Since T and T' depend only on mutation operations, which are identical, the ratio between $T[m, r]$ and $T'[m, r]$ will therefore remain constant for fixed m.

With properties P1 and P2, we are able to compare $\|d\|_1$ and $\|d'\|_1$ below

$$\begin{aligned} &\|d\|_1 - \|d'\|_1 \\ &= \mathbf{1}^t (I - T)^{-1}\mathbf{1} - \mathbf{1}^t (I - T')^{-1}\mathbf{1} \\ &= \mathbf{1}^t \left(\sum_{k=0}^{\infty} \left((T)^k - (T')^k \right) \right) \mathbf{1} \\ &\geq 0. \end{aligned}$$

The last step of the above equation is due to the fact that all elements of the matrix $\left((T)^k - (T')^k\right)$ in the summation are non-negative. Therefore, when the matrix is left-multiplied by $\mathbf{1}^t$ and then right-multiplied by $\mathbf{1}$, the resulting real value must be non-negative as well. This leads to $\|d\|_1 \geq \|d'\|_1$. ■

Theorem 2 suggests that R4 holds the potential to expedite the local search process performed at every position in the population. Even in the worst case, R4 will not reduce the chances of finding high-fitness solutions. Moreover, R4 may also help to improve the performance of EP at the global scale. In particular, R4 facilitates the rapid spread of high-fitness solutions across the population. Intuitively, for many optimization problems, mutations performed over high-fitness solutions are more likely to produce new solutions with even higher fitness. Hence, the population as a whole is expected to converge to global optimum more quickly.

Notice that as genetic diversity is encouraged through selection rules R1–R3, even the optimum solution x_{opt} is incapable of occupying all positions of the population. Nevertheless, in order to demonstrate the fast spreading speed of high-fitness solutions, an extreme case will be studied here. We assume that, at every position in the population, candidate solutions with the highest fitness will be selected with probability 1. Let $\lambda(g)$ denote the fraction of positions in the population, which have been occupied by x_{opt} at the gth generation. Based on these settings, the change of $\lambda(g)$ can be modeled through the difference equation below (for simplicity, $\lambda(g)$ is assumed to be continuous)

$$\lambda(g+1) = \lambda(g) + \lambda(g)\left(1 - \lambda(g)\right). \quad (9)$$

The first term at the right-hand side of (9) represents the fact that positions occupied by x_{opt} will remain occupied by x_{opt} because of the extreme selection strategy we use. The second term measures the fraction of positions newly occupied by x_{opt} during the gth generation as a result of R4. Solving (9) leads to

$$\lambda(g) = 1 - e^{2^g \beta_1} \quad (10)$$

where β_1 is a constant that depends on the value of $\lambda(0)$—the initial fraction of positions taken up by x_{opt}. Equation (10) demonstrates that the fast spreading speed of x_{opt}. As a simple example, suppose that $\mu = 1000$, and initially x_{opt} stays only at one position in the population. After no more than 13 generations, x_{opt} is able to occupy 99% of the population.

V. Function Optimization Experiments

In order to evaluate the effectiveness of the solution selection and exchange mechanisms proposed in this paper, an experimental study has been performed by utilizing a specific set of benchmark functions, which was provided by the CEC2005 special session [19]. Specifically, the performance of MEP on a total of 25 functions with varied complexity has been evaluated in the experiments. The tested functions can be divided into three categories, namely, unimodal functions f_1–f_5, multimodal functions f_6–f_{14}, and hybrid composition functions f_{15}–f_{25}, which possess the highest complexity.

The dimension of the solution space of each benchmark function is fixed at 30. A detailed description of each function can be found in [19]. In fact, many of the functions are the shifted, rotated, expanded, or composed variants of classical benchmark functions that are widely explored in the literature [6]. Some of these modifications have rendered simple search tricks inapplicable [20] and therefore are highly suitable for examining the real advantage of using rules R1–R5.

Besides MEP and FEP, comparison experiments have also been performed by using two recently introduced evolutionary algorithms, namely NSDE and ODE [20], [21]. Meanwhile, the popular real-coded genetic algorithm (rGA) with simulated binary crossover (SBX) [29] and polynomial mutation [30] has been explored in our comparison study as well. Relevant comparison results will be highlighted at the end of this section. Our implementations of FEP, NSDE, ODE, and rGA follow essentially the descriptions in the respective references. Throughout the experiments, MEP will stick to the settings below:

1) population size (i.e., the number of distinct positions in the population): $\mu = 100$;
2) the number of rounds of tournaments: $q = \sqrt{\mu} = 10$;
3) the initial standard deviation: $\eta = 0.5$;
4) the exponent parameter of the power-law distribution: $\alpha = 1.0$ (α is set to 1.5 for function f_4).

Most of the above settings are identical to those adopted by FEP in [6]. The only difference is that in [6], η is set to 3.0 initially while in this paper, $\eta = 0.5$ at the beginning. Notice that according to Fig. 1, the value of η will be modified every time a parent solution is being mutated. Therefore, the initial value of η may not have a persistent influence on the long-term performance of FEP. Moreover, according to our observations, by setting η to 0.5, both FEP and MEP are able to achieve comparable or even better performance than when $\eta = 3.0$.

