Benchmarking Stochastic Algorithms for Global Optimization Problems by Visualizing Confidence Intervals

Qunfeng Liu, Wei-Neng Chen, *Member, IEEE*, Jeremiah D. Deng, *Member, IEEE*, Tianlong Gu, Huaxiang Zhang, Zhengtao Yu, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The popular *performance profiles* and *data profiles* for benchmarking deterministic optimization algorithms are extended to benchmark stochastic algorithms for global optimization problems. A general confidence interval is employed to replace the significance test, which is popular in traditional benchmarking methods but suffering more and more criticisms. Through computing confidence bounds of the general confidence interval and visualizing them with performance profiles and (or) data profiles, our benchmarking method can be used to compare stochastic optimization algorithms by graphs. Compared with traditional benchmarking methods, our method is synthetic statistically and therefore is suitable for large sets of benchmark problems. Compared with some sample-mean-based benchmarking methods, e.g., the method adopted in black-box-optimization-benchmarking workshop/competition, our method considers not only sample means but also sample variances. The most important property of our method is that it is a distribution-free method, i.e., it does not depend on any distribution assumption of the population. This makes it a promising benchmarking method for stochastic optimization algorithms. Some examples are provided to illustrate how to use our method to compare stochastic optimization algorithms.**

***Index Terms*—Benchmarking, computational budget, evolutionary algorithms, global optimization, performance comparison, stochastic algorithms.**

Manuscript received July 5, 2016; revised November 2, 2016; accepted January 20, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 11271069, Grant 61622206, Grant 61379061, and Grant 61332002, and in part by the Natural Science Foundation of Guangdong Province, China under Grant 2015A030313648 and Grant 2015A030306024. This paper was recommended by Associate Editor F. Herrera. *(Corresponding authors: Wei-Neng Chen; Jun Zhang.)*

Q. Liu is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

W.-N. Chen and J. Zhang are with School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com; junzhang@ieee.org).

J. D. Deng is with the Department of Information Science, University of Otago, Dunedin 9054, New Zealand.

T. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

H. Zhang is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China.

Z. Yu is with the School of Information Engineering and Automation, Kunming University of Science and Technology, Kunmin 650500, China.

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

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/TCYB.2017.2659659

I. Introduction

In THIS paper, we consider how to benchmark stochastic algorithms for the following global optimization problem:

$$\min f(x), \quad \text{s.t.,} \quad x \in \Omega \subseteq \mathbb{R}^D \tag{1}$$

where the search space Ω is determined by bound constraints and (or) some other general constraints, and D is the dimension.

For a broad range of applications, the objective function $f(x)$ in problem (1) is multivariate and nonlinear, and there are often multiple local minima. Among all these local minima, the global minimum, a point where the objective function obtains its smallest value, is of main interest in many scientific or engineering problems. However, it is often very hard to find the global minimum. An important reason lies in that there is no suitable mathematical condition for ensuring a point to be the global minimum [1]–[3].

In order to solve the global optimization problem (1), two classes of algorithms were developed. One class seeks to find the global optimum with guaranteed convergence, e.g., the DIRECT algorithm [4] and its variants [5]–[9]. Since there is no necessary condition (let it be a sufficient condition) of global minimum, this class of algorithms have to explore the search space extensively, at least in the worst case. Therefore, they are only suitable for small scale objective functions.

Regardless of the global convergence, another class of algorithms for global optimization problems seeks to find lower objective function values more efficiently. These algorithms often adopt heuristics, natural or biological analogy [10]–[13]. Moreover, they are often stochastic algorithms. Most evolutionary algorithms and swarm intelligence algorithms belong to this class, for example, genetic algorithm (GA) [14], particle swarm optimization (PSO) [15], [16], and differential evolution (DE) [17]. Although "convergence" (or "stability") is also discussed in some papers about the second class of algorithms (see [18]–[21]), the term convergence means only to converge to positions of a certain performance level, not guaranteed to be the global optimum. Therefore, it is a very different concept from the "global convergence" discussed in the first class of algorithms.

Without the theoretical guarantee of the global convergence, there is only one way to verify the usefulness and

efficiency of the second class of algorithms—to perform well on some benchmark problems when compared with some relative algorithms. As a result, it is an important issue to develop suitable techniques for benchmarking the stochastic algorithms for global optimization problems. Unfortunately, it is not an easy task.

