Population-Based Algorithm Portfolios for Numerical Optimization

Fei Peng, *Student Member, IEEE,* Ke Tang, *Member, IEEE,* Guoliang Chen, and Xin Yao, *Fellow, IEEE*

***Abstract*—In this paper, we consider the scenario that a population-based algorithm is applied to a numerical optimization problem and a solution needs to be presented within a given time budget. Although a wide range of population-based algorithms, such as evolutionary algorithms, particle swarm optimizers, and differential evolution, have been developed and studied under this scenario, the performance of an algorithm may vary significantly from problem to problem. This implies that there is an inherent risk associated with the selection of algorithms. We propose that, instead of choosing an existing algorithm and investing the entire time budget in it, it would be less risky to distribute the time among multiple different algorithms. A new approach named population-based algorithm portfolio (PAP), which takes multiple algorithms as its constituent algorithms, is proposed based upon this idea. PAP runs each constituent algorithm with a part of the given time budget and encourages interaction among the constituent algorithms with a migration scheme. As a general framework rather than a specific algorithm, PAP is easy to implement and can accommodate any existing population-based search algorithms. In addition, a metric is also proposed to compare the risks of any two algorithms on a problem set. We have comprehensively evaluated PAP via investigating 11 instantiations of it on 27 benchmark functions. Empirical results have shown that PAP outperforms its constituent algorithms in terms of solution quality, risk, and probability of finding the global optimum. Further analyses have revealed that the advantages of PAP are mostly credited to the synergy between constituent algorithms, which should complement each other either over a set of problems, or during different stages of an optimization process.**

Manuscript received October 14, 2008; revised March 20, 2009, August 31, 2009, and November 30, 2009. Date of publication March 29, 2010; date of current version October 1, 2010. This paper was partially supported by the National Natural Science Foundation of China under Grants 60533020, 60802036 and U0835002, the Fund for Foreign Scholars in University Research and Teaching Programs in China under Grant B07033, and the Engineering and Physical Science Research Council in U.K. under Grant EP/D052785/1 on "SEBASE: Software Engineering By Automated Search."

F. Peng and K. Tang are with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: pzbhlp@mail.ustc.edu.cn; ketang@ustc.edu.cn).

G. Chen is with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China, and also with the National High-Performance Computing Center, Hefei 230027, China (e-mail: glchen@ustc.edu.cn).

X. Yao is with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China, and also with the Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2010.2040183

***Index Terms*—Algorithm portfolios, global optimization, metaheuristic algorithms, numerical optimization, population-based algorithms.**

I. Introduction

In the past decades, numerous population-based algorithms, such as evolutionary algorithms (EAs) [1], particle swarm optimizers [2], and differential evolution [3], have been used for numerical optimization. However, their performance may vary greatly from one problem to another and there is no best algorithm for all problems. In practice, there is an inherent risk associated with choosing a single algorithm for a given problem, as we do not know in advance which algorithm is optimal for the problem. For many real-world applications, finding the global optimum may not be practical or even possible. The best solution found by a given deadline is often considered to be very important. As opposed to an algorithm that performs extremely well on some problems but very poorly on others, an algorithm that works well on a large variety of problems would be more desirable for many real-world applications (of course, we do not want to depress the performance of the algorithm). To obtain such an algorithm, two issues need to be addressed. First, given a set of problems from different classes, the question of how to evaluate the risk of applying an available algorithm to these problems needs to be answered. Second, we need to find a way of minimizing this risk, i.e., to develop a low-risk algorithm.

Unfortunately, no formal measurable definition of an algorithm's risk on a set of different problems is available so far, not to mention any effort on reducing the risk. This paper presents a metric for comparing the risks associated with two algorithms. It also proposes a population-based algorithm portfolio (PAP), which not only reduces the risk of failing on problems from different classes, but also makes finding high-quality solutions more likely.

The basic idea of PAP is simple: instead of betting the entire time budget on a single algorithm, we "invest" our time in multiple algorithms. This idea has been explored for more than ten years, based on the theory of investment portfolios that was developed in the field of economics to answer the question: "How should one allocate his/her financial assets (stocks, bonds, etc.) in order to maximize the expected returns while minimizing risks" [4]. In analogy with investment portfolios, PAP focuses on the problem of how to allocate computation time among algorithms and fully utilize the advantages of these algorithms in order to maximize the expected utility of a

```
migration_size: the number of emigrants in each migration;
migration_interval: the generation numbers between two subsequent migrations;
Procedure: PAP
Initialize: Generate $m$ initial subpopulations, $pop_1, pop_2, \ldots, pop_m$;
While Stopping conditions are not satisfied do
    Evaluate all the individuals in $pop_1, pop_2, \ldots, pop_m$;
    For $i = 1 : m$
        Evolve $pop_i$ using algorithm $i$;
    End for
    If an interval of migration_interval generations is reached then activate migration procedure
        For $i = 1 : m$
            For $pop_i$, select migration_size best individuals from the other $m$-1 subpopulations,
            denoted as set $emigrants_i$;
            Set $pop'_i$ to $pop_i \cup emigrants_i$;
            Discard the migration_size worst individuals in $pop'_i$;
        End for
        For $i = 1 : m$
            Set $pop_i$ to $pop'_i$;
        End for
    End if
```

Fig. 1. Pseudo-code of PAP.

problem solving episode. Based on the portfolio idea, various approaches have been developed, either to minimize the expected time needed to solve problem instances or to maximize the expected quality of the solution while the available time is kept constant. However, all the existing approaches were proposed for combinatorial problems. For example, Huberman *et al.* [5] proposed an economic approach for constructing "portfolios" of multiple algorithms to solve combinatorial problems. They showed how executing two or more Las Vegas algorithms could significantly shorten the expected time for finding a feasible solution to the graph coloring problem. Gomes and Selman [6] applied a Las Vegas algorithm portfolio to the quasi-group completion problem. The purpose was also to reduce the expected time for finding a solution. In their empirical study, the portfolio approach led to a significant performance improvement. Fukunaga [4] extended the algorithm portfolio approach to resource-bounded combinatorial optimization by combining instances of EAs with multiple sets of control parameter value settings into algorithm portfolios.

Compared to the existing work on algorithm portfolios, this paper has two major contributions. First, the proposed PAP is developed for numerical optimization instead of combinatorial optimization. Second, we investigate the term "risk" in a different context. Previous work mainly aimed to reduce the risk of an algorithm on a specific optimization problem, which can be measured by applying the algorithm to the problem for multiple times. In this paper, we are more interested in reducing the risk over *a set of problems*, i.e., the risk is measured by applying the algorithm to *multiple* problems.

Our PAP is a general framework for combining different population-based search algorithms. It allocates computation time among more than one constituent algorithm and activates interaction among them so that they can benefit from one another. To be more specific, the computation time is measured by the number of fitness evaluations (FEs), and the allocation of FEs is implemented by dividing the whole population into several subpopulations and evolving them in a parallel manner (using different constituent algorithms). To demonstrate the efficacy of the proposed PAP, we chose four existing population-based algorithms, including self-adaptive differential evolution with neighborhood search (SaNSDE) [7], particle swarm optimizer with inertia weight (wPSO) [8], generalized generation gap (G3) model with generic parent-centric recombination (PCX) operator (G3PCX) [9], and covariance matrix adaptation evolution strategy (CMA-ES) [10], as the constituent algorithms. These algorithms belong to four well-known families of EAs and thus enable us to investigate PAP in the general context of population-based algorithms. Eleven instantiations of PAP were implemented using these constituent algorithms. Six of them consist of two constituent algorithms, another four are composed of three constituent algorithms, and the remaining one contains four constituent algorithms. The PAP instantiations were then evaluated on both the classical benchmark functions used by Yao *et al.* [11] and CEC2005 benchmark functions [12].

The rest of this paper is organized as follows. Section II describes the proposed PAP in details. Section III presents a new metric for comparing the risks associated with two algorithms. Experimental studies are presented in Section IV. Section V presents some further analyses to investigate the characteristics of PAP more comprehensively. Finally, we draw the conclusions and discuss future work in Section VI.

II. Population-Based Algorithm Portfolios for Reducing Risk

A. General Framework

Assume that an optimization problem f, some computation time (i.e., FEs) and m population-based algorithms $A = \{A_i | i = 1, 2, ..., m\}$ are given. We aim to find the optimal solution to the problem. In PAP, computation time is allocated to the m algorithms before the problem-solving process, and then the m algorithms search for the optimal solution within their own time budgets. Specifically, PAP holds m separate subpopulations, each constituent algorithm works on one of them. Migration of individuals among subpopulations is activated regularly to encourage information sharing. PAP adopts a very simple migration scheme. Two parameters $migration\_interval$ and $migration\_size$ are defined in advance, which determine the number of generations between two subsequent migrations and the number of emigrants in each migration, respectively. Every time a migration is activated, it involves all constituent algorithms. For each constituent algorithm (say, A_i), we combine the subpopulations of the remaining $m - 1$ algorithms, identify and duplicate the best $migration\_size$ individuals. Then, these copies are migrated to the subpopulation of A_i. After that, the worst $migration\_size$ individuals of the new subpopulation of A_i (i.e., with the emigrants) will be discarded. PAP terminates when the given FEs have been used up. The pseudo-code of PAP is shown in Fig. 1.

In practice, it is often the case that an algorithm continuously finds better solutions at the early stage of the search. As the search proceeds, it becomes more and more difficult to find a solution that is better than the best one found so far. In the worst case, the search might be trapped in a local optimum after a certain number of FEs and no further improvements can be made after that. If such a scenario occurs before the time budget is used up, keeping on running the same algorithm using the remaining FEs is a waste of time. By investing computation time in multiple algorithms, it is more likely that the search will keep progressing along the whole search episode, since the chance of all the constituent algorithms being stuck at the same point in the solution space is quite small. From the risk reduction perspective, the computation time allocation strategy alone reduces the risk of suffering from premature convergence. One might argue that we can switch to another algorithm once an algorithm is trapped in a local optimum for a long time. However, it is never easy to decide when to switch, and to which algorithm we should switch. Instead, allocating computational resources in advance might be a better choice due to its ease of implementation. The PAP presented in Fig. 1 can be seen as a first attempt to implement the resource allocation idea for numerical optimization.

In addition to the allocation of computation time, migration of individuals among subpopulations is also indispensable to PAP. Let us view the quality of the solution obtained by an algorithm as a function of the computation time t it consumes. As long as the best solution found so far is always retained during the search process (either in the population or at some other place), running an algorithm with additional time will never deteriorate the quality of the final solution. This fact can be formally presented as the following inequality:

$$E(q(A_i, f, t_i)) \leq E(q(A_i, f, T))), \quad \text{if } t_i \leq T \tag{1}$$

where $q(A_i, f, t_i)$ denotes the quality of the best solution obtained by A_i with time t_i. We use the expectation form because $q(A_i, f, t_i)$ is inherently a random variable that depends on random initializations. Since PAP allocates the computation time before searching for solutions, the time t_i assigned to A_i can be viewed as a constant. Hence, if the constituent algorithms of PAP conduct search independently (i.e., no migration), the following inequality holds for any problem f:

$$\begin{aligned} E[q(PAP_{ind}, f, T)] &= \max\{E[q(A_i, f, t_i)]\} \\ &\leq \max\{E[q(A_i, f, T)]\}, \quad i = 1, ..., m, \\ &\text{Subject to} \sum_{i=1}^{m} t_i = T \end{aligned} \tag{2}$$

where PAP_{ind} denotes PAP whose constituent algorithms work independently. Equation (2) shows that allocating computation time to multiple algorithms and running them independently is really a bad idea. Hence, migration among subpopulations is necessary so as to break the law demonstrated by (2).

B. Related Work

The PAP can be viewed as a framework that utilizes different search biases for problem solving. In this section, we review some previous relevant work and discuss the differences between them and the PAP. In general, previous work related to combining different search biases can be summarized into two categories.

The first category attempts to integrate multiple search biases from operators or algorithms. All the constituent operators or algorithms work on a shared population of solutions. Every constituent operator/algorithm has access to any solution in the population. The major effort for designing such a method is to decide for each constituent operator/algorithm when to work, what to work on, and how often to work. In the improved fast evolutionary programming [11], two offspring are generated for each parent, one by Cauchy mutation and the other by Gaussian mutation. The two offspring are then compared and the better one survives to the next generation. The mixed strategy evolutionary programming proposed by Dong *et al.* [15] utilizes four different mutation strategies: Gaussion, Cauchy, Lévy, and single-point mutations. Each individual chooses one of the four mutation strategies to generate its offspring based on a mixed strategy distribution, which is dynamically adjusted according to the performance of mutation strategies. In [16], a self-adaptive DE algorithm was proposed. It mixes four different DE mutation variants using a probabilistic model and adapts the probability of the four variants during the search based on their previous success rates. Whitacre *et al.* [17] and Thierens [18] independently investigated approaches for adapting the probabilities of applying different operators. DaCosta *et al.* [19] focused more on adapting the operator selection scheme with the Multiarmed Bandit paradigm.