NSDE and ODE share two common parameters, namely a mutation parameter F_m that controls the mutation operation, which is conceptually equivalent to 0.5 in (4), and a crossover rate F_c. In this paper, $F_m = 0.5$ and $F_c = 0.8$. The values of both parameters are taken from the recommended value ranges. Other parameter settings of the two algorithms are determined according to the list of settings above and [20] and [21]. For rGA, the crossover probability is set to 0.7 and the mutation rate is at 0.01.

A. Performance of MEP and FEP on Unimodal and Multimodal Functions

Table I summarizes the performance of MEP and FEP with respect to unimodal and multimodal functions f_1–f_{14}. The

TABLE I

EXPERIMENTAL RESULTS OF MEP AND FEP FOR UNIMODAL AND MULTIMODAL FUNCTIONS, AVERAGED OVER 25 INDEPENDENT RUNS. FOR EACH FUNCTION, THE BEST MEAN PERFORMANCE ACHIEVED BY MEP IS MARKED USING BOLD FACE. COLUMN t-TEST (1) LISTS THE T-TEST RESULTS BETWEEN THE BEST PERFORMED MEP AND FEP. COLUMN t-TEST (2) LISTS THE T-TEST RESULTS BETWEEN MEP WITH $Pc = 0.08$ AND FEP

Test function	Number of generations	**FEP**	**MEP**			t-test (1)	t-test (2)
			$Pc = 0.08$	$Pc = 0.50$	$Pc = 0.90$		
f_1	1500	4.46e–4	9.13e–6	3.55e–6	**1.42e–6**	52.7	51.8
f_2	1500	549.5	439.2	10.409	**1.219**	84.3	7.97
f_3	10000	1.435e+6	6.12e+5	**5.041e+5**	2.93e+6	14.59	12.92
f_4	3000	3.978e+4	1.379e+4	130.4	**43.195**	159.3	77.96
f_5	20000	2.599e+4	2.52e+3	690.9	**568.7**	126.3	116.7
f_6	1500	58.66	**42.09**	70.09	60.99	7.4	7.4
f_7	3000	0.289	0.288	**0.288**	**0.288**	23.2	21.2
f_8	1500	20.53	20.53	**20.52**	20.67	0.226	–0.088
f_9	1500	0.595	**2.886e–3**	7.11e–3	0.182	18.6	18.6
f_{10}	1500	317.1	129.8	100.79	**88.22**	76.6	64.8
f_{11}	1500	34.50	**31.32**	32.07	33.23	9.41	9.41
f_{12}	5000	7.89e+3	**7.97e+3**	1.097e+4	9.87e+3	–0.430	–0.430
f_{13}	1500	1.147	**1.096**	1.167	1.712	13.7	13.7
f_{14}	1500	13.02	13.28	12.97	**12.44**	26.7	–5.88

experimental results of MEP under three different settings of Pc (i.e., 0.08, 0.5, and 0.9) have been listed in this table. Also indicated in the table is the number of generations (NoG) performed by MEP and FEP to optimize each of the 14 functions.

It is clear from Table I that at least a particular setting of Pc, namely $Pc = 0.08$, outperforms (or remains competitive with) FEP over the 14 benchmark functions. This observation is clearly evidenced by the last column of Table I, where the t-test results between MEP ($Pc = 0.08$) and FEP are provided. Specifically, MEP ($Pc = 0.08$) strictly outperforms FEP over 11 functions f_1–f_{13}. For functions f_8 and f_{12}, the difference between MEP ($Pc = 0.08$) and FEP is insignificant. FEP is evidently better than MEP ($Pc = 0.08$) only on one function f_{14}.

In accordance with the discussions in Section III-C, the extra function evaluations incurred by MEP with $Pc = 0.08$ are 4%, which is negligible in general. This result suggests that the four rules R1–R4 proposed in Section III are effective in improving the performance of EP.

1) Optimizing Unimodal Functions: Functions $f_1 - f_5$ are unimodal functions. Figs. 3–10 depict, respectively, the mean best solutions found by MEP and FEP at every generation after 25 independent trials for each of these functions. As evidenced by these figures, MEP consistently outperforms FEP at the end of the search. The performance difference is prominent especially when Pc takes a large value. In particular, when $Pc = 0.9$, MEP achieves the best performance over four functions. The setting of $Pc = 0.5$ leads to the best performance on the remaining one function f_3.

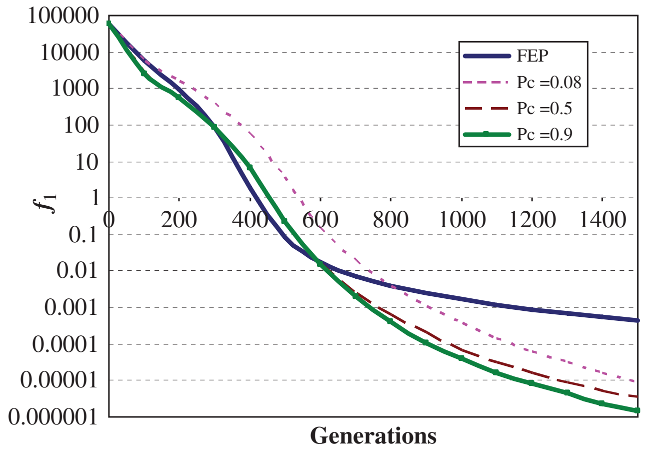

Fig. 3. Evolutionary process of MEP and FEP for function f_1.