A traditional way to benchmark stochastic optimization algorithms is to compare the average minimal objective function values (sample means) found in many independent runs, and verify whether there is a significant difference between any two algorithms [10], [22]–[25]. Specifically, given a fixed computational budget, for each problem, we run each algorithm for many times, and then test whether there is a significant difference between the performance of any two algorithms [24], [25]. In this way, the ranking of the algorithms on each problem can be obtained. Such a process is repeated on all benchmark problems, and the algorithm that performs the best on the most problems is the final winner [10], [22].

Therefore large results tables are often needed to report the large number of sample means and sample variances, and moreover, a large number of figures are often needed to display the performance histories of the sample means for each algorithm and each problem [10], [22]–[25]. As a result, the above traditional method is only suitable for a small set of benchmark problems.

Recently, intermediate cutpoints are adopted to provide more rankings during the process. For example, in the CEC-2014 competition [26], 14 "cutpoints" are adopted during 200 000 function evaluations, and the traditional method is employed at each cutpoint to provide a ranking. In this way, more dynamic information is displayed. However, many tables and (or) many figures are still needed to display the numerical results. In other words, it is still not convenient for a large set of benchmark problems, and therefore, it prevents sufficient numerical evidence (see Section II-B for more discussions).

A difficult issue for benchmarking stochastic algorithms is how to deal with the randomness, especially the sample variances. Suitable parametric significance tests are popularly used in traditional benchmarking methods. However, most parametric significance tests depend on some distribution assumptions which may not hold in practice. As a result, the approaches based on parametric significance test have received notable criticisms due to the distribution assumptions and the frequent improper use of p-values [27]–[30]. Several nonparametric significance tests have been suggested to replace parametric tests to avoid such problems [31].

Recently, the cumulative distribution function (CDF) approach was adopted in the black-box-optimization-benchmarking (BBOB) workshop/competition [32]. Such an approach allows to compare optimization algorithms on a large set of benchmark functions, and provide dynamic rankings on much more cutpoints. However, it is only applied to the sample means, and the sample variances are not considered at all. Although 100 test instances are required for each problem, which makes the sample means more representative, it is still not good to ignore the sample variances (see Section II for more details and discussions).

In this paper, we extend two popular methods—performance profiles method (PPM) [33] and data profiles method (DPM) [34]—to benchmark stochastic global optimization algorithms. PPM and DPM were originally designed for benchmark deterministic local optimization algorithms, deterministic derivative-free optimization algorithms, and some other deterministic algorithms [33], [34]. Both methods adopt CDFs, just like the approach adopted in the BBOB competition, and therefore are suitable for a large set of benchmark problems (see Section III for a brief review and a comparison between these two methods and the BBOB approach).

In order to extend the PPM and DPM for benchmarking stochastic algorithms, the proposed approach adopts a general confidence interval instead of significance tests. The main motivation lies in the fact that confidence interval provides the magnitude of the real effect and therefore it is convenient to generate the CDF of the confidence bounds. In other words, our approach allows to consider both sample means and sample variances with different CDFs. Thus, compared with the method adopted in the CEC competition, our method is convenient for a large set of benchmark functions; compared with the method adopted in the BBOB competition, our method takes the sample variance into account (see Section IV for more details and discussions).

The most important property of the proposed benchmarking method is that it is distribution-free, i.e., it does not depend on nor make use of population's distribution. The only condition is that the independent runs of algorithms on each test problem should be large enough, say, greater than 30. In one word, a distribution-free, confidence interval-based method is proposed in this paper for benchmarking stochastic global optimization algorithms.

The rest of this paper is organized as follows. In Section II, we provide a brief review about the traditional methods for benchmarking stochastic global optimization algorithms. In Section III, we review PPM and DPM for benchmarking deterministic optimization algorithms. In Section IV, we propose our method for benchmarking stochastic global optimization algorithms. In Section V, we first benchmark three popular evolutionary algorithms with the purpose to illustrate how to benchmark stochastic global optimization algorithms with the proposed method. Then we apply our method to the BBOB-2009 data to compare 31 global optimization algorithms. Finally, in Section VI, we give some conclusions.

II. Review of Some Traditional Methods

In this section, we briefly review some traditional methods for benchmarking stochastic optimization algorithms, especially the traditional static comparison methods, the dynamic ranking method adopted in the CEC competition [26], and the dynamic comparison method adopted in the BBOB workshop/competition [32].

A. Static Comparison

Earlier methods consider the performance given a fixed computational budget. For example, given μ_f function

evaluations, run each algorithm m times on each benchmark problem, and then verify if there is a significant difference between algorithms [10], [22]–[25], [35]. Since such comparison is only based on a fixed computational budget, we call it static comparison.