More recently, Vrugt and Robinson [20] proposed a multialgorithm genetically adaptive method (AMALGAM), which incorporates various algorithms in one framework. At each generation, offspring are generated using different algorithms [20], and the number of offspring created by each algorithm is adapted based on their performance in the previous generations. Another well-studied approach, namely the asynchronous teams (A-Teams), works in a similar manner [21], [22]. A lot of emphasis has been put into developing universal structures or even a toolbox for generalizing A-Teams to various real-world problems [23]. At the same time as this paper was under review, Mallipeddi and Suganthan [24] proposed an ensemble of constraint handling techniques (ECHT) for constrained optimization problems. ECHT also utilizes multiple subpopulations, each of which is assigned with a constraint handling technique. At each generation, each subpopulation generates new individuals based on its own. Then, each subpopulation is combined with new individuals generated by all subpopulations, and selection is conducted with the corresponding constraint handling technique. Compared to all the above-mentioned work, in which all search biases have access to the whole population, PAP allows only its constituent algorithms to work on a subpopulation, i.e., full access to the whole population is prohibited. Migration is the only route that different subpopulations communicate with each other. This strategy, used by PAP, reduces the likelihood of different constituent algorithms repeating similar search behaviors or sharing similar search biases.

From the perspective of employing a multipopulation model and migration operators, PAP is related to the distributed EAs (dEAs), which have several subpopulations (islands) and perform sparse exchange of individuals among them [25], [26]. However, PAP is quite different from previous dEAs in three major ways. First, most dEAs run the same EA on all subpopulations [26]–[31] (either with the same or different control parameters), while PAP employs different EAs. Second, the migration scheme of PAP does not assume any topology of subpopulations, while many dEAs do. By this means, we solely focus on investigating whether it is worthy of allocating the limited computation time to different algorithms so that the overall performance can be enhanced. In contrast, a specific topology/structure for the relationship between subpopulations is usually explicitly defined in existing dEAs [26]–[29], [32], [33]. Third, we are more interested in reducing the overall risk of an algorithm on *a spectrum of problems*. This issue has never been investigated in the literature of dEAs.

In addition to dEAs, another notable work that is related to PAP is the isolation strategy employed by hierarchically organized evolution strategies (HOES) [34], [35]. In HOES, a population is partitioned into a number of subpopulations, and ES is run on them with different control parameters (e.g., search step sizes). The isolation strategy works in such a way that the subpopulations evolve separately for a predefined isolation period (e.g., a few generations), and then the control parameters are updated based on the individuals generated in the subpopulations during the isolation period. Unlike PAP, the motivation of the isolation strategy is not to make use of different search biases, but to optimize the control parameters of a search algorithm, i.e., the adaption of control parameters was formulated as an optimization problem. Such a difference in underlying motivation has led to different research foci between PAP and existing work on isolation strategy.

C. PAP Implementation

In practice, one may implement PAP following a few steps. First, a set of constituent algorithms should be identified. Then, these constituent algorithms need to be coded (if no implementation is readily available), and the migration scheme can be easily implemented with a few lines of codes. Finally, one needs to decide how much computation time to allocate to each constituent algorithm, and set the two migration parameters. When choosing constituent algorithms, an intuition is that they should be more or less complementary. As will be shown by our empirical studies, choosing complementary constituent algorithms leads to better performance than applying the same algorithm to the problems in parallel. More specifically, the constituent algorithms should not only employ different operators, but also exhibit different behaviors on the problem set. A detailed analysis of this issue is given in Section V. For the second step, PAP should be capable of accommodating any existing population-based search algorithms since we expect it to be a general framework for combining different algorithms. However, due to the fact that some existing algorithms might have their own specific configurations, they merit a bit more attention when being incorporated into the PAP framework. Below, we elaborate on a few such cases.

Some algorithms, such as PSO and SaNSDE, store and update the best solution found during the whole course of the optimization in a special variable not belonging to the population (e.g., the *gbest* of PSO). This solution is used to guide the moves of the individuals in the population. When including this type of algorithms into PAP, we suggest updating the *gbest* once a better solution emerges in the whole population, no matter whether it is reached by the constituent algorithms themselves or not. Some other algorithm, like G3PCX, will terminate automatically if the status of the population has reached some predefined criterion. If a constituent algorithm of this type terminates before using up the FEs assigned to it, the remaining FEs will be allocated to the other constituent algorithms in proportion to the sizes of their subpopulations.

Another type of algorithms, including CMA-ES and estimation of distribution algorithms (EDA) [36], depends on a model to generate new solutions. For example, CMA-ES updates its covariance matrix based on the current population, and the next generation of solutions is obtained by sampling according to the covariance matrix. In this case, when a migration occurs and some emigrants survive in the subpopulation of CMA-ES, the covariance matrix should be updated after the migration (i.e., based on the new subpopulation). All the above modifications aim to make existing algorithms compatible with PAP. In practice, they merely involve negligible programming work. Therefore, it is generally straightforward to implement PAP.

The final step before running PAP is to allocate FEs and set the parameters *migration_interval* and *migration_size*. In general, one may allocate FEs according to some prior knowledge about the constituent algorithms. Fortunately, such kind of information is available for most well-investigated algorithms. For example, the population size required by CMA-ES can be calculated by the equation provided in [37], PSO has been shown to work well with a population size of 20–40 [38], [39], while DE usually needs a larger population size (typically 3.33 times of the dimensionality of the problem) than PSO [3], [7], [40]. Alternatively, one may also develop a strategy to adapt the FE allocation scheme during the search procedure. Though the latter approach might deserve further investigation, our experimental study has showed that the former approach works quite well already. With regard to the migration parameters, our experimental study has showed that $migration\_interval = MAX\_GEN/20$ (MAX_GEN denotes the maximum number of generations) and $migration\_size = 1$ work well for all the 11 instantiations of PAP. To understand the impact of PAP parameter setting on its performance, a sensitivity analysis over 16 different pairs of *migration_interval* and *migration_size* has been carried out. The results show that PAP is not so sensitive to the values of these parameters. Hence, the above two values are recommended as the default setting.

III. Metric for Comparing Risks Associated With Two Algorithms

Consider a set of problems $F = \{f_k | k = 1, 2, ..., n\}$ and a set of candidate algorithms $A = \{A_i | i = 1, 2, ..., m\}$. We are interested in the risk of an algorithm A_i on F. Intuitively, the definition of risk should reflect two aspects. First, it should indicate how likely an algorithm would fail to solve a problem in F. Second, if an algorithm performs better on a set of problems, it is associated with smaller risk. Following these intuitions, let us start from an attempt of quantifying the risk of an algorithm. We define the probability of A_i failing on a problem belonging to F as the risk of an algorithm A_i, which can be written as

$$P(A_i \text{ fails to solve a problem belonging to } F) = \sum_{k=1}^{n} P(A_i \text{ fails to solve } f_k | f_k) P(f_k); \, f_k \in F. \quad (3)$$

In case no prior knowledge suggests an alternative distribution, we may further assume that all problems in F have the same prior probability $P(f_k)$. Then, (3) becomes

$$P(A_i \text{ fails to solve a problem belonging to } F) = \frac{1}{n} \sum_{k=1}^{n} P(A_i \text{ fails to solve } f_k | f_k); \, f_k \in F. \quad (4)$$

To use (4) to calculate the risk, the term "failure" must be defined first. Obviously, an off-the-shelf choice is to regard an algorithm as failing on a problem if it did not find the global optimal solution. However, we do not know the real global optimal solution to a real-world problem. Even if we consider the case of benchmark functions, whose global optima are known in advance, it is possible that no algorithm can find the global optimum of a problem. Consequently, if F consists of a lot of such functions, (4) can hardly differentiate between such algorithms, and thus is useless in practice. A possible compromise is to say that A_i fails on f_k if the quality of the obtained solution does not reach a human-defined threshold. However, since the value of the threshold can be very subjective, (4) will also become subjective, and will not be a good metric for evaluating algorithms. In one word, quantifying the risk associated with an algorithm is a nontrivial task due to the difficulty of defining the "failure" of an algorithm on a problem. A more realistic alternative is to compare the risks associated with different algorithms based on the solution quality they achieved. Recall that our ultimate goal of studying risk is to choose an appropriate algorithm. This alternative, although not allowing for an explicit calculation of the risk, is sufficient for our purpose. Therefore, we propose the following method to compare the risks of two algorithms on a set of problems.

Denote $q_{i,k}$ as the quality of the best solution obtained by A_i on f_k in a single run. We consider the following question: given a problem set and two algorithms A_i and A_j, which one is associated with a higher risk on F? Similar to (3) and (4), our answer is: A_i is less risky than A_j *iff* the conditional probability of A_i outperforming A_j with respect to F is larger than the conditional probability of A_j outperforming A_i with respect to F. Here, we say A_i outperforms A_j if $q_{i,k} > q_{j,k}$ (i.e., A_i obtained a solution of higher quality). Under the assumption that every f_k has the same prior probability, the probability of A_i outperforming A_j can be calculated by the following equation:

$$P(A_i \text{ outperforming } A_j | F) = \frac{1}{n} \sum_{k=1}^{n} P(q_{i,k} > q_{j,k} | f_k); \forall f_k \in F. \quad (5)$$

Similarly, we have

$$P(A_j \text{ outperforming } A_i | F) = \frac{1}{n} \sum_{k=1}^{n} P(q_{i,k} < q_{j,k} | f_k); \quad f_k \in F \quad (6)$$

and

$$P(A_i \text{ outperforming } A_j | F) + P(A_j \text{ outperforming } A_i | F) + P(A_i \text{ performs the same as } A_j | F) = 1. \quad (7)$$

Equations (5) and (6) require estimating $P(q_{i,k} > q_{j,k})$ and $P(q_{i,k} < q_{j,k})$. This can be done by applying the algorithms on

f_k for multiple times. Suppose that A_i and A_j are run on f_k for s_i and s_j times, respectively. We randomly pick a solution obtained by A_i and compare it to a randomly chosen solution obtained by A_j. In total, we can get $s_i \times s_j$ distinct pairs of solutions. By counting the times that a solution of A_i beats a solution of A_j and dividing it by $s_i \times s_j$, we get the estimate of the probability that A_i outperforms A_j on f_k. Equation (5) can then be estimated by repeating the above procedure for each f_k in F. The conditional probability of A_j outperforming A_i with respect to F can be estimated in the same way. Since the performance of A_i and A_j depends on the available budget of computation time, so does (5). As a result, the value of (5) may vary significantly with different time budgets.

Equation (5) indicates that the larger the term $P(A_i \text{ outperforming } A_j|F)$, the less risky A_i is in comparison to A_j. Taking a closer look at the estimation of $P(q_{i,k} > q_{j,k})$, we can find that it is closely linked to the test statistic U used in the Wilcoxon rank-sum test [41]. The only difference is that we do not count the case that A_i and A_j perform the same (i.e., our estimation is equivalent to U if A_i and A_j never make a draw in the $s_i \times s_j$ comparisons). This implies that (5) is naturally a metric of the overall performance of A_i on the whole problem set F. The metric is consistent with our intuition that an algorithm is associated with less risk if it overall performs better than another algorithm on most problems. From the practical point of view, one main advantage of (5) is that it "normalizes" the performance of A_i on f_k's in terms of probabilities, so that they can be averaged without biasing to any specific f_k. If directly averaging the solution quality over different problems, those problems whose objective functions have larger magnitudes will dominate those with much smaller magnitudes.

It is important to note that the meaning of the term "risk" is rather vague, and a precise definition might not exist. The criterion proposed in this section should be used within the PAP framework only, as we have done in our empirical studies. When evaluating algorithms, it should be used together with other traditional performance metrics in order to draw a complete picture of the behavior of algorithms.