Our experiments suggest that R5 helps to accelerate the search process towards the global optimum. In other words, the combined solutions created according to (4) may drive the search quickly towards high-fitness regions such that focused search can be conducted to identify near-optimal solutions. This explanation is supported by the experimental results to be reported at the end of this section, where it is shown that both NSDE and ODE work well with unimodal functions. Notice that NSDE and ODE are variations of DE. They are all based on an operation similar to (4) for creating new candidate solutions.

2) Optimizing Multimodal Functions: Functions having many local optima are usually considered more difficult to

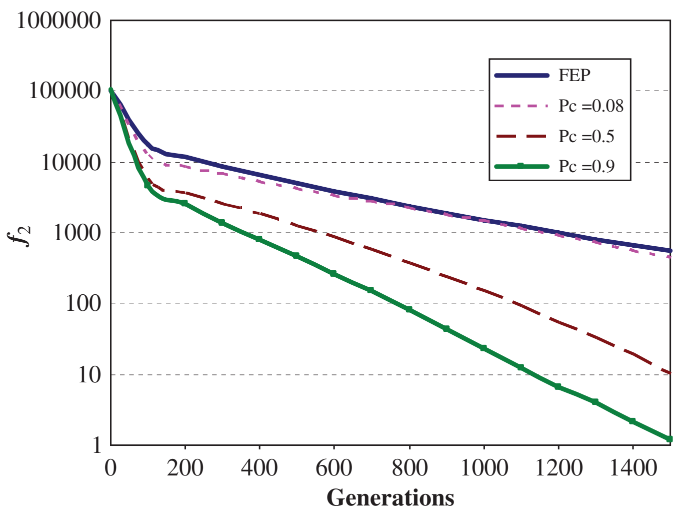

Fig. 4. Evolutionary process of MEP and FEP for function f_2.

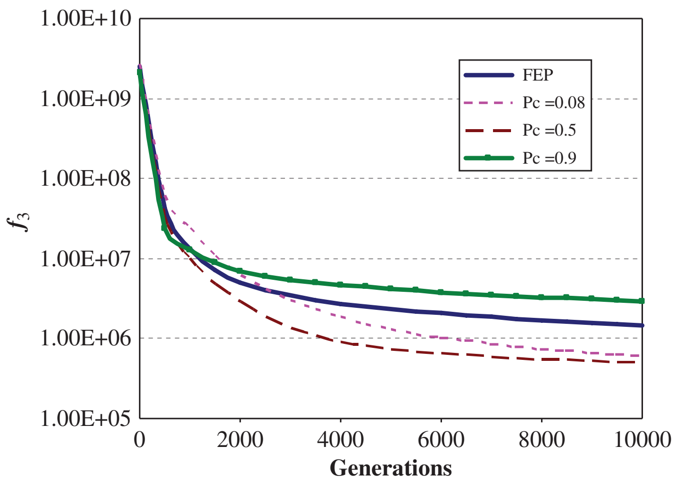

Fig. 5. Evolutionary process of MEP and FEP for function f_3.

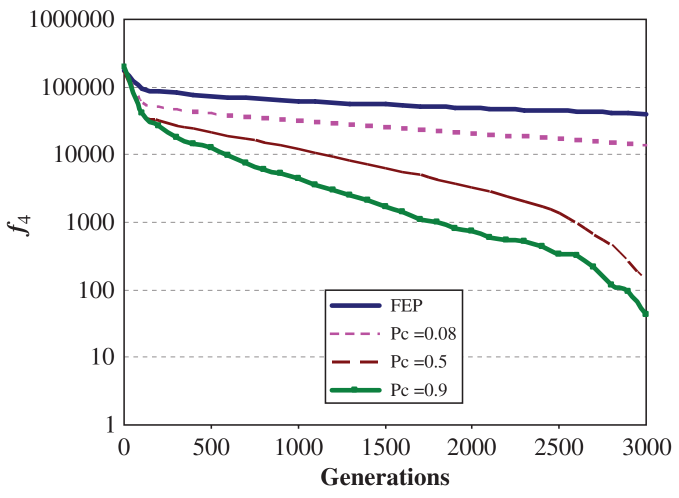

Fig. 6. Evolutionary process of MEP and FEP for function f_4.

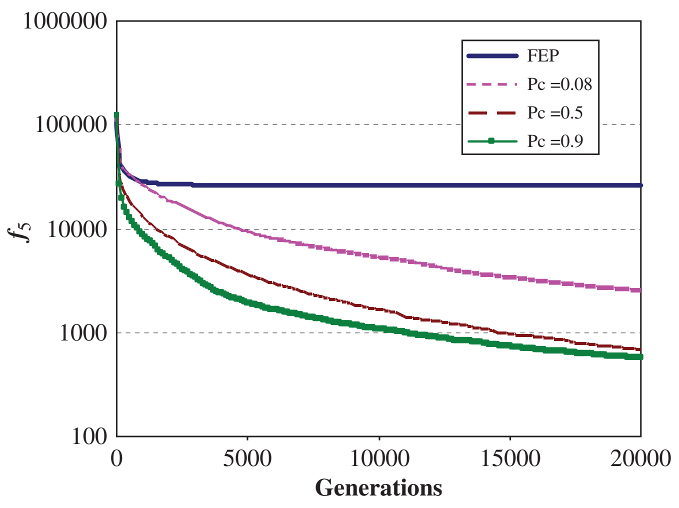

Fig. 7. Evolutionary process of MEP and FEP for function f_5.