In static comparison, the obtained minimal function value is memorized for each run. Finally, m minimal function values are stored for each algorithm and each problem, which are then used to compute the sample mean, the sample standard deviation [10], [23], [24], [35]–[39] and (or) the success rate [22], [25], [38]. Then a proper significance test, e.g., the Student's t-test, the Welch's t-test or the Wilcoxon's rank-sum test, is adopted to verify whether there is a significant difference between two group of data—m minimal function values in each group, and each group corresponds to an algorithm.

If the difference is significant, then one algorithm has smaller mean, i.e., better performance, than the other. Otherwise, there is not sufficient evidence for stating that one algorithm is better than the other [40]. In this way, for each benchmark problem, there is a ranking of the algorithms. Integrating the rankings for each problem, then a comprehensive ranking of algorithms can be obtained.

There are different ways to integrate these rankings. For example, count the number of benchmark problems on which one algorithm performs the best [10], [22]–[24], [36], [38], [41], the larger the number is, the better the algorithm is. Another way is to sum or average the rankings directly to obtain the final ranking [25], [41]. In [41], an interesting radar chart was developed to show ranking values on different kinds of test sets.

B. Dynamic Ranking

A disadvantage of the above static comparison is that the obtained ranking is based on a fixed computational budget. It is clear that the ranking may be different if another computational budget is selected. A better way is to provide several rankings at several cutpoints within a fixed computational budget.

For example, in the popular CEC competitions,[1] function values at several cutpoints are requested, which are then used to provide dynamic rankings at these cutpoints. In the CEC-2014 competition [26], function evaluations $\mu_f = 10\,000 * D$ and 14 cutpoints $\lambda\mu_f, \lambda = 0.01, 0.02, 0.03, 0.05, 0.1, 0.2, \ldots, 0.9, 1.0$ are adopted, where D is the dimension. Fig. 1 shows an example of dynamic ranking, where five algorithms are compared dynamically at 14 cutpoints. Finally, the algorithm A1 wins at the 14th cutpoint.

The final results provided by the static comparison and the dynamic rankings are some rankings. From these rankings, it is hard to know the information among the raw data. For example, how far is one algorithm's sample mean from the best algorithm's sample mean? In order to display the raw performance data, a popular way is to summarize

[1]An annual competition in the IEEE Conference on Evolutionary Computation since 2005 (see http://www.ntu.edu.sg/home/EPNSugan/for more details).

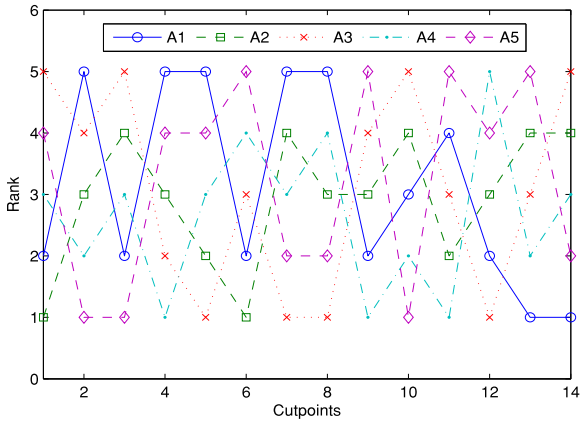


Fig. 1. Example of dynamic ranking. Five algorithms are ranked at 14 different cutpoints. Finally, the algorithm A1 wins at the 14th cutpoint.

the sample means and the sample standard deviations (or sample variances) for each algorithm on each problem in a table [10], [22]–[25], [35], which is often very large when the number of algorithms or (and) the number of benchmark problems is not small. Another popular way is to display the history of the sample means for each algorithm on each problem with figures [10], [22]–[25]. However, it needs a large number of figures to show the histories of these sample means, one figure for each problem.

Therefore, the above methods are not suitable for a large set of benchmark problems. Without testing on a large number of benchmark problems, it is insufficient for any claim about which algorithm performs the best. We note that if the benchmark problems are uniformly sampled from a representative test suit (ideally, an infinite set), then the results of numerical comparison will not depend on the number of benchmark problems, and even a single run per problem instance is sufficient when average performance is concerned [42]. However, such condition is too strict to be guaranteed in most popular sets of test functions. Therefore, numerical results based on a large set of test functions are often more reliable than those based on a small set. In this sense, it is a good property for a methodology if it is suitable for numerical comparison on a large benchmark set.