IV. Experimental Studies

In this section, the effectiveness of PAP is empirically evaluated on 27 benchmark functions. Four algorithms, namely SaNSDE, wPSO, G3PCX, and CMA-ES were employed as the basic constituent algorithms. These algorithms can be used to implement six instantiation of PAP with two distinct constituent algorithms, four instantiations of PAP with three constituent algorithms, and one instantiation of PAP with all four algorithms. To fully evaluate PAP's potential as a general framework, we carried out experiments with all 11 instantiations. They were compared to the four constituent algorithms alone to verify whether there would be any advantages of PAP over its constituent algorithms. We considered a fixed population for each instantiation of PAP. For this reason, the restart CMA-ES with increasing population size (IPOP-CMA-ES or G-CMA-ES) [13], which is an improved version of CMA-ES, was not investigated under the framework of PAP, because it adopts a dynamic population size. However, G-CMA-ES appears to be one of the state-of-the-art algorithms for numerical optimization (according to the technical report of CEC2005 competition [14]). Hence, comparison has also been made between the PAP instantiations and G-CMA-ES. To evaluate the effect of fine-tuning the migration parameters (i.e., *migration_size* and *migration_interval*) of PAP, we carried out a sensitivity analysis involving 16 different settings of the parameters. Therefore, our experimental study altogether involved running 11×16 (the PAP instantiations with different migration parameters) + 4 (the basic algorithms) + 1 (G-CMA-ES) = 181 algorithms on 27 benchmark functions.

A. Problem Set

The first 13 functions were selected from the classical benchmark functions used in [11], here denoted as $f_1 - f_{13}$. The other 14 functions were selected from the benchmark functions of the special session on real-parameter optimization of the 2005 IEEE Congress on Evolutionary Computation (CEC2005) [12], denoted as $f_{cec1} - f_{cec14}$. These 27 functions span a diverse set of problem features, such as multimodality, ruggedness, ill-conditioning, interdependency, etc. They provided an ideal platform for our investigation on reducing risk on a large variety of problems. Short descriptions of these functions are presented in Tables I and II. More details of these functions can be found in [11] and [12]. In our experiments, all the functions were solved in 30 dimensions.

B. Experimental Settings

All the results presented in this paper were obtained by executing 30 independent runs for each experiment. Since we expect the PAP framework to be general enough so that alternative algorithms can be incorporated with little effort, it should not rely much on the refinement of the constituent algorithms. Hence, we did not fine-tune the parameters of the constituent algorithms to fit PAP. When implementing SaNSDE, we used all the parameter settings suggested in the original publication [7]. As suggested in [8], a linearly decreasing inertia weight over the course of the search is employed in our implementation of wPSO. The two coefficients of wPSO were both set to 1.49445. We assumed the researchers who proposed G3PCX and CMA-ES are at the best position to implement the two algorithms and fine-tune the parameters. Hence, we simply used the source code of G3PCX and CMA-ES provided by their authors (the codes are available online), and adopted the parameters suggested in the corresponding publications [9], [10]. There exist a few variants of PCX operator. As suggested in [9], we chose the variant which employs the best individual in the population as the main parent for generating offspring. Furthermore, G3PCX and CMA-ES will terminate when some conditions are met. It is possible that they terminate before using up the allowed FEs. The remaining FEs are assigned to the other constituent algorithms. To make a fair comparison, when running G3PCX and CMA-ES alone, we restarted them in case the termination condition was met before the allowed FEs were used up.

TABLE I

CLASSICAL TEST FUNCTIONS USED IN THIS PAPER, INCLUDING A SHORT DESCRIPTION OF THEIR CHARACTERISTICS

	Unimodal Functions	Characteristics
f_1	Sphere Function	Separable, scalable
f_2	Schwefel's Problem 2.22	Separable, scalable
f_3	Schwefel's Problem 1.2	Nonseparable, scalable
f_4	Schwefel's Problem 2.21	Nonseparable, scalable
f_6	Step Function	Separable, scalable
f_7	Quartic Function, i.e., Noise	Separable, scalable
	Multimodal Functions	
f_5	Generalized Rosenbrock's Function	Nonseparable, scalable, narrow valley from local to global optimum
f_8	Generalized Schwefel's Problem 2.26	Separable, scalable, numerous local optima
f_9	Generalized Rastrigin's Function	Separable, scalable, numerous local optima
f_{10}	Ackley's Function	Separable, scalable, numerous local optima
f_{11}	Generalized Griewank Function	Separable, scalable, numerous local optima
f_{12}	Generalized Penalized Function	Separable, scalable, numerous local optima
f_{13}	Generalized Penalized Function	Separable, scalable, numerous local optima

A detailed description of these functions can be found in [11].

TABLE II

CEC2005 TEST FUNCTIONS USED IN THIS PAPER, INCLUDING A SHORT DESCRIPTION OF THEIR CHARACTERISTICS

	Unimodal Functions	Characteristics
f_{cec1}	Shifted Sphere Function	Shifted, separable, scalable
f_{cec2}	Shifted Schwefel's Problem 1.2	Shifted, nonseparable, scalable
f_{cec3}	Shifted Rotated High Conditioned Elliptic Function	Shifted, rotated, nonseparable, scalable
f_{cec4}	Shifted Schwefel's Problem 1.2 with Noise in Fitness	Shifted, nonseparable, scalable, noise in fitness
f_{cec5}	Schwefel's Problem 2.6 with Global Optimum on Bounds	Nonseparable, scalable
	Multimodal Functions	
f_{cec6}	Shifted Rosenbrock's Function	Shifted, nonseparable, scalable, narrow valley from local to global optimum
f_{cec7}	Shifted Rotated Griewank's Function without Bounds	Rotated, shifted, nonseparable, scalable
f_{cec8}	Shifted Rotated Ackley's Function with Global Optimum on Bounds	Rotated, shifted, nonseparable, scalable
f_{cec9}	Shifted Rastrigin's Function	Shifted, separable, scalable, numerous local optima
f_{cec10}	Shifted Rotated Rastrigin's Function	Shifted, rotated, nonseparable, scalable, numerous local optima
f_{cec11}	Shifted Rotated Weierstrass Function	Shifted, rotated, nonseparable, scalable
f_{cec12}	Schwefel's Problem 2.13	Shifted, nonseparable, scalable
	Expanded Functions	
f_{cec13}	Shifted Expanded Griewank's + Rosenbrock's Function	Shifted, nonseparable, scalable
f_{cec14}	Shifted Rotated Expanded Scaffer's F6 Function	Shifted, nonseparable, scalable

A detailed description of these functions can be found in [12].

For all the algorithms, the maximum number of FEs (MAX_FES) was set to 300 000. All the PAP instantiations worked on a population of size 100, except for the one incorporating wPSO and CMA-ES. The population size of this instantiation was set to 50. The reason is that a PSO-type algorithm typically utilized a population size around 30 in the literature [38], [39], while 14 was recommended as the optimal population size for CMA-ES on 30-D problems [10], [37]. Hence, setting the combination of wPSO and CMA-ES to work on altogether 50 individuals is more consistent with the literature. When allocating FEs to the constituent algorithms, we simply followed a few *ad hoc* rules obtained from the original publications of the corresponding constituent algorithms. First, the population size of CMA-ES was set to 14 throughout the experiments. Second, a population of size 100 was most commonly adopted for problems of dimensionality 30 in the literature of DE [3], [7], while a population size smaller than 50 may lead to significant deterioration of performance. Hence, we always maintained a population size larger than or equal to 50 for SaNSDE. Third, since most studies on PSO utilized a population size of 20, 30 or 40, we kept the population size of wPSO at this level. Finally, all the remaining

TABLE III
SIZES OF SUBPOPULATIONS ALLOCATED TO THE CONSTITUENT ALGORITHMS OF EACH INSTANTIATION OF PAP

	DE	PSO	PCX	ES
DE+PSO	75	25	–	–
DE+PCX	70	–	30	–
DE+ES	86	–	–	14
PSO+PCX	–	30	70	–
PSO+ES	–	36	–	14
PCX+ES	–	–	86	14
DE+PSO+PCX	50	26	24	–
DE+PSO+ES	70	16	–	14
DE+PCX+ES	70	–	16	14
PSO+PCX+ES	–	36	50	14
DE+PSO+PCX+ES	50	18	18	14

DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively.

FEs were allocated to G3PCX. Table III presents the detailed information of the population/subpopulation sizes for all 11 PAP instantiations. In the table, SaNSDE, wPSO, G3PCX, and CMA-ES are denoted by DE, PSO, PCX, and ES for the sake of brevity. When running the basic algorithms alone, the population sizes of SaNSDE, wPSO, G3PCX, and CMA-ES were set to 100, 40, 100, and 14, respectively. The initial population size of G-CMA-ES was set to 14, and doubled for every restart.

In regard to the $migration\_size$ and $migration\_interval$, we considered four values for each of them and thus 16 pairs of parameters were tested. The $migration\_size$ was set to 1, 2, 4, or 8. The $migration\_interval$ was set to the maximum generation (MAX_GEN) divided by 20, 30, 40, or 50. After some preliminary experiments, we found that $migration\_size = 1$ and $migration\_interval = MAX\_GEN/20$ generally worked well for all 11 instantiations, so they are recommended as the default values. For the sake of brevity, only the results obtained with this setting will be presented for comparison between PAP and other algorithms. Results obtained with other settings were mainly used for the sensitivity analysis.

C. Experimental Results

Tables IV–VI present the results (in terms of solution quality)[1] obtained by the 16 algorithms in 30 independent runs. For each algorithm, the best, median, and worst results are given. According to the IEEE Standard for Floating-Point Arithmetic (IEEE 754), the total precision of the double precision floating-point format is 53 bits (approximately 16 decimal digits). Hence, directly carrying out further analyses (e.g., statistical tests) on the output of the program may introduce errors caused by the precision threshold of our computer. To deal with this issue, two solutions will be regarded as the same if the quality of them are both smaller than some predefined value-to-reach. By manually checking the results that each algorithm obtained in each run, we found that 1e−13 was the smallest value-to-reach that would not introduce arithmetic errors (as defined by the computer) into our further analyses. Therefore, we preprocessed the entries of Tables IV–VI with this value-to-reach, i.e., if an output of our computer program was smaller than 1e−13, we replaced it with 0.0. All analyses presented in the rest of this paper were also conducted based on the value-to-reach of 1e−13.

Two-sided Wilcoxon rank-sum tests with significance level 0.05 have been conducted to compare each PAP instantiation with its constituent algorithms and G-CMA-ES. Additional tests have also been conducted with two arbitrarily chosen values-to-reach, 1e−06 and 1e−02, to verify whether different values-to-reach lead to different conclusions. Table VII summarizes the results over the 27 test functions. By comparing the PAP instantiations to their constituent algorithms, we found that PAP outperformed its constituent algorithms in most cases. Taking the results obtained with values-to-reach 1e−13 as an example, negative results were observed only in four cases: wPSO + G3PCX, PSO + CMA-ES, G3PCX + CMA-ES, and wPSO + G3PCX + CMA-ES. A more careful examination on the detailed results of Wilcoxon tests revealed that wPSO and G3PCX were generally inferior to the other two basic algorithms and on quite a few functions both of them performed poorly. We found that wPSO and G3PCX outperformed CMA-ES on only five (f_7, f_8, f_{cec4}, f_{cec9}, f_{cec14}) and two functions (f_8, f_{cec14}), respectively. In other words, the constituent algorithms in these four cases are not complementary. Hence, the success of PAP does depend on some kind of synergy between its constituent algorithms. The above observations also hold in cases of setting value-to-reach to 1e−02 and 1e−06, and thereby support our expectation that PAP is capable of finding better solutions than its constituent algorithms. Moreover, looking at the last column of Table VII, the instantiations of PAP even showed competitive performance in comparison with G-CMA-ES. Particular attention was brought to the combination of SaNSDE and G3PCX, which slightly outperformed G-CMA-ES. Since this instantiation does not take any advantage of CMA-ES, it clearly demonstrated that a combination of some relatively "weak" algorithms can be stronger than a state-of-the-art algorithm.

When assessing the algorithms' risk, this can be done by estimating the probability that one algorithm outperformed the other on the 27 functions. Since calculating the risk metric involves comparing the quality of two solutions, we also implemented it with the value-to-reach of 1e−13, 1e−06, and 1e−02. The calculated probabilities are presented in Table VIII. Again, we compared the PAP instantiations to their constituent algorithms as well as G-CMA-ES. In each cell, the first number is the estimated probability of the PAP instantiation outperforming the corresponding constituent algorithm, and the second number is the estimated probability of the constituent algorithm outperforming the corresponding PAP instantiation. For brevity, we omitted the probability that the two algorithms made a draw. If the first number is larger than the second one, we may say that the PAP is associated

[1]All test functions used in this paper are minimization problems. Thus, we measure the quality of a solution $\mathbf{x}$ via its function error value $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $\mathbf{x}^*$ is the global optimum of f. When discussing the solution quality of an algorithm, we refer to the quality of the best solution (i.e., the final solution) obtained by the algorithm.