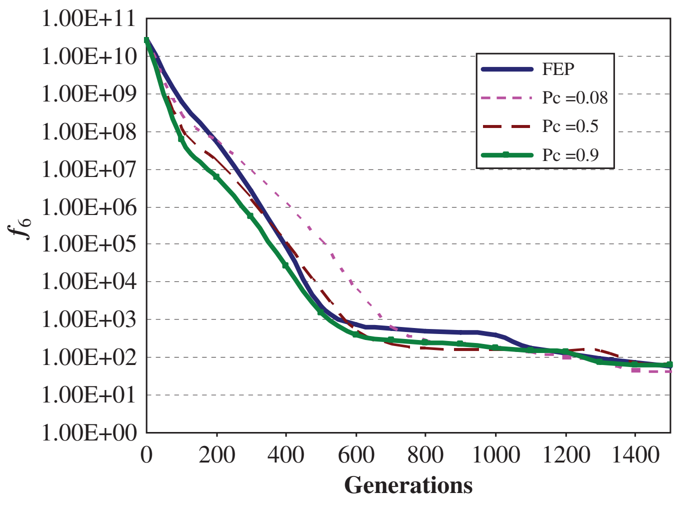

Fig. 8. Evolutionary process of MEP and FEP for function f_6.

optimize than unimodal functions. With respect to the nine multimodal functions $f_6 - f_{14}$, the mean best solutions found by MEP and FEP at each generation are summarized respectively in Figs. 11–16. Again the results are obtained after 25 independent experiments.

Based on Figs. 11–16 as well as the experimental results in Table I, it is found that excessive use of R5 failed to offer a consistent advantage over FEP as in the case of unimodal functions. Specifically, for five functions, the best performance of MEP is observed when $Pc = 0.08$. Meanwhile, for four functions (i.e., f_6, f_8, f_{12}, and f_{13}), FEP even slightly outperforms MEP with $Pc = 0.9$. This observation suggests that, depending on the functions considered, increased use of R5 may actually distract the search process and slow down the convergence speed, as illustrated by Figs. 10, 14, and 15.

Nevertheless, due to the inherent complexity of multimodal functions, R5 can become very useful as well since, at least for three functions (i.e., f_7, f_{10}, and f_{14}), the best performance of MEP is obtained when $Pc = 0.9$. Overall, the appropriate

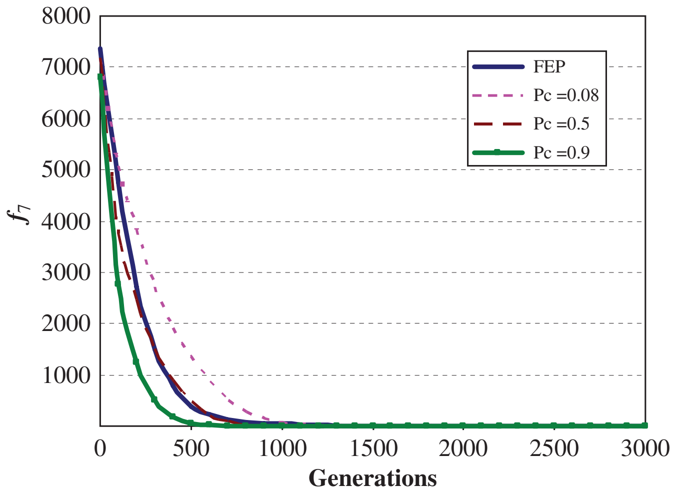

Fig. 9. Evolutionary process of MEP and FEP for function f_7.

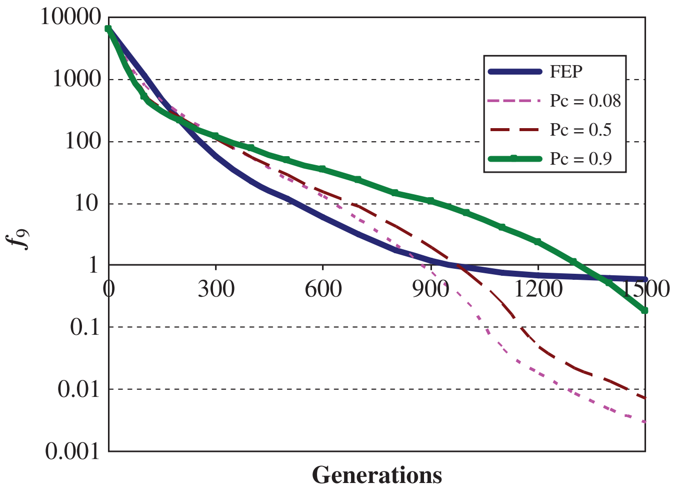

Fig. 11. Evolutionary process of MEP and FEP for function f_9.