If the history of the best function values is provided, then ranking values at much more cutpoints can be obtained. For instance, in [42] and [43], the evolution of the ranking values of five algorithms are shown at as many points as function evaluations or time steps. Moreover, confidence intervals of the ranking values are also shown in graph (see Fig. 2 for an example). Several nonparametric tests are adopted to verify whether the ranking values are significantly different. These nonparameter test methods are distribution-free. However, the above methods provide only the ranking values in graphs, without revealing the magnitude of the performance difference.

C. Dynamic Comparison by Cumulative Distribution Function

In order to compare algorithms on a large set of benchmark problems, it is necessary to synthesize the information

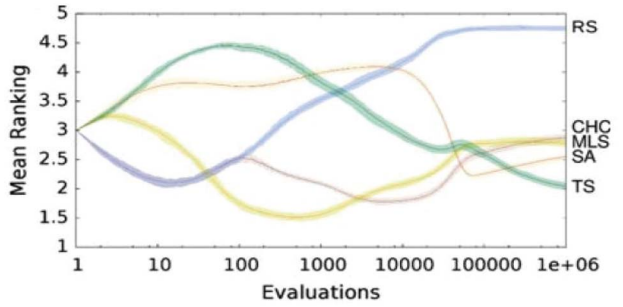


Fig. 2. Evolution of the ranking values of five algorithms on 1000 problems [42]. The lines are the mean ranking of five algorithms and the areas around them measure their confidence intervals, which are often narrow.

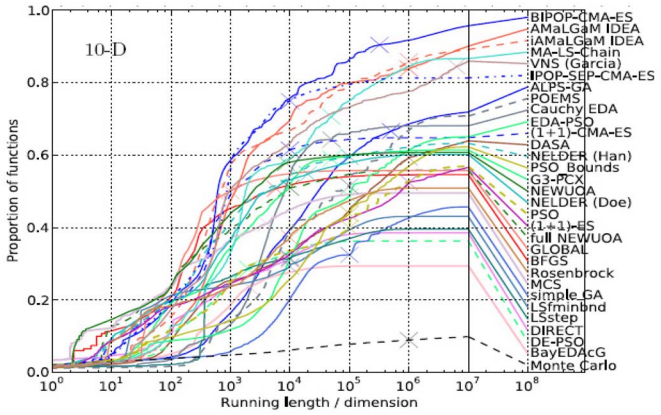


Fig. 3. CDFs obtained from BBOB-2009 [32].

obtained from testing on each problem. The CDF is a convenient tool for this purpose, and it presents the magnitude of performance difference. For example, it is adopted to compare dozens of algorithms in the popular BBOB workshop/competition[2] hold at the Genetic and Evolutionary Computation Conference since 2009 and at the IEEE CEC since 2015.

For instance, the CDF-based approach was adopted to compare 31 real-parameter optimization algorithms from BBOB-2009 [32]. Fig. 3 (from [32]) shows the empirical runtime distributions (i.e., CDF) for these algorithms on all functions in dimension 10.

For each graph in Fig. 3, the runtime was measured as the number of function evaluations needed to reach a given target precision

$$\Delta f_t = f_{\text{target}} - f_{\text{opt}} \tag{2}$$

where f_{opt} is the optimal function value. Twenty-four BBOB benchmark functions were tested, and 50 different target precisions $\Delta f_t = 10^{1.8}, 10^{1.6}, 10^{1.4}, \ldots, 10^{-8}$ were adopted. For each function-Δf_t-pair, 100 instances of runtime were generated, and then the CDF for each algorithm was obtained.

The x-value in Fig. 3 is the computational budget measured as the number of function evaluations divided by dimension, and the y-value is the proportion of solved function-Δf_t-pairs.

[2]See http://coco.gforge.inria.fr/doku.php?id=start for more details about this competition.

Therefore, the graph tells the proportion of function-Δf_t-pairs that the Δf_t-value is reached within any given computational budget. Moreover, given any y-value, the horizontal distance between any two graphs reveals the average runtime difference, which is regarded as the most useful aggregated performance measure in [32].

In a sense, the CDF-based approach adopted in the BBOB competition can provide more information (e.g., how many problems are solved within a given budget) than the other approaches such as the dynamic ranking adopted in CEC competition. Moreover, it is convenient to compare algorithms on a large set of benchmark functions. For instance, 1200 function-Δf_t pairs were tested in BBOB-2009. However, only the average behaviors (i.e., the sample means) were considered in the BBOB competition. On the contrary, in the method adopted in the CEC competition, [42], [43], the sample variances are also analyzed through conducting some statistics tests. It is well known that both average and variance are important for any random variable. Therefore, we believe that it is a disadvantage for the BBOB approach to ignore the sample variances.