TABLE IV

FUNCTION ERROR VALUES OF THE SOLUTIONS OBTAINED BY SANSDE, WPSO, G3PCX, CMA-ES, G-CMA-ES AND THE 11 PAP INSTANTIATIONS ON f_1 TO $f_9 (D = 30)$

Function	f_1			f_2			f_3		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8.59e−13	4.69e−10
PCX	**0**	**0**	**0**	1.30e−02	7.94e−01	3.04e+00	**0**	**0**	**0**
ES	**0**	**0**	**0**	5.12e−11	1.05e−10	1.22e−10	**0**	**0**	**0**
G-CMA-ES	**0**	**0**	**0**	2.23e−11	3.24e−11	4.31e−11	**0**	**0**	**0**
DE+PSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PCX	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+PCX	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PCX+ES	**0**	**0**	**0**	2.93e−13	5.65e−11	2.08e−10	**0**	**0**	**0**
DE+PSO+PCX	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Function	f_4			f_5			f_6		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	1.14e−07	1.92e−06	1.34e−04	**0**	3.71e−12	3.99e+00	**0**	**0**	**0**
PSO	2.27e−09	1.43e−07	2.85e−06	1.11e−06	6.32e+00	1.65e+01	**0**	**0**	**0**
PCX	2.93e−12	1.31e−07	9.51e−03	**0**	**0**	3.99e+00	2.00e+00	7.00e+00	1.50e+01
ES	9.42e−12	1.28e−11	1.65e−11	**0**	**0**	**0**	**0**	**0**	**0**
G-CMA-ES	2.31e−12	3.54e−12	5.30e−12	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO	1.00e−07	1.99e−06	2.45e−05	1.90e−10	1.17e−09	1.42e−07	**0**	**0**	**0**
DE+PCX	8.92e−06	9.12e−04	3.04e−02	**0**	**0**	**0**	**0**	**0**	**0**
DE+ES	**0**	**0**	2.14e−13	**0**	3.73e−13	3.99e+00	**0**	**0**	**0**
PSO+PCX	4.83e−05	5.96e−02	2.05e+00	1.98e−05	3.85e−03	6.85e+01	**0**	**0**	**0**
PSO+ES	**0**	**0**	**0**	**0**	**0**	3.17e+00	**0**	**0**	**0**
PCX+ES	**0**	8.70e−13	5.71e−12	7.98e−08	8.55e−07	1.17e−02	**0**	**0**	**0**
DE+PSO+PCX	9.56e−07	3.05e−05	4.90e−04	1.00e−12	1.95e−10	3.99e+00	**0**	**0**	**0**
DE+PSO+ES	**0**	**0**	2.26e−13	2.97e−13	2.99e−11	5.46e−08	**0**	**0**	**0**
DE+PCX+ES	**0**	**0**	7.09e−13	**0**	**0**	5.27e−11	**0**	**0**	**0**
PSO+PCX+ES	**0**	**0**	1.36e−12	**0**	1.25e−02	5.26e−01	**0**	**0**	**0**
DE+PSO+PCX+ES	**0**	**0**	**0**	**0**	6.40e−11	7.49e−07	**0**	**0**	**0**
Function	f_7			f_8			f_9		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	1.47e−03	3.22e−03	7.44e−03	**3.82e−04**	**3.82e−04**	**3.82e−04**	**0**	**0**	**0**
PSO	9.99e−04	2.99e−03	5.78e−03	1.18e+03	2.01e+03	2.96e+03	2.29e+01	3.68e+01	5.97e+01
PCX	2.43e−01	1.20e+00	2.41e+00	6.12e+02	2.17e+03	3.93e+03	8.76e+01	1.11e+02	1.77e+02
ES	3.72e−02	6.13e−02	9.75e−02	1.24e+04	1.25e+04	1.25e+04	9.95e−01	1.99e+00	3.98e+00
G-CMA-ES	2.52e−02	6.40e−02	8.53e−02	8.61e+03	1.18e+04	1.25e+04	**0**	9.95e−01	2.98e+00
DE+PSO	8.78e−04	1.80e−03	3.41e−03	**3.82e−04**	**3.82e−04**	**3.82e−04**	**0**	**0**	**0**
DE+PCX	1.30e−03	2.48e−03	5.26e−03	**3.82e−04**	**3.82e−04**	**3.82e−04**	**0**	**0**	**0**
DE+ES	1.81e−03	3.97e−03	6.68e−03	**3.82e−04**	**3.82e−04**	**3.82e−04**	**0**	**0**	**0**
PSO+PCX	2.17e−03	4.70e−03	9.78e−03	2.27e+03	3.28e+03	6.16e+03	1.59e+01	5.07e+01	7.66e+01
PSO+ES	1.56e−03	3.25e−03	6.41e−03	1.24e+03	2.15e+03	3.51e+03	**0**	4.97e+00	1.19e+01
PCX+ES	1.63e−02	3.71e−02	9.64e−02	2.13e+03	4.50e+03	6.54e+03	1.99e+00	4.97e+00	7.96e+00
DE+PSO+PCX	**5.43e−04**	**1.53e−03**	**3.07e−03**	**3.82e−04**	**3.82e−04**	1.18e+02	**0**	**0**	**0**
DE+PSO+ES	7.15e−04	1.72e−03	4.98e−03	**3.82e−04**	**3.82e−04**	**3.82e−04**	**0**	**0**	**0**
DE+PCX+ES	1.14e−03	2.70e−03	4.15e−03	**3.82e−04**	**3.82e−04**	**3.82e−04**	**0**	**0**	**0**
PSO+PCX+ES	1.35e−03	3.73e−03	7.43e−03	1.42e+03	4.15e+03	5.53e+03	**0**	3.98e+00	8.95e+00
DE+PSO+PCX+ES	7.49e−04	1.84e−03	3.40e−03	**3.82e−04**	**3.82e−04**	1.18e+02	**0**	**0**	**0**

DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. All the results were obtained based on 30 independent runs with FEs = 3e + 05. The column headed "best," "median," and "worst" presents the best, median, and worst results among the 30 runs, respectively. For each column, the best result is highlighted in boldface.

TABLE V

FUNCTION ERROR VALUES OF THE SOLUTIONS OBTAINED BY SANSDE, WPSO, G3PCX, CMA-ES, G-CMA-ES AND THE 11 PAP INSTANTIATIONS ON f_{10} TO f_{13} AND f_{cec1} TO f_{cec5} $(D = 30)$

Function	f_{10}			f_{11}			f_{12}		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO	**0**	**0**	1.90e+00	**0**	1.23e−02	5.16e−02	**0**	**0**	1.04e−01
PCX	2.41e+00	5.76e+00	1.40e+01	**0**	**0**	**0**	**0**	3.12e+00	2.61e+01
ES	1.45e−11	1.82e−11	2.11e−11	**0**	**0**	**0**	**0**	**0**	**0**
G-CMA-ES	3.85e−12	5.50e−12	7.27e−12	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PCX	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+PCX	**0**	**0**	1.34e+00	**0**	7.40e−03	5.64e−02	**0**	4.15e−01	2.71e+00
PSO+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PCX+ES	**0**	**0**	2.07e−11	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO+PCX	**0**	**0**	**0**	**0**	**0**	7.40e−03	**0**	**0**	**0**
DE+PSO+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Function	f_{13}			f_{cec1}			f_{cec2}		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1.14e−13	2.27e−13
PSO	**0**	**0**	1.10e−02	**0**	**0**	2.27e−13	9.38e−12	1.48e−10	1.98e−06
PCX	**0**	6.58e+00	6.86e+01	1.14e−13	1.71e−13	1.71e−13	1.25e−12	2.22e−12	3.58e−12
ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
G-CMA-ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1.71e−13	3.41e−13
DE+PCX	**0**	**0**	**0**	**0**	**0**	**0**	1.14e−13	1.14e−13	2.27e−13
DE+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+PCX	**0**	**0**	1.24e+00	**0**	1.14e−13	1.14e−13	3.98e−13	7.96e−13	1.59e−12
PSO+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PCX+ES	**0**	**0**	1.10e−02	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO+PCX	**0**	**0**	**0**	**0**	**0**	**0**	1.14e−13	1.71e−13	2.27e−13
DE+PSO+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
PSO+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
DE+PSO+PCX+ES	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Function	f_{cec3}			f_{cec4}			f_{cec5}		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	1.45e+04	3.94e+04	1.73e+05	**7.76e−08**	**5.35e−06**	**4.26e−04**	**4.40e−04**	**7.01e−02**	**8.62e+00**
PSO	3.57e+05	1.06e+06	4.06e+06	1.24e+02	4.98e+02	1.96e+03	2.43e+03	4.08e+03	5.88e+03
PCX	6.92e+01	7.86e+02	5.35e+03	4.18e+04	8.14e+04	1.27e+05	2.90e+03	5.06e+03	6.54e+03
ES	**0**	**0**	**0**	8.48e+02	1.48e+04	3.80e+04	1.23e+00	2.39e+01	7.42e+01
G-CMA-ES	**0**	**0**	**0**	1.41e+02	9.01e+03	4.22e+04	7.28e−01	2.25e+01	1.13e+02
DE+PSO	2.46e+04	5.78e+04	1.52e+05	3.15e−06	1.82e−04	1.48e−03	6.41e−02	4.12e+00	4.33e+02
DE+PCX	1.20e+04	4.50e+04	1.24e+05	7.70e−07	9.68e−05	1.45e−01	3.11e−03	6.03e−01	5.18e+02
DE+ES	1.32e−07	2.85e−03	8.42e−01	9.37e−07	5.80e−05	8.93e−03	5.50e−02	1.33e+02	8.82e+02
PSO+PCX	2.04e+03	1.31e+04	7.69e+04	4.02e+02	2.57e+03	1.18e+04	4.30e+03	9.14e+03	1.61e+04
PSO+ES	**0**	**0**	**0**	1.01e+01	1.04e+03	2.46e+03	4.77e+01	2.66e+02	7.75e+02
PCX+ES	1.25e−05	2.87e−03	1.02e+00	1.13e+02	1.76e+04	1.31e+05	1.75e+00	2.36e+01	5.65e+02
DE+PSO+PCX	1.48e+04	7.05e+04	2.20e+05	1.08e−05	1.71e−03	6.77e−01	3.28e+00	1.89e+02	1.17e+03
DE+PSO+ES	2.53e−05	1.95e−03	9.50e−01	1.51e−05	4.92e−04	2.46e−01	6.64e−01	3.21e+02	1.50e+03
DE+PCX+ES	**0**	**0**	1.14e−13	2.18e−06	3.46e−04	5.43e−02	1.75e−02	3.84e+02	1.11e+03
PSO+PCX+ES	**0**	2.36e−04	9.11e−01	6.58e+02	2.77e+03	1.00e+04	3.06e+01	2.23e+02	7.75e+02
DE+PSO+PCX+ES	**0**	**0**	**0**	3.79e−06	2.00e−02	1.55e+00	5.01e+01	3.25e+02	9.11e+02

DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. All the results were obtained based on 30 independent runs with FEs = 3e + 05. The column headed "best," "median," and "worst" presents the best, median and worst results among the 30 runs, respectively. For each column, the best result is highlighted in boldface.