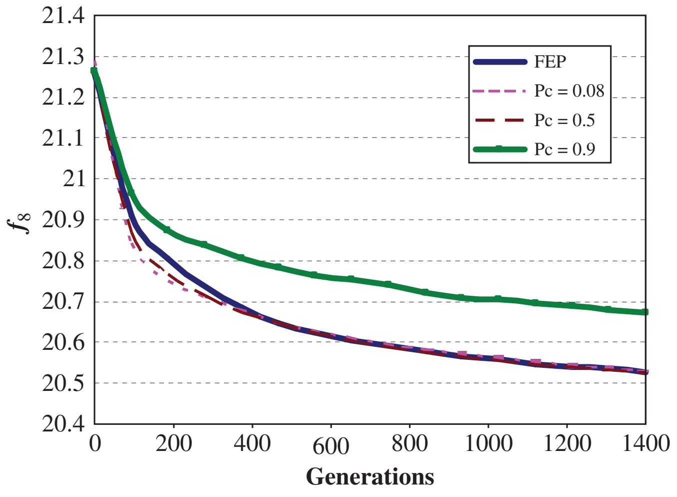

Fig. 10. Evolutionary process of MEP and FEP for function f_8.

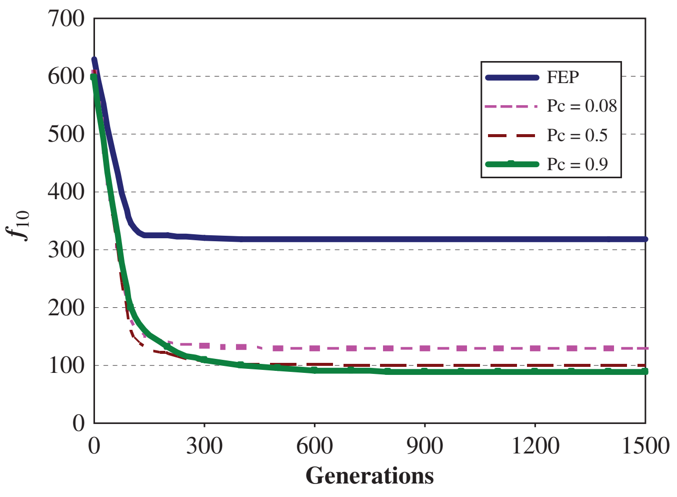

Fig. 12. Evolutionary process of MEP and FEP for function f_{10}.

value of Pc is closely related to the functions being solved. After considering the extra computational cost involved with large Pc, we believe that, for ordinary multimodal functions, Pc is better to take a small value not greater than 0.5 in order to obtain a good balance between performance and cost. As we highlighted in Section III-C, the effectiveness of MEP may be further improved by exploiting a mechanism that can adaptively adjust the value of Pc based on past search history. The applicability of this idea will be investigated in the future.

B. Performance of MEP and FEP on Hybrid Composition Functions

Table II compares the performance of MEP and FEP with respect to a group of hybrid composition functions $f_{15}-f_{25}$. The experiments are performed with exactly the same settings as those reported in Section V-A. Hybrid composition functions are considered far more complex than unimodal and multimodal functions. As a consequence, as shown in Table II,

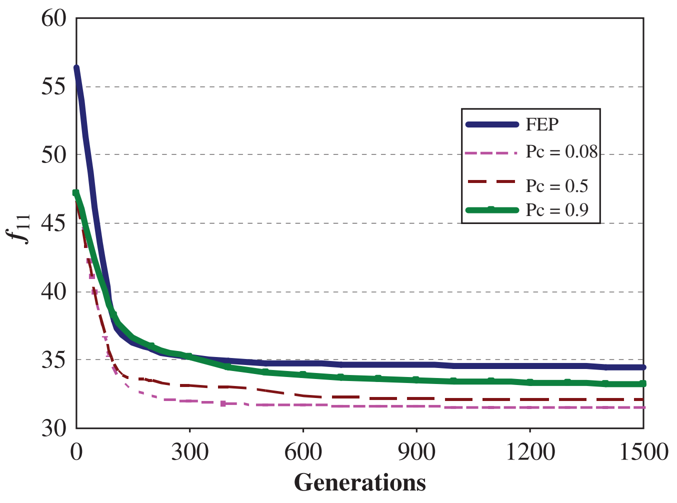

Fig. 13. Evolutionary process of MEP and FEP for function f_{11}.

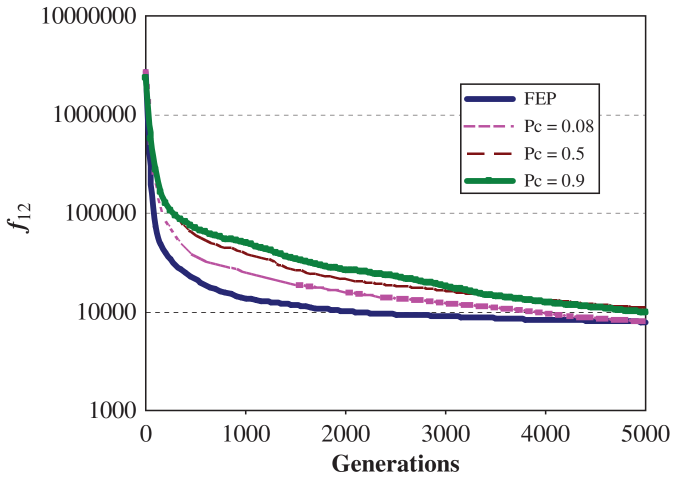

Fig. 14. Evolutionary process of MEP and FEP for function f_{12}.