III. PPM and DPM for Deterministic Optimization Algorithms

In this section, we briefly review the PPM proposed in [33] and the DPM proposed in [34], and then compare them with the BBOB approach.

A. PPM and DPM

Both PPM and DPM are very popular in the community of mathematical programming. They adopt the CDFs to display the raw data gathered during the benchmark process, an approach that is suitable for a large set of benchmark problems.

In order to describe the PPM and the DPM, we denote $\mathbb{S}$ as the set of algorithms that need to be compared, $\mathbb{P}$ as the set of benchmark problems, and suppose that there are n_s algorithms in $\mathbb{S}$ and n_p benchmark problems in $\mathbb{P}$, respectively.

First, given a budget of function evaluations μ_f, we test each algorithm $s \in \mathbb{S}$ on each problem $p \in \mathbb{P}$, and record the history of the obtained best function values. After all tests are finished, we obtain a matrix H with size $\mu_f \times n_p \times n_s$, the element $H(k, j, i)$ denotes the found best function value during k function evaluations when test the ith algorithm on the jth problem.

Denote $t_{p,s}$ as the number of function evaluations needed for algorithm s on problem p to find a position x such that the following convergence condition holds:

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L) \tag{3}$$

where $f(x_0)$ is the objective function value at the starting point x_0, $\tau > 0$ is a tolerance and f_L is the smallest objective function value obtained by any algorithm within μ_f function evaluations. $t_{p.s} = \infty$ if the condition (3) does not satisfy after μ_f function evaluations.

Remark 1: All algorithms are required to search from the same starting point x_0. The definition of f_L guarantees that

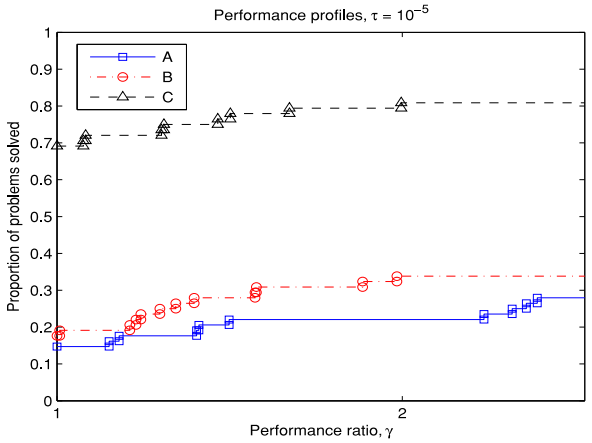


Fig. 4. Example of PPM where three algorithms are compared. The algorithm C solves about 70% problems with the best efficiency, and finally solve more than 80% problems. On the contrary, the algorithms A and B perform much worse than C.

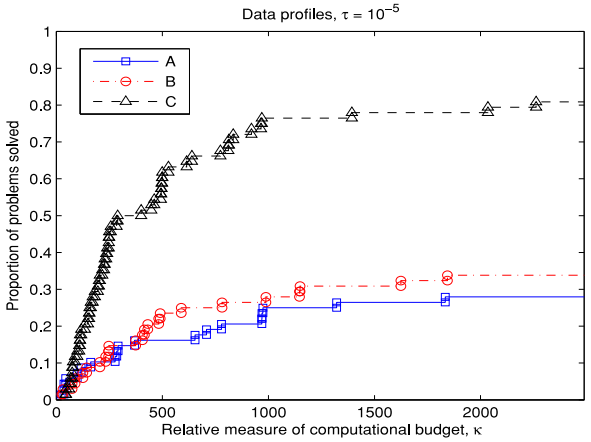


Fig. 5. Example of DPM where three algorithms are compared. The algorithm C solves more problems than both A and B for almost any given relative measure of computational budget, and finally solve more than 80% problems. Algorithms A and B only solve about 30% problems finally.

there is at least one algorithm will satisfy the condition (3) for any $\tau > 0$ and the given μ_f (see [34] for more details).

If the convergence condition (3) is satisfied within μ_f function evaluations, then we say that problem p is solved by algorithm s. By checking this condition for each test problem, we obtain the proportion of problems solved by algorithm s. With this process repeated for each algorithm under different computation cost, we can further compare these proportions for different computation budget. This is the common basic idea of PPM and DPM. The main difference between PPM and DPM lies in measures of computation budget.

Then we describe how to use the matrix H to define the data profile and the performance profile for each algorithm.