TABLE VI

FUNCTION ERROR VALUES OF THE SOLUTIONS OBTAINED BY SANSDE, WPSO, G3PCX, CMA-ES, G-CMA-ES AND THE 11 PAP INSTANTIATIONS ON f_{cec6} TO f_{cec14} ($D = 30$)

Function	f_{cec6}			f_{cec7}			f_{cec8}		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	**0**	5.37e−11	3.99e+00	**0**	9.86e−03	5.41e−02	2.08e+01	2.09e+01	2.10e+01
PSO	6.19e−03	1.05e+01	1.57e+02	3.41e−13	1.72e−02	4.67e−02	2.06e+01	2.09e+01	2.10e+01
PCX	1.60e−10	6.62e−07	8.08e+02	2.56e−13	4.83e−13	8.24e−13	**2.00e+01**	**2.00e+01**	**2.00e+01**
ES	**0**	**0**	**0**	**0**	**0**	**0**	**2.00e+01**	**2.00e+01**	2.10e+01
G-CMA-ES	**0**	**0**	**0**	**0**	**0**	7.40e−03	**2.00e+01**	**2.00e+01**	2.10e+01
DE+PSO	7.58e−10	3.50e−08	3.99e+00	**0**	9.86e−03	3.20e−02	**2.00e+01**	2.02e+01	2.06e+01
DE+PCX	**0**	1.14e−13	1.35e−10	**0**	9.86e−03	6.14e−02	2.06e+01	2.09e+01	2.10e+01
DE+ES	**0**	2.44e−09	3.99e+00	**0**	**0**	7.40e−03	**2.00e+01**	2.03e+01	2.10e+01
PSO+PCX	4.96e−04	1.51e−02	2.68e+02	2.27e−13	1.23e−02	4.42e−02	**2.00e+01**	**2.00e+01**	2.09e+01
PSO+ES	**0**	**0**	3.99e+00	**0**	**0**	7.40e−03	**2.00e+01**	2.08e+01	2.10e+01
PCX+ES	2.11e−07	1.68e−05	9.32e+01	**0**	**0**	7.40e−03	**2.00e+01**	**2.00e+01**	**2.00e+01**
DE+PSO+PCX	3.98e−13	1.44e−09	3.99e+00	**0**	1.72e−02	4.92e−02	**2.00e+01**	2.01e+01	2.09e+01
DE+PSO+ES	3.96e−10	3.26e−07	4.01e+00	**0**	**0**	**0**	**2.00e+01**	2.01e+01	2.10e+01
DE+PCX+ES	1.14e−13	2.16e−12	3.99e+00	**0**	**0**	7.40e−03	**2.00e+01**	2.06e+01	2.10e+01
PSO+PCX+ES	**0**	5.89e−01	8.96e+00	**0**	**0**	9.86e−03	**2.00e+01**	**2.00e+01**	2.10e+01
DE+PSO+PCX+ES	**0**	1.06e−07	3.99e+00	**0**	**0**	9.86e−03	**2.00e+01**	**2.00e+01**	2.03e+01
Function	f_{cec9}			f_{cec10}			f_{cec11}		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	**0**	**0**	**0**	3.18e+01	5.37e+01	6.40e+01	2.44e+01	2.72e+01	3.05e+01
PSO	1.89e+01	3.38e+01	5.67e+01	3.08e+01	5.77e+01	1.22e+02	1.53e+01	2.12e+01	2.96e+01
PCX	1.46e+02	2.15e+02	3.42e+02	1.57e+02	2.08e+02	3.63e+02	2.35e+01	2.64e+01	2.93e+01
ES	2.89e+01	5.87e+01	1.84e+02	1.99e+01	3.08e+01	4.18e+01	1.76e+00	6.27e+00	**1.02e+01**
G-CMA-ES	**0**	9.95e−01	6.96e+00	**3.11e−07**	**9.95e−01**	**3.98e+00**	**3.28e−08**	**2.39e−07**	1.75e+01
DE+PSO	**0**	**0**	**0**	3.04e+01	4.28e+01	6.50e+01	1.70e+01	2.24e+01	2.78e+01
DE+PCX	**0**	**0**	**0**	3.06e+01	4.25e+01	6.56e+01	2.32e+01	2.65e+01	2.95e+01
DE+ES	**0**	**0**	**0**	2.69e+01	4.38e+01	6.18e+01	2.47e+00	8.33e+00	2.95e+01
PSO+PCX	3.98e+01	7.06e+01	1.31e+02	5.67e+01	2.70e+02	4.26e+02	1.69e+01	2.42e+01	3.63e+01
PSO+ES	1.89e+01	9.35e+01	1.55e+02	1.79e+01	4.28e+01	6.67e+01	3.80e+00	7.46e+00	2.53e+01
PCX+ES	3.18e+01	1.89e+02	2.26e+02	3.18e+01	4.48e+01	7.16e+01	2.46e+00	7.42e+00	1.97e+01
DE+PSO+PCX	**0**	**0**	9.95e−01	2.86e+01	4.58e+01	7.86e+01	1.25e+01	2.42e+01	2.95e+01
DE+PSO+ES	**0**	**0**	**0**	2.57e+01	4.48e+01	6.17e+01	2.92e+00	7.35e+00	2.37e+01
DE+PCX+ES	**0**	**0**	**0**	3.07e+01	3.98e+01	6.08e+01	2.61e+00	8.11e+00	2.85e+01
PSO+PCX+ES	5.57e+01	8.95e+01	1.38e+02	3.18e+01	4.58e+01	6.57e+01	3.14e+00	7.70e+00	3.23e+01
DE+PSO+PCX+ES	**0**	**0**	**0**	1.89e+01	4.18e+01	5.77e+01	2.97e+00	8.64e+00	2.91e+01
Function	f_{cec12}			f_{cec13}			f_{cec14}		
Algorithm	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DE	7.83e+03	1.47e+04	2.46e+04	1.87e+00	2.15e+00	2.33e+00	1.23e+01	1.28e+01	1.33e+01
PSO	2.26e+03	3.09e+04	8.82e+04	1.77e+00	3.06e+00	4.99e+00	**1.03e+01**	**1.21e+01**	1.30e+01
PCX	3.10e−11	6.01e+02	1.14e+04	9.07e+00	1.68e+01	2.85e+01	1.32e+01	1.38e+01	1.45e+01
ES	**0**	7.87e+01	2.17e+03	8.40e−01	2.08e+00	2.75e+00	1.46e+01	1.46e+01	1.46e+01
G-CMA-ES	**0**	3.33e+02	1.10e+04	1.02e+00	2.03e+00	2.94e+00	1.46e+01	1.46e+01	1.46e+01
DE+PSO	8.13e+01	6.71e+03	3.46e+04	1.14e+00	1.63e+00	2.36e+00	1.11e+01	1.24e+01	**1.27e+01**
DE+PCX	6.55e+03	1.42e+04	2.50e+04	1.44e+00	1.77e+00	1.97e+00	1.20e+01	1.27e+01	1.31e+01
DE+ES	**0**	1.19e+03	1.51e+04	1.08e+00	1.78e+00	2.22e+00	1.19e+01	1.27e+01	1.33e+01
PSO+PCX	6.22e−11	5.18e+02	9.03e+03	3.53e+00	7.28e+00	1.52e+01	1.21e+01	1.30e+01	1.40e+01
PSO+ES	**0**	4.35e+02	1.08e+04	1.27e+00	2.22e+00	3.91e+00	1.10e+01	1.23e+01	1.35e+01
PCX+ES	**0**	**3.94e+01**	**1.89e+03**	4.55e−01	2.41e+00	4.31e+00	1.34e+01	1.40e+01	1.45e+01
DE+PSO+PCX	2.96e+02	6.45e+03	2.00e+04	1.07e+00	**1.33e+00**	1.83e+00	1.14e+01	1.23e+01	1.28e+01
DE+PSO+ES	**0**	1.15e+03	1.14e+04	6.85e−01	1.50e+00	2.26e+00	1.13e+01	1.23e+01	1.30e+01
DE+PCX+ES	**0**	1.74e+03	1.08e+04	7.76e−01	1.57e+00	2.04e+00	1.21e+01	1.27e+01	1.33e+01
PSO+PCX+ES	**0**	3.57e+02	1.01e+04	1.09e+00	2.49e+00	3.64e+00	1.19e+01	1.26e+01	1.35e+01
DE+PSO+PCX+ES	**0**	4.77e+02	1.08e+04	**2.91e−01**	1.37e+00	**1.82e+00**	1.19e+01	1.24e+01	1.31e+01

DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. All the results were obtained based on 30 independent runs with FEs = 3e + 05. The column headed "best," "median," and "worst" presents the best, median, and worst results among the 30 runs, respectively. For each column, the best result is highlighted in boldface.

TABLE VII
COMPARISON BETWEEN PAP INSTANTIATIONS AND THEIR CONSTITUENT ALGORITHMS AND G-CMA-ES ON THE 27 TEST FUNCTIONS (TWO-SIDED WILCOXON RANK-SUM TEST WITH SIGNIFICANCE LEVEL 0.05 WAS USED)

Value-to-Reach	PAP	DE	PSO	PCX	ES	G-CMA-ES
1.0e−13	DE+PSO	7–15–5	19–7–1	–	–	10–7–10
	DE+PCX	6–18–3	–	18–4–5	–	10–8–9
	DE+ES	10–14–3	–	–	10–9–8	10–9–8
	PSO+PCX	–	7–7–13	13–5–9	–	6–4–17
	PSO+ES	–	17–8–2	–	7–12–8	7–13–7
	PCX+ES	–	–	20–5–2	5–14–8	5–13–9
	DE+PSO+PCX	7–14–6	18–7–2	18–3–6	–	9–7–11
	DE+PSO+ES	11–12–4	21–5–1	–	10–9–8	10–9–8
	DE+PCX+ES	11–14–2	–	21–5–1	10–10–7	10–10–7
	PSO+PCX+ES	–	17–6–4	19–5–3	7–12–8	7–11–9
	DE+PSO+PCX+ES	11–13–3	21–5–1	20–6–1	10–11–6	10–11–6
1.0e−06	DE+PSO	7–17–3	16–10–1	–	–	8–11–8
	DE+PCX	4–20–3	–	16–6–5	–	8–12–7
	DE+ES	8–16–3	–	–	7–13–7	7–13–7
	PSO+PCX	–	5–9–13	11–7–9	–	4–8–15
	PSO+ES	–	14–11–2	–	4–16–7	4–17–6
	PCX+ES	–	–	17–9–1	3–16–8	4–15–8
	DE+PSO+PCX	7–16–4	15–10–2	17–5–5	–	7–11–9
	DE+PSO+ES	10–14–3	18–8–1	–	7–13–7	7–13–7
	DE+PCX+ES	9–16–2	–	19–7–1	7–15–5	7–15–5
	PSO+PCX+ES	–	14–9–4	16–8–3	4–15–8	4–14–9
	DE+PSO+PCX+ES	10–14–3	18–8–1	18–8–1	7–15–5	7–15–5
1.0e−02	DE+PSO	6–19–2	15–12–0	–	–	8–13–6
	DE+PCX	3–22–2	–	16–7–4	–	8–13–6
	DE+ES	7–19–1	–	–	7–13–7	7–13–7
	PSO+PCX	–	5–10–12	11–7–9	–	4–8–15
	PSO+ES	–	13–12–2	–	4–16–7	4–17–6
	PCX+ES	–	–	16–10–1	3–17–7	4–16–7
	DE+PSO+PCX	6–18–3	14–12–1	16–7–4	–	7–13–7
	DE+PSO+ES	8–18–1	16–10–1	–	7–13–7	7–13–7
	DE+PCX+ES	7–19–1	–	18–8–1	7–15–5	7–15–5
	PSO+PCX+ES	–	13–10–4	15–9–3	4–15–8	4–14–9
	DE+PSO+PCX+ES	8–17–2	16–10–1	18–8–1	7–15–5	7–15–5

DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. Results are presented in form of win–draw–lose, standing for the numbers of functions on which the corresponding PAP instantiation is superior, comparable (i.e., statistically insignificant), and inferior to the compared constituent algorithm.

with less risk than the compared algorithm. For example, when comparing the PAP instantiation that combines SaNSDE with wPSO to SaNSDE, the statistic "0.30–0.24" indicates that PAP outperformed SaNSDE with a probability 0.30, while was outperformed by SaNSDE with a probability 0.24. From Table VIII, similar patterns can be observed in case of all three precision levels. First, most PAP instantiations were associated with less risks in comparison with their constituent algorithms. It was not surprising to discover negative results on the combinations between G3PCX, wPSO, and CMA-ES, since CMA-ES is superior or comparable to the other two algorithms on almost all functions. Second, many PAP instantiations, such as the combination of SaNSDE and CMA-ES and the combination of all four basic algorithms, were associated with less risks than G-CMA-ES. Hence, the efficacy of the PAP framework with respect to risk reduction has been experimentally shown.

Although the results presented in Tables VII and VIII summarize the performance of PAP over the whole set of benchmark functions, these might not be sufficient to draw a complete picture of the performance of PAP. Taking a closer look at the evolutionary process of PAP on individual functions will give more insight into PAP's behaviors. Hence, we further carried out case studies on three selected functions, including f_9 (generalized Rastrigin's function), f_2 (Schwefel's problem 2.22), and f_{cec12} (Schwefel's problem 2.13). Two PAP instantiations were considered in the case studies. First, we considered a PAP with only two constituent algorithms. This could make it easier to observe the relationship between the behaviors of PAP and its constituent algorithms. Since SaNSDE and CMA-ES generally performed better than wPSO and G3PCX, the PAP with SaNSDE and CMA-ES was a good choice for our study. The PAP with four constituent algorithms was also considered, because it incorporates the most constituent algorithms and outperformed all of them as well as G-CMA-ES, as shown in both Tables VII and VIII.