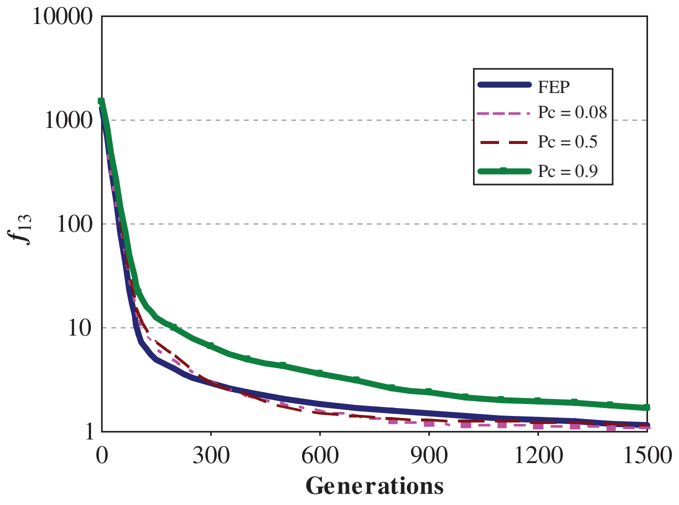

Fig. 15. Evolutionary process of MEP and FEP for function f_{13}.

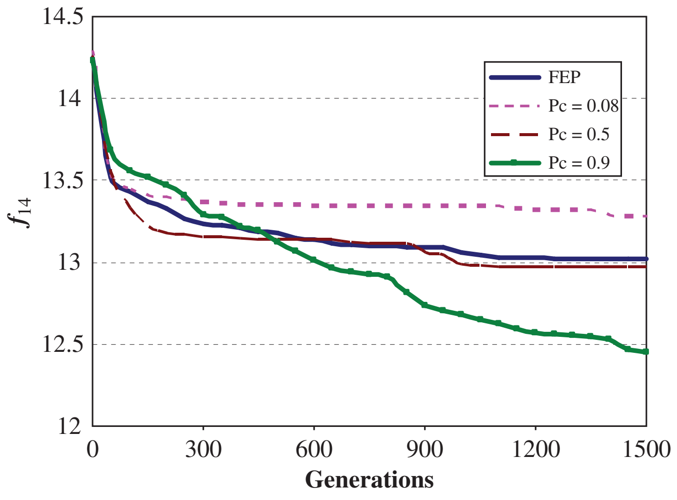

Fig. 16. Evolutionary process of MEP and FEP for function f_{14}.

no algorithms have successfully identified near optimal solutions.

Similar to Section V-A, MEP performs consistently better than FEP over all the 11 functions, and sometimes the advantage of MEP is very prominent as well. This serves as a good indication that rules R1–R4 are quite effective for hybrid composition functions. Additionally, the combination operation as defined by (4) seems to be very useful, since extensive use of R5 will generally lead to comparatively good results. For nine functions, the best mean performance of MEP is witnessed when $Pc = 0.9$. This observation is further supported by the experiments of NSDE and ODE, which suggest that the two algorithms also work relatively well for hybrid functions.

Based on the experiment results reported so far, it possible to draw the conclusion that MEP, which is essentially a generalization of FEP, is more suitable for addressing FOPs. Through preserving and exploiting genetic diversity, the five rules R1–R5 introduced in this paper exhibit demonstrated effectiveness as they significantly enhanced the performance of FEP.

C. Comparison With the Performance of NSDE, ODE, and rGA

The effectiveness of MEP will be further examined in this section through a comparison study that utilizes three competing evolutionary algorithms, namely NSDE, ODE, and rGA. We choose these algorithms because they have been proved to be very effective in handling complex FOPs. Especially in the case of NSDE and ODE, both of them are recently proposed evolutionary algorithms that adopt variations of (4) to explore the solution space. This fact stands for a good basis for our comparison experiments. Additionally, rGA is a highly popular algorithm and has been extensively investigated in the literature. The SBX crossover operator and the polynomial mutation operator used by rGA in our experiments are also strongly recommended for solving FOPs.

Table III lists the results derived from the experiments based on the 25 benchmark functions of CEC2005. Also included in Table III is the performance of MEP when $Pc = 0.5$. For each row of this table, the best performed algorithm is considered the competition winner and is marked in bold face. As evidenced in Table III, MEP is able to achieve a level of performance which is comparable to NSDE, ODE, and rGA. For the 25 functions, MEP managed to win on 12 functions, which cover almost half of the competitions. Specifically, over the nine multimodal functions f_6–f_{14}, MEP achieves the best performance on six functions and only loses on three functions f_6, f_7, and f_{12}, whereas for unimodal and hybrid composition functions, the number of competitions won by MEP is comparable to the number of competitions won by other algorithms.