In PPM [33], the performance profile is defined for any algorithm $s \in \mathbb{S}$ as the following CDF:

$$\rho_s(\gamma) = \frac{1}{n_p}\left|\left\{p \in \mathbb{P} : \frac{t_{p,s}}{\min\{t_{p,s}, s \in \mathbb{S}\}} \le \gamma\right\}\right| \tag{4}$$

where $|\cdot|$ returns the number of elements of a set. In (4), the ratio

$$\frac{t_{p,s}}{\min\{t_{p,s}, s \in \mathbb{S}\}}$$

is often called as the performance ratio. Therefore, $\rho_s(\gamma)$ measures the proportion of problems that can be solved by the algorithm s within performance ratio γ. It is clear that the performance ratio has a minimal value 1, and therefore, γ should not be less than 1. Particularly, $\rho_s(1)$ measures the proportion of problems that the algorithm s can perform the best among all algorithms in $\mathbb{S}$.

Fig. 4 shows an example of PPM, where three algorithms A, B, and C are compared at the level of tolerance $\tau = 10^{-5}$.

Remark 2: It can be seen from condition (3) that tolerance τ determines the relative accuracy of solution x to f_L. In the original PPM and DPM, $\tau = 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}$ are adopted, respectively, to compare algorithms on different levels of accuracy. Higher accuracy is often desired in many applications, therefore, $\tau = 10^{-5}$ or 10^{-7} is more popular. In the following numerical comparisons, only the results of $\tau = 10^{-5}$ are reported because they are similar to those of $\tau = 10^{-7}$.

From Fig. 4, we see that algorithm C can solve about 70% problems with the best efficiency, and solve about 80% of problems finally. It is clear that algorithm C performs much better than A and B.

In DPM [34], the data profile is defined for any algorithm $s \in \mathbb{S}$ as the following CDF:

$$d_s(\kappa) = \frac{1}{n_p}\left|\left\{p \in \mathbb{P} : \frac{t_{p,s}}{D_p + 1} \le \kappa\right\}\right| \tag{5}$$

where D_p is the dimension of the problem p. In (5), the ratio

$$\frac{t_{p,s}}{D_p + 1}$$

is called as the relative measure of computational budget. Since $D_p + 1$ refers to the number of function evaluations needed to compute one simplex gradient [44], such ratio is also called as the number of simplex gradients [34].

Fig. 5 shows an example of DPM, which adopts the same matrix H as that in Fig. 4. From Fig. 5, we see that the algorithm C solves about 80% problems finally, just the same as that seen from Fig. 4. Actually, if the performance profiles and the data profiles are generated from the same matrix H, then for each algorithm, the proportion of problems that can be solved finally in terms of performance profile is the same as that in data profile. This property was described in [34] with an equation in the form of

$$\lim_{\kappa \to \infty} d_s(\kappa) = \lim_{\alpha \to \infty} \rho_s(\alpha).$$

Remark 3: As revealed in (4) and (5), both data profiles and performance profiles are CDFs, and therefore are monotone increasing curves. The data profiles show how the proportion of solved-problems increases as the relative measure of computational budget increases, while the performance profiles show how the proportion of solved-problems increases as the performance ratio increases. Therefore, if we care about how the ranking values and performance differences change

as computational cost increases, then both PPM and DPM are necessary since they provide complementary information. However, if only the final results of numerical comparison are concerned, then either PPM or DPM is enough since they provide the same final results.

B. Comparing DPM With the BBOB Approach

Although the approach adopted in the BBOB competition is designed for benchmarking stochastic optimization algorithms, it is very similar to DPM in consequence of the following facts. First, the BBOB approach only considers the sample means but ignores the sample variances, which makes it similar to a deterministic method. Second, both BBOB approach and DPM adopt CDF. What is more, the x- and y-value of the CDFs of both methods (see Figs. 3 and 5, respectively) are very similar. Actually, they have the same y-values—the proportion of problems solved. Their x-values are very similar since both of them measure a relative computational budget—the number of function evaluations divided by dimension (BBOB approach) or dimension plus 1 (DPM).

Their difference lies mainly in the convergence rules. The BBOB approach seeks to find a solution f_{target} which is near enough (defined by the precision Δf_t [see (2)]) to the optimal function value. Not surprisingly, the required computational budget is often high to satisfy the convergence condition. On the contrary, the convergence rule of DPM [see condition (3)] is much looser, and it can work under any computational budget since there is always at least one algorithm that can satisfy the condition.

Roughly speaking, the BBOB approach is suitable for computational cheap benchmark functions whose global optimal function values are known. On the contrary, DPM can be used on different kinds of benchmark functions, regardless of whether their global optimal function values are known or not. If the best function value f_L in condition (3) is replaced by the global optimal function value f_{opt}, DPM is almost the same as the BBOB approach. In other words, DPM is more generic than the BBOB approach.