Figs. 2–4 present the evolutionary curves of the PAP with CMA-ES and SaNSDE on the three functions. For each algorithm, we sorted the 30 runs by the quality of the final solutions and picked out the median ones. The corresponding

TABLE VIII
COMPARISON BETWEEN PAP INSTANTIATIONS AND THEIR CONSTITUENT ALGORITHMS AND G-CMA-ES IN TERMS OF RISK (USING THE METRIC PROPOSED IN SECTION III)

Value-to-Reach	PAP	DE	PSO	PCX	ES	G-CMA-ES
	DE+PSO	0.30–0.24	0.63–0.11	–	–	0.35–0.37
	DE+PCX	0.31–0.22	–	0.65–0.20	–	0.35–0.33
	DE+ES	0.37–0.17	–	–	0.40–0.26	0.39–0.27
	PSO+PCX	–	0.32–0.47	0.55–0.35	–	0.23–0.59
	PSO+ES	–	0.58–0.16	–	0.33–0.23	0.30–0.26
1.0e−13	PCX+ES	–	–	0.74–0.14	0.30–0.36	0.26–0.39
	DE+PSO+PCX	0.29–0.26	0.62–0.12	0.66–0.22	–	0.33–0.40
	DE+PSO+ES	0.38–0.16	0.69–0.05	–	0.40–0.26	0.39–0.27
	DE+PCX+ES	0.40–0.14	–	0.79–0.07	0.39–0.22	0.38–0.22
	PSO+PCX+ES	–	0.55–0.19	0.75–0.13	0.33–0.32	0.30–0.35
	DE+PSO+PCX+ES	0.40–0.14	0.70–0.05	0.78–0.08	0.41–0.21	0.39–0.22
	DE+PSO	0.28–0.16	0.56–0.10	–	–	0.27–0.25
	DE+PCX	0.23–0.21	–	0.56–0.20	–	0.27–0.27
	DE+ES	0.29–0.14	–	–	0.29–0.20	0.27–0.21
	PSO+PCX	–	0.26–0.45	0.47–0.35	–	0.16–0.52
	PSO+ES	–	0.48–0.16	–	0.21–0.23	0.19–0.26
1.0e−06	PCX+ES	–	–	0.61–0.12	0.20–0.32	0.18–0.33
	DE+PSO+PCX	0.26–0.19	0.54–0.12	0.58–0.18	–	0.25–0.29
	DE+PSO+ES	0.32–0.11	0.59–0.05	–	0.29–0.20	0.27–0.21
	DE+PCX+ES	0.31–0.12	–	0.64–0.06	0.28–0.17	0.26–0.17
	PSO+PCX+ES	–	0.45–0.19	0.62–0.13	0.22–0.32	0.18–0.35
	DE+PSO+PCX+ES	0.33–0.11	0.59–0.05	0.65–0.06	0.29–0.16	0.27–0.18
	DE+PSO	0.23–0.10	0.51–0.07	–	–	0.26–0.22
	DE+PCX	0.20–0.14	–	0.55–0.15	–	0.26–0.22
	DE+ES	0.24–0.08	–	–	0.29–0.17	0.26–0.18
	PSO+PCX	–	0.24–0.41	0.47–0.30	–	0.16–0.46
	PSO+ES	–	0.43–0.13	–	0.21–0.23	0.18–0.26
1.0e−02	PCX+ES	–	–	0.59–0.10	0.20–0.26	0.18–0.28
	DE+PSO+PCX	0.22–0.13	0.50–0.08	0.56–0.14	–	0.24–0.25
	DE+PSO+ES	0.26–0.06	0.53–0.04	–	0.29–0.16	0.26–0.18
	DE+PCX+ES	0.24–0.08	–	0.62–0.06	0.28–0.16	0.25–0.17
	PSO+PCX+ES	–	0.41–0.17	0.60–0.11	0.22–0.29	0.18–0.32
	DE+PSO+PCX+ES	0.27–0.08	0.53–0.04	0.63–0.05	0.29–0.15	0.26–0.17

DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. The two numbers in each cell stand for the probabilities that the PAP and the constituent algorithm outperformed each other.

evolutionary curve was then plotted with respect to the value-to-reach 1e−13 (since semi-log graphs were plotted here, all values smaller than this number were set to 1e−13 rather than 0 before being plotted). Three different scenarios can be observed from the figures. First, CMA-ES converged rapidly on f_9, but the solution obtained was not good. On the other hand, SaNSDE progressed much slower than CMA-ES, but continuously found better solutions and eventually arrived at the value-to-reach. By combining the two basic algorithms, the PAP progressed very fast at the beginning, which probably should be credited to CMA-ES. After that, it stagnated for a while. Once SaNSDE obtained a sufficiently good solution, the PAP started progressing again, and finally reached the value-to-reach as well. Note that the PAP always evolved faster than SaNSDE during the search. This illustrated that the PAP managed to take advantage of both CMA-ES and SaNSDE, and thus accelerated the evolutionary process. The scenario showed by Fig. 3 is similar to that of Fig. 2. In this case, CMA-ES converged faster than SaNSDE on f_2, while SaNSDE obtained a better solution. PAP always obtained better or comparable solutions in comparison with SaNSDE throughout the evolutionary process. Finally, Fig. 4 provides a negative case in which PAP did not achieve the best performance. On f_{cec12}, CMA-ES not only converged faster than SaNSDE, but also consistently obtained better solutions. Hence, there is hardly any advantage that the PAP could take from SaNSDE, and assigning computation resources to it was probably a waste. Moreover, we found that CMA-ES evolved very fast at the early stage and then stagnated for a long time. At the late stage of evolution, the solution quality was improved with a sudden jump. Such an improvement was achieved after restarting CMA-ES for many times. Since the PAP assigned a lot of computational resource to SaNSDE, it could not afford so many restarts for CMA-ES. In consequence, the PAP was inferior to CMA-ES both in terms of solution quality and convergence speed in this case. Nevertheless, the PAP always outperformed SaNSDE. Thus, employing PAP at least alleviated the risk of selecting the wrong algorithm.

Figs. 5–7 present the evolutionary process of the PAP with 4 constituent algorithms. The three figures essentially tell the

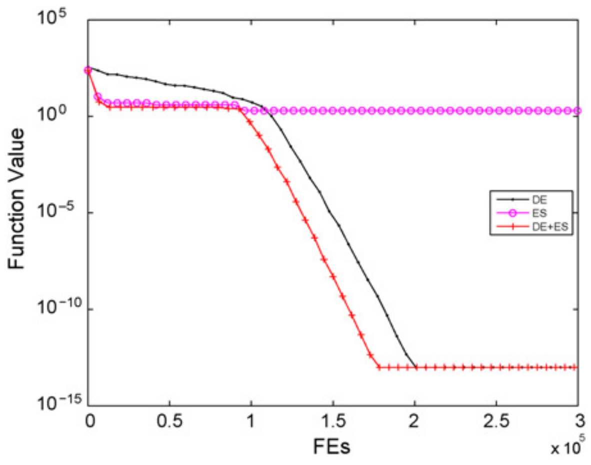


Fig. 2. Evolutionary process of DE+ES, DE, and ES on function f_9. DE and ES stand for SaNSDE and CMA-ES, respectively. The value-to-reach was set to 1e−13.

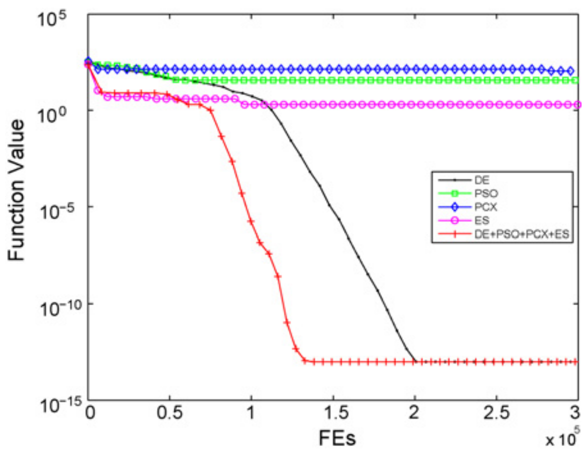


Fig. 5. Evolutionary process of DE+PSO+PCX+ES and its constituent algorithms on function f_9. DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. The value-to-reach was set to 1e−13.

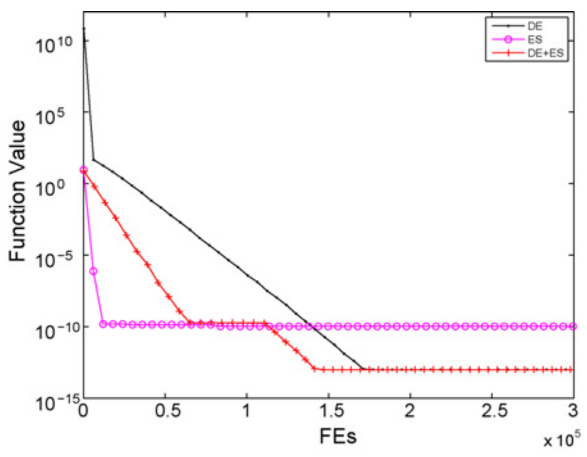


Fig. 3. Evolutionary process of DE+ES, DE, and ES on function f_2. DE and ES stand for SaNSDE and CMA-ES, respectively. The value-to-reach was set to 1e−13.

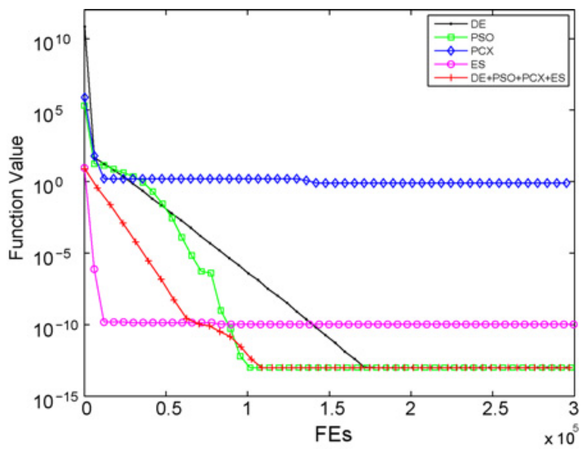


Fig. 6. Evolutionary process of DE+PSO+PCX+ES and its constituent algorithms on function f_2. DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. The value-to-reach was set to 1e−13.

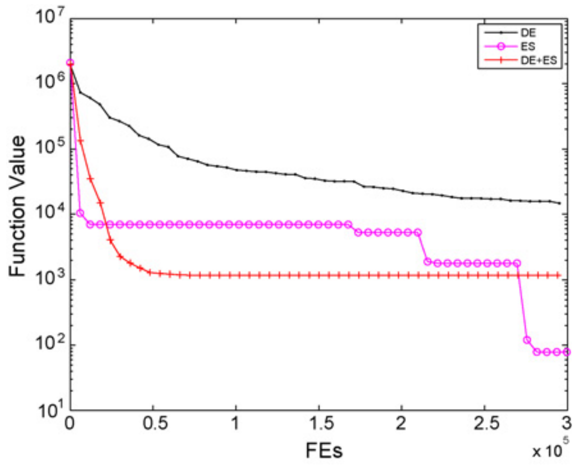


Fig. 4. Evolutionary process of DE+ES, DE, and ES on function f_{cec12}. DE and ES stand for SaNSDE and CMA-ES, respectively. The value-to-reach was set to 1e−13.

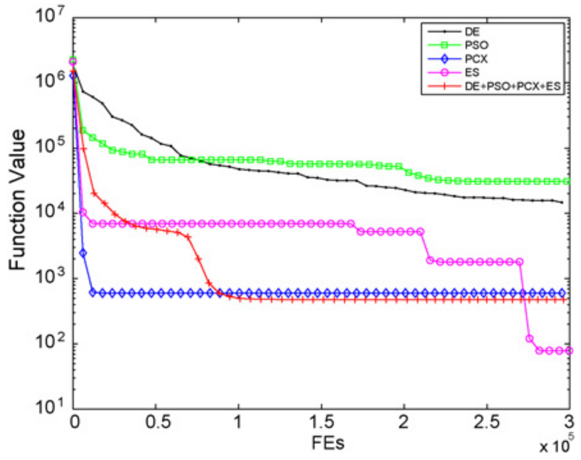


Fig. 7. Evolutionary process of DE+PSO+PCX+ES and its constituent algorithms on function f_{cec12}. DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. The value-to-reach was set to 1e−13.

TABLE IX
NUMBER OF FUNCTIONS ON WHICH 16 DIFFERENT SETTINGS OF THE MIGRATION PARAMETERS DID NOT RESULT IN STATISTICALLY DIFFERENT PERFORMANCE (IN TERMS OF SOLUTION QUALITY) OF THE PAP INSTANTIATIONS

Algorithm	No. of Functions on Which no Statistically Different Results Were Obtained
DE+PSO	19
DE+PCX	25
DE+ES	22
PSO+PCX	17
PSO+ES	25
PCX+ES	26
DE+PSO+PCX	16
DE+PSO+ES	18
DE+PCX+ES	21
PSO+PCX+ES	20
DE+PSO+PCX+ES	20

Statistical significance test was conducted using the Kruskal–Wallis test with significance level 0.05. DE, PSO, PCX, and ES stand for SaNSDE, wPSO, G3PCX, and CMA-ES, respectively.

same story as Figs. 2–4, and hence verified previous analyses on PAP with more than two constituent algorithms.