Notice that, among the 25 functions, ODE outperforms FEP on 17 functions. A similar result is witnessed for NSDE as well. In association with Table III, it is therefore clear that MEP can significantly enhance the effectiveness of FEP such

TABLE II
EXPERIMENTAL RESULTS OF MEP AND FEP FOR HYBRID COMPOSITION FUNCTIONS, AVERAGED OVER 25 INDEPENDENT RUNS. FOR EACH FUNCTION, THE BEST MEAN PERFORMANCE ACHIEVED BY MEP IS MARKED USING BOLD FACE. COLUMN t-TEST (1) LISTS THE T-TEST RESULTS BETWEEN THE BEST PERFORMED MEP AND FEP. COLUMN t-TEST (2) LISTS THE T-TEST RESULTS BETWEEN MEP WITH $Pc = 0.08$ AND FEP

Test function	Number of generations	**FEP**	**MEP**			t-test (1)	t-test (2)
			$Pc = 0.08$	$Pc = 0.50$	$Pc = 0.90$		
f_{15}	1500	318.2	240.8	315.7	**216.8**	14.9	10.7
f_{16}	1500	440.2	370.8	316.4	**268.1**	61.7	24.9
f_{17}	1500	445.3	380.9	320.4	**280.6**	58.2	21.8
f_{18}	1500	980.9	944.6	972.4	**920.7**	15.4	8.65
f_{19}	1500	983.4	948.1	975.2	**925.2**	14.1	7.88
f_{20}	1500	982.4	946.4	977.0	**922.7**	16.3	9.23
f_{21}	1500	920.4	620.1	556.3	**500.1**	14.8	10.3
f_{22}	1500	1097.1	924.6	902.5	**889.0**	54.3	31.2
f_{23}	1500	1097.6	551.6	568.4	**534.2**	35.7	21.5
f_{24}	1500	850.5	620.8	**460.0**	704.4	8.81	3.69
f_{25}	1500	1662.6	1643.8	**1641.2**	1652.1	3.34	2.56

TABLE III
MEAN PERFORMANCE OF NSDE, ODE, AND RGA, AVERAGED OVER 25 INDEPENDENT RUNS. ALSO ENCLOSED IN THE TABLE IS THE PERFORMANCE OF MEP WHEN $Pc = 0.5$. FOR EACH FUNCTION, THE BEST PERFORMANCE ACHIEVED BY ONE OF THE ALGORITHMS IS MARKED USING BOLD FACE

Test function	MEP ($Pc = 0.5$)	ODE	NSDE	rGA
f_1	3.55e–6	**8.04e–28**	7.09e–11	0.099
f_2	**10.409**	270.5	1404.3	3161.2
f_3	**5.041e+5**	2.28e+7	1.80e+7	4.09e+6
f_4	130.4	771.6	**6.71**	83.76
f_5	690.9	**454.8**	574.1	1098.5
f_6	70.09	25.87	**21.77**	139.4
f_7	0.288	0.288	0.288	**0.016**
f_8	**20.52**	20.98	20.98	21.11
f_9	**7.11e–3**	100.83	114.62	298.4
f_{10}	**100.79**	164.8	206.5	241.8
f_{11}	**32.07**	40.57	40.23	40.35
f_{12}	1.097e+4	2.17e+4	**9.13e+3**	1.901e+4
f_{13}	**1.167**	13.25	17.01	8.44
f_{14}	**12.97**	13.35	13.43	14.28
f_{15}	**315.7**	370.2	346.3	532.3
f_{16}	**316.4**	318.9	358.3	511.2
f_{17}	320.4	**277.8**	312.6	684.4
f_{18}	972.4	915.6	**914.2**	963.4
f_{19}	975.2	918.3	**917.3**	1006.2
f_{20}	977.0	**916.57**	916.6	984.7
f_{21}	556.3	**500.0**	**500.0**	1092.3
f_{22}	902.5	883.2	881.3	**737.1**
f_{23}	568.4	**534.16**	534.17	1016.9
f_{24}	**460.0**	486.3	848.7	478.7
f_{25}	**1641.2**	1649.7	1649.1	1668.3

that it becomes a good replacement of NSDE, ODE, and rGA in many situations.

VI. CONCLUSION

This paper has been aimed at preserving and exploiting genetic diversity in evolutionary programming (EP) algorithms. Instead of following *tournament selection*, three selection rules R1–R3 have been proposed in order to encourage both fitness diversity and solution diversity. Meanwhile, two solution exchange rules R4 and R5 have been introduced in an attempt to further exploit preserved genetic diversity. Theoretical analysis has been performed, which showed that through proper use of the five rules R1–R5, during each generation, MEP will enjoy more opportunities of identifying high-fitness solutions. Our claims have also been examined on 25 benchmark functions. The experimental results strongly indicate that MEP can significantly enhance the performance of FEP due to the use of R1–R5.

Overall, this paper not only proved the importance of preserving and exploiting genetic diversity, but also presented a practically applicable approach for achieving this objective. Because of the inherent simplicity of R1–R5, we believe that they can be possibly utilized to improve the performance of other evolutionary algorithms in order to address a wide range of problems such as multiobjective and constrained optimization problems. Nevertheless, more research efforts are required to verify this idea.