IV. Benchmarking Stochastic Optimization Algorithms

PPM and DPM reviewed in the above section are suitable only for deterministic optimization algorithms. It is clear that stochastic optimization algorithms are much harder to compare than deterministic ones due to the randomness.

In order to deal with stochastic optimization algorithms, we introduce a general confidence interval into DPM and PPM. In this way, we can generate CDFs for both sample means and sample variances (which is contained in confidence bounds). We will show that the proposed method is convenient for benchmarking stochastic optimization algorithms. For convenience, we denote our methods for benchmarking stochastic global optimization algorithms based on PPM and DPM as PPMS and DPMS, respectively. And we need run PPM (or DPM) two times for each run of PPMS (or DPMS).

A. General Confidence Interval

Definition 1: Let $\{X_1, \ldots, X_n\}$ be an independent sample from a population X with mean μ but unknown distribution. Then we call

$$\left[\overline{X} - \frac{\lambda S}{\sqrt{n}}, \quad \overline{X} + \frac{\lambda S}{\sqrt{n}}\right] \tag{6}$$

a general confidence interval for μ, where $\lambda \geq 2$ and

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 \tag{7}$$

are the sample mean and the sample variance, respectively. Moreover, $\overline{X} - (\lambda S/\sqrt{n})$, $\overline{X} + (\lambda S/\sqrt{n})$ are called the confidence lower and upper bound, respectively.

When the population X is normally distributed and $\lambda = 2$, (6) defines a theoretical confidence interval for μ, the confidence level is about 95%. In other cases, (6) defines a general confidence interval for μ, but the confidence level is unknown. However, we can increase it through simply adopting a larger constant λ, say $\lambda = 3$.

B. Confidence Intervals and Significance Test

In this section, we first describe how the general confidence intervals (6) can determine whether there is significant performance difference between two algorithms, and then provide some reasons why we propose to replace significance test with confidence interval.

Suppose we have two samples, each contains n best function values found by one algorithm. Then we obtain two confidence intervals

$$\left[\overline{X}_1 - \frac{\lambda S_1}{\sqrt{n}}, \quad \overline{X}_1 + \frac{\lambda S_1}{\sqrt{n}}\right], \left[\overline{X}_2 - \frac{\lambda S_2}{\sqrt{n}}, \quad \overline{X}_2 + \frac{\lambda S_2}{\sqrt{n}}\right] \tag{8}$$

where $\overline{X}_i, S_i^2$ are the sample mean and sample variance of the ith algorithm, $i = 1, 2$, respectively. The following proposition shows that if the confidence intervals (8) do not overlap then there is significant performance difference between these two algorithms. The only condition is that n is large enough, say $n \geq 30$, regardless of the distributions of the two populations.

Proposition 1: Suppose that we are comparing two algorithms' performance, and the confidence intervals (8) are their general confidence intervals, respectively, where n is large enough, say $n \geq 30$. If the confidence intervals (8) do not overlap, then their performances are significantly different. However, the converse is not true.

Proof: Suppose that the confidence intervals do not overlap. Without loss of generality, let

$$\overline{X}_1 + \frac{\lambda S_1}{\sqrt{n}} < \overline{X}_2 - \frac{\lambda S_2}{\sqrt{n}}.$$

Then

$$\overline{X}_2 - \overline{X}_1 > \frac{\lambda(S_1 + S_2)}{\sqrt{n}} \geq \lambda\sqrt{\frac{S_1^2 + S_2^2}{n}}.$$

When n is large enough, say $n \geq 30$, both $\overline{X}_1$ and $\overline{X}_2$ satisfy approximately normal distribution due to the central limitation theorem, regardless the populations' distributions.

Therefore, $\overline{X}_2 - \overline{X}_1$ also satisfies approximately a normal distribution whose variance is $[(S_1^2 + S_2^2)/n]$ [45]. Hence the z value in a z-test is

$$z = \frac{\overline{X}_2 - \overline{X}_1}{\sqrt{\frac{S_1^2+S_2^2}{n}}} > \lambda \geq 2.$$

In other words, when $n \geq 30$, the two-tailed z-test p-value is less than 0.025, i.e., the performances are significantly different.