Two conclusions regarding the migration scheme of PAP can be drawn from the superiority of PAP over the compared algorithms. First, the migration scheme is of great importance to the success of PAP. Second, 1 and $MAX\_GEN/20$ are two appropriate and robust values for the parameters $migration\_size$ and $migration\_interval$. To further investigate the influence of these parameters on PAP, we carried out a sensitivity analysis to check whether the performance of PAP will change significantly with other parameter settings. As stated in Section VI-B, 16 different pairs of $migration\_interval$ and $migration\_size$ were tested for every instantiation of PAP. For each pair, 30 independent runs were executed on all 27 benchmark functions. Then, for each instantiation on each function, Kruskal–Wallis one-way analysis of variance by ranks was employed to test whether the 16 pairs of parameters had led to significantly different performance. After that, for each instantiation of PAP, we counted the number of the benchmark functions on which all 16 pairs of parameters made no difference. The larger the number, the more insensitive an instantiation is to the parameters. For the sake of brevity, we only summarize in Table IX these numbers for the 11 PAP instantiations, while omit the full details. It can be observed that, in the worst case (SaNSDE+wPSO+G3PCX), the PAP instantiation is insensitive to the migration parameters on 16 out of 27 functions.

V. Further Analysis

The previous section empirically verified the efficacy of PAP. However, the reason why PAP performed well is still not fully understood. In Section II-C, we suggested that the constituent algorithms of PAP should be carefully chosen so that they are complementary. Is this really the key issue? Can we achieve comparable performance by simply running the same algorithm in parallel?

A. *Compare PAP With Parallel EAs*

To answer the above question, we carried out an additional experiment to compare PAP with parallel EAs. The PAP with four constituent algorithms was chosen as the representative PAP instantiation in this experiment. The comparison was made between this PAP instantiation and the parallel version of all its four constituent algorithms, i.e., parallel SaNSDE (PDE), parallel wPSO (PPSO), parallel G3PCX (PPCX) and parallel CMA-ES (PES). All four parallel EAs were run on the same benchmark functions described in Section IV with 300 000 FEs. Each parallel EAs maintained four subpopulations. For PDE, PPSO, PPCX, and PES, each subpopulation consisted of 25, 10, 26, and 14 individuals, respectively. According to previous experimental results, $migration\_size$ and $migration\_interval$ were set to 1 and $MAX\_GEN/20$ throughout the experiment.

Table X summarizes the comparison between PAP and the four parallel EAs on the 27 functions. The comparison was made based on 30 independent runs using two-sided Wilcoxon rank-sum tests and the risk metric. It can be observed that PAP outperformed all the compared parallel EAs. Hence, employing different constituent algorithms in PAP indeed boosted its performance.

B. *On Choosing Constituent Algorithms for PAP*

The previous section further demonstrated that the advantage of PAP was not due to parallelization, but to the use of different constituent algorithms. However, we also observed from Section IV that not all 11 PAP instantiations performed comparably. This difference in performance apparently lies in the different constituent algorithms they employed. Now the question is which type of algorithms we should use to implement a PAP. In Section II-C, we suggested that the constituent algorithms should be complementary to each another. The following analysis attempts to elaborate more on this issue.

We start from the case of two constituent algorithms, say A_1 and A_2. Our question is, how we can properly choose A_1 and A_2 so that the PAP is associated with less risk in comparison to another algorithm A^*. Here, A^* can be any algorithm, including A_1 and A_2.[2] Since the selection of A_1 and A_2 is irrelevant to A^*, seeking a PAP that is less risky than A^* can be carried out by minimizing the probability that the PAP obtains a worse solution than A^* over a problem set. We further assume that the PAP runs A_1 and A_2 independently (i.e., the migration scheme is omitted), and let $P_{1,k}$ and $P_{2,k}$ be the probabilities that A_1 and A_2 obtain a better solution than A^*. According to (5) and (6), the minimization problem takes the following form:

$$\min\ R = \frac{1}{n}\sum_{k=1}^{n}(1 - P_{1,k})(1 - P_{2,k}) \tag{8}$$

[2]If we cast all the computation time on A_1 (or A_2), in most cases, the solution obtained will be better than the solution obtained by assigning only a part of time budget to it (as we do in PAP). Hence, the A_1 (A_2) in the two scenarios can be regarded as two algorithms.

TABLE X

COMPARISON BETWEEN THE PAP INSTANTIATION WITH FOUR CONSTITUENT ALGORITHMS AND PARALLEL EAS

		PDE	PPSO	PPCX	PES	
DE+PSO+PCX+ES	Wilcoxon test results	7–15–5	23–4–0	24–3–0	10–10–7	
$(ms = 1, mi = MAX\_GEN/20)$	Risk metric results	0.34–0.21	0.72–0.03	0.87–0.01	0.39–0.23	

PDE, PPSO, PPCX, and PES stand for parallel versions of SaNSDE, wPSO, G3PCX, and CMA-ES, respectively. The two-sided Wilcoxon rank-sum test (with significance level 0.05) was conducted based on solution quality and the results are presented in form of win–draw–lose. The statistics presented in the third row were calculated using the proposed risk metric. All comparisons were based on the value-to-reach of 1e−13 for the reason explained in Section IV-C.

where k is the index of problems. With some simple derivation, we can find that

$$\begin{aligned}
&\frac{1}{n}\sum_{k=1}^{n}(1-P_{1,k})(1-P_{2,k}) \\
&\quad= 1+\frac{1}{n}\sum_{k=1}^{n}P_{1,k}P_{2,k}-\bar{P}_1-\bar{P}_2 \\
&\quad= 1-\bar{P}_1-\bar{P}_2+\frac{1}{n}\sum_{k=1}^{n}(P_{1,k}-\bar{P}_1)(P_{2,k}-\bar{P}_2) \\
&\quad+\frac{1}{n}\bar{P}_1\sum_{k=1}^{n}P_{2,k}+\frac{1}{n}\bar{P}_2\sum_{k=1}^{n}P_{1,k}-\bar{P}_1\bar{P}_2 \\
&\quad= 1-\bar{P}_1-\bar{P}_2+\bar{P}_1\bar{P}_2+\frac{1}{n}\sum_{k=1}^{n}(P_{1,k}-\bar{P}_1)(P_{2,k}-\bar{P}_2) \\
&\quad= (1-\bar{P}_1)(1-\bar{P}_2)+\frac{1}{n}\sum_{k=1}^{n}(P_{1,k}-\bar{P}_1)(P_{2,k}-\bar{P}_2) \qquad (9)
\end{aligned}$$

where $\bar{P}_1=\frac{1}{n}\sum_{k=1}^{n}P_{1,k}$ and $\bar{P}_2=\frac{1}{n}\sum_{k=1}^{n}P_{2,k}$.

Taking a closer look at the above derivations, we can find that larger $\bar{P}_1$ and $\bar{P}_2$ generally lead to a smaller R, which means A_1 and A_2 should perform sufficiently well on the problem set. Meanwhile, the term $\sum_{k=1}^{n}(P_{1,k}-\bar{P}_1)(P_{2,k}-\bar{P}_2)$ needs to be as small as possible. This observation implies that we hope $(P_{1,k}-\bar{P}_1)$ to be negative if $(P_{2,k}-\bar{P}_2)$ is positive and vice versa. More precisely, the performance of A_1 on a problem is desired to be above its "average" performance over the problem set when the performance of A_2 is below its "average" performance and vice versa. This elaborates the term "complementary" that has been briefly mentioned in Section II-C.

The above analysis can be generalized to more than two constituent algorithms by considering the algorithm selection process in a sequential manner. For the m-algorithms case, one first identifies two algorithms A_1 and A_2 as a basis, followed by seeking the remaining algorithms one by one. Suppose we have decided i algorithms and now need to identify an additional algorithm, say A_{i+1}. We may regard the combination of all the previously selected algorithms as a single algorithm A_c, a potentially good A_{i+1} should then not only be good by itself, but also be complementary to A_c. Viewing the m-algorithms in a sequential manner also helps us understanding the limit of PAP. Specifically, when there are only two constituent algorithms, one can easily identify two complementary algorithms that both generally perform well. As the number of algorithms increases, the performance of PAP is expected to improve as long as new good constituent algorithms can be included. However, such improvement makes it more and more difficult to identify the next satisfactory constituent algorithm. Therefore, the performance of PAP will eventually start decreasing because of the inclusion of some undesirable constituent algorithms. In other words, there must be some kind of "upper bound" of PAP's performance improvement. A rigorous analysis along this line is nontrivial, and deserves in-depth investigation in the future. Moreover, we also need to analyze the effect of migration in the future.

Although our theoretical analysis has been simplified, it actually explains the experimental results quite well. Take SaNSDE and CMA-ES as examples, both exhibit better performance than G3PCX and wPSO. According to the Wilcoxon test, we found that SaNSDE achieved significantly better solutions than CMA-ES on f_7, f_8, f_9, f_{cec4}, f_{cec5}, f_{cec9}, and f_{cec14}, while CMA-ES performed much better than SaNSDE on f_{cec3}, f_{cec7}, f_{cec8}, f_{cec10}, f_{cec11}, f_{cec12}. In other words, the two algorithms favor quite different functions. When employing them as the constituent algorithms, the resulting PAP not only beat both SaNSDE and CMA-ES themselves, but also outperformed G-CMA-ES on 10 out of 27 functions. In contrast, the PAP instantiation using wPSO and G3PCX appeared to be an obvious negative example. G3PCX and wPSO concurrently failed on many functions, e.g., f_9, f_{11}, f_{cec1}, f_{cec2}, f_{cec5}, f_{cec6}, f_{cec10}, and f_{cec13}. Consequently, the PAP instantiation employing these two algorithms was not even as good as wPSO. The above two examples demonstrate that our theoretical analysis, at least to some extent, can be used to explain experimental results.

C. Can PAP Increase the Probability of Finding the Global Optimum?

So far, all the experimental studies compared the quality of the solutions obtained by an algorithm. The attractive performance showed by PAP essentially indicated that PAP managed to get good solutions within a given time budget. However, a good solution might be a local optimum. Hence, the evidence shown in previous sections did not necessarily show that PAP is capable of improving the probability of finding the global optimum. Investigating PAP from this perspective will provide us with a more comprehensive understanding of its characteristics. In practice, we may measure the probability of finding a solution that reaches the precision threshold of the computer hardware. In the context of this paper, this was done by calculating the rates of an algorithm reaching the value-to-reach 1e−13 in 30 independent runs.

Eight algorithms, including three PAP instantiations, the four basic algorithms and G-CMA-ES were used in this investigation. The PAP instantiation employing SaNSDE, wPSO, and CMA-ES as its constituent algorithms was chosen as

TABLE XI

COMPARISON AMONG THREE PAP INSTANTIATIONS, THE FOUR BASIC ALGORITHMS AND G-CMA-ES IN TERMS OF PROBABILITY OF REACHING THE VALUE-TO-REACH OF 1E-13

	DE	PSO	PCX	ES	G-CMA-ES	DE+ES	DE+PSO+ES	DE+PSO+PCX+ES
f_1	1	1	1	1	1	1	1	1
f_2	1	1	0	0	0	1	1	1
f_3	1	0.37	1	1	1	1	1	1
f_4	0	0	0	0	0	0.90	0.97	1
f_5	0.03	0	0.73	1	1	0.17	1	0.33
f_6	1	1	0	1	1	1	1	1
f_7	0	0	0	0	0	0	0	0
f_8	0	0	0	0	0	0	0	0
f_9	1	0	0	0	0.10	1	1	1
f_{10}	1	0.97	0	0	0	1	1	1
f_{11}	1	0.33	1	1	1	1	1	1
f_{12}	1	0.80	0.27	1	1	1	1	1
f_{13}	1	0.93	0.03	1	1	1	1	1
f_{cec1}	1	0.60	0	1	1	1	1	1
f_{cec2}	0.10	0	0	1	1	1	1	1
f_{cec3}	0	0	0	1	1	0	0	1
f_{cec4}	0	0	0	0	0	0	0	0
f_{cec5}	0	0	0	0	0	0	0	0
f_{cec6}	0.03	0	0	1	1	0.03	0	0.03
f_{cec7}	0.27	0	0	1	0.97	0.97	1	0.93
f_{cec8}	0	0	0	0	0	0	0	0
f_{cec9}	1	0	0	0	0.03	1	1	1
f_{cec10}	0	0	0	0	0	0	0	0
f_{cec11}	0	0	0	0	0	0	0	0
f_{cec12}	0	0	0	0.10	0.20	0.07	0.07	0.07
f_{cec13}	0	0	0	0	0	0	0	0
f_{cec14}	0	0	0	0	0	0	0	0
Average	0.42	0.26	0.15	0.45	0.46	0.52	0.52	0.57

DE, PSO, PCX, and ES Stand for SaNSDE, wPSO, G3PCX, and CMA-ES, Respectively.

the representative PAP with three constituent algorithms. The PAP instantiation with SaNSDE and CMA-ES was chosen as the representative PAP with two constituent algorithms. In addition, we also considered the PAP instantiation with four constituent algorithms.