There are many other research potentials. For example, it is interesting to see whether the five rules presented in this paper can be applied *adaptively*, based on the functions being optimized. That is, driven by statistical data observed over past generations, MEP is able to adjust, for example, the combination probability Pc and the exponent parameter α in order to maintain a high convergence speed at different search stages. One technique that utilizes fuzzy rules has been suggested in [31] for GAs. More studies are necessary before similar adaptive rules can be established for MEP as well.

REFERENCES

[1] X. Yao and Y. Xu, "Recent advances in evolutionary computation," *Int. J. Comput. Sci. and Technol.*, vol. 21, no. 1, pp. 1–18, 2006.

[2] D. B. Fogel, *Evolutionary Computation: Toward New Philosophy of Machine Intelligence*, 1st ed. New York: IEEE Press, 1995, ch. 3, sec. 3.3.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[4] D. B. Fogel and J. W. Atmar, "Comparing genetic operators with gaussian mutation in simulated evolutionary processes using linear systems," *Biological Cybern.*, vol. 63, no. 2, pp. 111–114, Jun. 1990.

[5] D. B. Fogel, "Applying evolutionary programming to selected traveling salesman problems," *Cybern. Syst.*, vol. 24, no. 1, pp. 27–36, Jan–Feb. 1993.

[6] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[7] C. Y. Lee and X. Yao, "Evolutionary programming using mutations based on the levy probability distribution," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 1–13, Feb. 2004.

[8] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection," in *Proc. 6th Int. Conf. Genetic Algorithms*, Pittsburgh, PA, 1995, pp. 9–16.

[9] S. Legg, M. Hutter, and A. Kumar, "Tournament versus fitness uniform selection," in *Proc. 2004 Congr. Evol. Comput.*, Portland, OR, pp. 2144–2151.

[10] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1991, pp. 69–93.

[11] D. Whitley, "The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms*, San Francisco, CA, 1989, pp. 116–123.

[12] M. Hutter and S. Legg, "Fitness uniform optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 568–589, Oct. 2006.

[13] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genetic Algorithms Applicat.*, Hillsdale, NJ, 1987, pp. 41–49.

[14] Z. G. Wang and Y. S. Wong, "Multi-niche crowding in the development of parallel genetic simulated annealing," in *Proc. 2005 Conf. Genetic Evol. Comput*, Washington DC, pp. 1555–1556.

[15] R. J. Collins and D. R. Jefferson, "Selection in massively parallel genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, San Diego, CA, 1991, pp. 249–256.

[16] G. Rudolph, "On takeover times in spatially structured populations: Array and ring," in *Proc. 2nd Asia-Pacific Conf. Genetic Algorithms Applicat.*, Hong Kong, 2000, pp. 144–151.

[17] J. Garnier, L. Kallel, and M. Schoenauer, "Rigorous hitting times for binary mutations," *Evol. Computing*, vol. 7, no. 2, pp. 167–203, 1999.

[18] J. He and X. Yao, "Towards an analytic framework for analysing the computation time of evolutionary algorithms," *Artificial Intell.*, vol. 145, no. 1–2, pp. 59–97, Apr. 2003.

[19] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, (2005, Dec. 1). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan

[20] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proc. 2007 IEEE Congr. Evol. Comput.*, Singapore, pp. 3523–3530.

[21] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[22] D. B. Fogel, "Evolving artificial intelligence," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. California, San Diego, CA, 1992.

[23] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 1–23, 1993.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[25] R. Thomsen, "Flexible ligand docking using differential evolution," in *Proc. 2003 Congr. Evol. Comput.*, Canberra, Australia, pp. 2354–2361.

[26] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 82–102, Apr. 1999.

[27] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic strategy for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[28] M. Iosifescu, *Finite Markov Processes and Their Applications*, 1st ed. New York: Wiley, 1980, ch. 2, sec. 1, pp. 99–101.

[29] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," Dept. Mech. Eng., Indian Inst. Technol., Kanpur, India, Tech. Rep. Litk/Me/Smd-94027, 1994.

[30] M. M. Raghuwanshi and O. G. Kakde, "Survey on multiobjective evolutionary and real coded genetic algorithms," *Complexity Int.*, vol. 11, no. 1, pp. 150–161, Jan. 2005.

[31] J. Zhang, H. S. H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 326–335, Jun. 2007.

Gang Chen received the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2006.

He is currently a Visiting Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His main research interests are multiagent systems, peer-to-peer networks, machine learning, and evolutionary algorithms.

Chor Ping Low (M'95) received the Ph.D. degree in computer science from the National University of Singapore in 1994.

He is currently an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His current research interests include network performance modeling and analysis, combinatorial optimization, and engineering informatics.

Zhonghua Yang (SM'02) received the Ph.D. degree in computing and information technology from Griffith University, Brisbane, Australia.

He is currently an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His earlier career included working for Singapore Institute of Manufacturing Technology, Griffith University, University of Queensland, St. Lucia, Australia, University of Alberta, Canada, and Imperial College, London, U.K. He spent a significant part of his career with the Chinese aerospace industry. His research interests include grid/distributed computing, semantic web services, and multiagent systems.