Table XI presents the rates of the algorithms attaining the value-to-reach in 30 independent runs. These rates were calculated based on results of the experiments described in Section IV. It can be observed that the PAP instantiation with four constituent algorithms was overall the best among the compared algorithms. It achieved the highest average rate of 0.57. The average rates of the other two PAP instantiations were both 0.52, while the average rates of SaNSDE, wPSO, G3PCX, CMA-ES, and G-CMA-ES were 0.42, 0.26, 0.15, 0.45, and 0.46. In comparison with their constituent algorithms, the PAP instantiations achieved higher or equal rates on 23, 24, and 23 functions, respectively. Therefore, the efficacy of PAP was again demonstrated.

VI. CONCLUSION

This paper investigated solving numerical optimization problems for which solutions must be presented within a limited time budget. Although numerous algorithms are readily applicable for this type of problems, their performance usually varies significantly from problem to problem. This implies that there is an inherent risk associated with the selection of an algorithm. Unfortunately, identifying a suitable (or optimal) algorithm for a specific problem is a nontrivial task due to the lack of prior knowledge. The limited time budget also prohibits us from trying out different algorithms and then choosing the best one. Instead of betting the entire time budget on a single algorithm, we proposed that such a risk can be reduced by distributing the time budget to multiple algorithms. Based on this idea, a general framework called PAP has been proposed in the context of population-based search algorithms. PAP typically consists of a number of constituent algorithms, each of which is allowed to run with a portion of the time budget. Allocation of computation time is implemented by dividing the whole population into a number of subpopulations, and maintaining one for each constituent algorithm. To further boost the performance, interaction among constituent algorithms is carried out through regularly migrating individuals among the subpopulations. We proposed a pairwise metric to compare the risks associated with two algorithms. Such a metric can be used to evaluate how effective our PAP is, together with other common metrics. Given a set of functions, the proposed metric essentially measures how likely it is that an algorithm will find a better solution than another algorithm by the end of a given time budget.

To evaluate the effectiveness of PAP, 11 instantiations of PAP were implemented based on four existing constituent algorithms, including SaNSDE, wPSO, G3PCX, and CMA-ES. The performance of each instantiation was compared to its constituent algorithms on 27 benchmark functions. Our experimental results showed that seven out of the 11 PAP instantiations outperformed their constituent algorithms in terms of solution quality and the proposed risk metric. Furthermore, 7 out of the 11 instantiations even achieved superior or comparable performance in comparison with G-CMA-ES, which was known to be superior to any of the four constituent algorithms. Our empirical studies also revealed that PAP is capable of increasing the probability of finding the global optimum and is insensitive to control parameters of the migration scheme. Further analyses have been conducted to investigate in what circumstance PAP may outperform its constituent algorithms. Complementarity was identified as a key issue.

Though PAP has been shown to be a promising framework, the resource (time) allocation strategy utilized in this paper was somewhat *ad hoc*: We manually allocated the resource before running the PAP. When the optimization task becomes tougher, either in terms of the inherent difficulty of the problem or in terms of an extremely limited time budget, fixed allocation of computational resources might be too rigid to guarantee good PAP performance. Therefore, an adaptive allocation strategy deserves further investigation. We will address this issue in the future.

Acknowledgment

The authors would like to thank Dr. T. Weise for proofreading the manuscript.

References

[1] K. De Jong, *Evolutionary Computation: A Unified Approach*. Cambridge, MA: MIT Press, 2006.

[2] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromachine Human Sci. (MHS)*, Piscataway, NJ, 1995, pp. 39–43.

[3] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[4] A. S. Fukunaga, "Genetic algorithm portfolios," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, La Jolla, CA, 2000, pp. 16–19.

[5] B. A. Huberman, R. M. Lukose, and T. Hogg, "An economics approach to hard computational problems," *Science*, vol. 275, no. 5296, pp. 51–54, Jan. 1997.

[6] C. P. Gomes and B. Selmon, "Algorithm portfolios," *Artif. Intell.*, vol. 126, nos. 1–2, pp. 43–62, Feb. 2001.

[7] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Hongkong, China, 2008, pp. 1110–1116.

[8] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Anchorage, AK, 1998, pp. 69–73.

[9] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput.*, vol. 10, no. 4, pp. 371–395, 2002.

[10] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 2. Edinburgh, U.K., 2005, pp. 1777–1784.

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

[12] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC-2005 special session on real-parameter optimization," Nanyang Tech. Univ., Singapore, Tech. Rep., May 2005 [Online]. Available: http://www3.ntu.edu.sg/home/EPNSugan/index_files/CEC-05/Tech-Report-May-30-05.pdf

[13] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE Congr. Evol. Comput. (CEC'05)*, vol. 2. Edinburgh, U.K., 2005, pp. 1769–1776.

[14] N. Hansen. (2006 Jul.). Compilation of Results on the 2005 CEC Benchmark Function Sets [Online]. Available: http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf

[15] H. Dong, J. He, H. Huang, and W. Hou, "Evolutionary programming using mixed mutation strategies," *Inform. Sci.*, vol. 177, no. 1, pp. 312–327, Jan. 2007.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[17] J. Whitacre, T. Pham, and R. Sarker, "Credit assignment in adaptive evolutionary algorithms," in *Proc. 8th Ann. Conf. Genetic Evol. Comput. (GECCO)*, Seattle, WA, 2006, pp. 1353–1360.

[18] D. Thierens, "An adaptive pursuit strategy for allocating operator probabilities," in *Proc. Conf Genetic Evol. Comput. (GECCO)*, Washington D.C., 2005, pp. 1539–1546.

[19] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multiarmed bandits," in *Proc. 10th Ann. Conf. Genetic Evol. Comput. (GECCO)*, Atlanta, GA, 2008, pp. 913–920.

[20] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 243–259, Apr. 2009.

[21] P. Souza and S. Talukdar, "Asynchronous organizations for multialgorithm problems," in *Proc. ACM/SIGAPP Symp. Appl. Comput.: States-of-the-Art Practice*, Indianapolis, IN, 1993, pp. 286–293.

[22] S. Talukdar, L. Baerentzen, A. Gove, and P. Souza, "Asynchronous teams: Cooperation schemes for autonomous agents," *J. Heurist.*, vol. 4, no. 4, pp. 295–321, Dec. 1998.

[23] J. Rachlin, R. Goodwin, S. Murthy, R. Akkiraju, F. Wu, S. Kumaran, and R. Das, "A-teams: An agent architecture for optimization and decision-support," in *Proc. 5th Int. Workshop Agent Theories, Architectures, Languages (ATAL): Intell. Agents*, 1999, pp. 261–276.

[24] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, Aug. 2010.

[25] E. Cantu-Paz, *Efficient and Accurate Parallel Genetic Algorithms*. Norwell, MA: Kluwer, 2000.

[26] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, Oct. 2002.

[27] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, vol. 4, no. 4, pp. 31–52, May 1999.

[28] S. Tongchim and X. Yao, "Parallel evolutionary programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Portland, OR, 2004, pp. 1362–1367.

[29] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N Vrahatis, "Parallel differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Portland, OR, 2004, pp. 2023–2029.

[30] R. Tanese, "Parallel genetic algorithms for a hypercube," in *Proc. 2nd Int. Conf. Genetic Algorithms (ICGA)*, Cambridge, MA, 1987, pp. 177–183.

[31] S. C. Lin, W. F. Punch, and E. D. Goodman, "Coarse-grain parallel genetic algorithms: Categorization and a new approach," in *Proc. 6th IEEE Symp. Parallel Distributed Process. (SPDP)*, Dallas, TX, 1994, pp. 28–37.

[32] D. Schlierkamp-Voosen and H. Mühlenbein, "Strategy adaptation by competing subpopulations," in *Parallel Problem Solving from Nature*, vol. 3. Berlin, Germany: Springer-Verlag, 1994, pp. 199–208.

[33] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 43–63, Apr. 2000.

[34] M. Herdy, "Reproductive isolation as strategy parameter in hierarchically organized evolution strategies," in *Parallel Problem Solving from Nature*, vol. 2. Amsterdam, The Netherlands: Elsevier, 1992, pp. 207–217.

[35] D. Arnold and A. MacLeod, "Hierarchically organized evolution strategies on the parabolic ridge," in *Proc. 8th Ann. Conf. Genetic Evol. Comput. (GECCO)*, Seattle, WA, 2006, pp. 437–444.

[36] P. Larrañaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA: Kluwer, 2001.

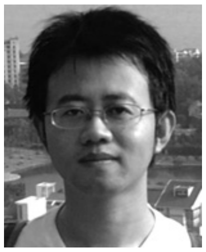
[37] N. Hansen. (2005 Nov.). The CMA Evolution Strategy: A Tutorial [Online]. Available: http://www.lri.fr/∼hansen/cmatutorial.pdf

[38] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[39] Y. Chen, W. Peng, and M. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Trans. Syst., Man, Cybern.–Part B: Cybern.*, vol. 37, no. 6, pp. 1460–1470, Dec. 2007.

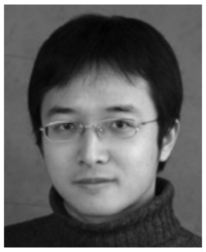
[40] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inform. Sci.*, vol. 178, no. 15, pp. 2985–2999, Aug. 2008.

[41] S. Siegel, *Nonparametric Statistics for the Behavioral Sciences*. New York: McGraw-Hill, 1956.



Fei Peng (S'09) received the B.S. degree in computer science from the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, in 2006. He is currently working toward the Ph.D. degree from the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC.

His current research interests include evolutionary computation and metaheuristic algorithms.



Ke Tang (S'05–M'07) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007.

Since 2007, he has been an Associate Professor with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. He is the coauthor of more than 30 refereed publications. His major research interests include machine learning, pattern analysis, evolutionary computation, data mining, metaheuristic algorithms, and real-world applications.

Dr. Tang is an Associate Editor/Editorial Board Member of four international journals and the Chair of the IEEE Task Force on Large Scale Global Optimization.

Guoliang Chen received the B.S. degree from Xian Jiaotong University, Xian, China, in 1961.

Since 1973, he has been with the University of Science and Technology of China, Hefei, China, where he is currently the Academic Committee Chair of the Nature Inspired Computation and Applications Laboratory, a Professor with the School of Computer Science and Technology, and the Director of the School of Software Engineering. From 1981 to 1983, he was a Visiting Scholar with Purdue University, Purdue, IN. He is currently also the Director of the National High-Performance Computing Center, Hefei, China. He has published nine books and more than 200 research papers. His research interests include parallel algorithms, computer architecture, computer networks, and computational intelligence.

Prof. Chen is an Academician of the Chinese Academy of Sciences. He was the recipient of the National Excellent Teaching Award of China in 2003.

Xin Yao (M'91–SM'96–F'03) received the B.S. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.S. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC, in 1990, all in computer science.

From 1985 to 1990, he was an Associate Lecturer and Lecturer with USTC, while working toward the Ph.D. degree in simulated annealing and evolutionary algorithms. In 1990, he was a Postdoctoral Fellow with the Computer Sciences Laboratory, Australian National University, Canberra, Australia, where he continued his work on simulated annealing and evolutionary algorithms. In 1991, he was with Knowledge-Based Systems Group, Commonwealth Scientific and Industrial Research Organization Division of Building, Construction and Engineering, Melbourne, Australia, where he worked primarily on an industrial project on automatic inspection of sewage pipes. In 1992, he returned to Canberra to take up a Lectureship with the School of Computer Science, University College, University of New South Wales, Australian Defense Force Academy, Sydney, Australia, where he was later promoted to a Senior Lecturer and Associate Professor. Since April 1999, he has been a Professor (Chair) of computer science at the University of Birmingham, Birmingham, U.K. He is currently the Director of the Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham. He is currently also a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. His work has been published in more than 300 refereed publications. His major research interests include evolutionary artificial neural networks, automatic modularization of machine-learning systems, evolutionary optimization, constraint-handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications.

Dr. Yao was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008, an Associate Editor or Editorial Board Member of 12 other journals, and the Editor of the World Scientific Book Series on Advances in Natural Computation. He was the recipient of the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.