We are going to prove with an example that the converse is not true. Suppose $\overline{X}_1 = 9, \overline{X}_2 = 17, n = 100$, and $S_1 = S_2 = 2.5$. Then

$$z = \frac{17 - 9}{\sqrt{2.5^2 + 2.5^2}} = 2.26$$

therefore, the performances are significant different at the significance level 0.05. However, the confidence intervals are $(4, 14)$ and $(12, 22)$, respectively, and they are overlapping. ■

Remark 4: In order to satisfy the central limit theorem, the sample size n should be large enough. Empirically, $n \geq 30$ is often necessary, and $n \geq 50$ often results in good approximation for most cases [45]. Therefore, $n \geq 30$ is required for our method, and $n \geq 50$ would be better.

The above proposition implies that the confidence interval method is stricter than the significance test method. If the confidence interval method implies a significant difference exists, so does the significance test method. However, the converse maybe not true.

Moreover, there are other good reasons to replace significance test with confidence intervals. First, unlike the parametric significance test, the general confidence intervals (6) do not depend on any distribution assumptions which are often used improperly and brings numerous criticisms for significance test [27]–[30]. Second, confidence intervals provide information about the magnitude of the real effect. This leads to the last but maybe the most important reason that the magnitude of the real effect provided by confidence interval and the sample means can be displayed in the same way with DPM and (or) PPM. This fact, combined with Proposition 1, makes the dynamic comparison of stochastic algorithms easy to implement. What we need to do is just to display not only the sample means but also the confidence bounds with DPM and (or) PPM, and Proposition 1 will determine whether there is a significant performance difference.

C. Sample Mean Matrix and Confidence Bound Matrix

Given a budget of function evaluations μ_f, we run each algorithm $s \in \mathbb{S}$ on each problem $p \in \mathbb{P}$ for n_r times, and record the found best function values. After all tests are finished, we obtain a matrix H with size $\mu_f \times n_r \times n_p \times n_s$, the element $H(k, r, j, i)$ denotes the found best function value during k function evaluations at the rth run when test the ith algorithm on the jth problem. Then the matrix H is used to generate the sample mean matrix $\overline{H}$, the sample variance matrix S_H^2, the confidence upper bound matrix H_{upper} and the confidence lower bound matrix H_{lower} for each algorithm $i = 1, \ldots, n_s$ and each problem $j = 1, \ldots, n_p$ with $k = 1, \ldots, \mu_f$ function evaluations.

Specifically, the sample mean matrix $\overline{H}$ for algorithm i and problem j with k function evaluations is defined by

$$\overline{H}(k, j, i) = \frac{1}{n_r} \sum_{r=1}^{n_r} H(k, r, j, i) \tag{9}$$

and the sample variance matrix S_H^2 is defined by

$$S_H^2(k, j, i) = \frac{1}{n_r - 1} \sum_{r=1}^{n_r} \left[H(k, r, j, i) - \overline{H}(k, j, i)\right]^2. \tag{10}$$

Here $\overline{H}(k, j, i)$ and $S_H^2(k, j, i)$ denote the sample mean and the sample variance of the sample $\{H(k, 1, j, i), \ldots, H(k, n_r, j, i)\}$, respectively. According to the confidence interval (6), the confidence upper and lower bound matrix $H_{\text{upper}}, H_{\text{lower}}$ for algorithm i and problem j at k function evaluations are defined by

$$H_{\text{upper}}(k, j, i) = \overline{H}(k, j, i) + \frac{\lambda S_H(k, j, i)}{\sqrt{n_r}} \tag{11a}$$

$$H_{\text{lower}}(k, j, i) = \overline{H}(k, j, i) - \frac{\lambda S_H(k, j, i)}{\sqrt{n_r}}. \tag{11b}$$

Since the sample mean matrix $\overline{H}$, the confidence bounds matrix H_{upper} and H_{lower} share the same structure as the data matrix H obtained in the deterministic case, all of them can be displayed with PPM (4) and (or) DPM (5). This fact, combined with Proposition 1, implies that the stochastic global optimization algorithms can be benchmarked by the proposed PPMS and DPMS. The following two sections will show how this can be done. In the rest of this paper, we adopt $\lambda = 2$ to compute the confidence bounds (11a) and (11b) unless other value of λ is provided explicitly.

D. Comparing Two Stochastic Optimization Algorithms

In this section, we consider the case that there are only two stochastic optimization algorithms, i.e., $n_s = 2$, and the case of $n_s \geq 3$ will be discussed in next section.

Before displaying the data, it needs to note the choice of the starting point x_0 in the convergence condition (3). If the stochastic global optimization algorithm starts its search from a unique point, then this point is regarded as the starting point x_0. If the stochastic global optimization algorithms is population-based, then the point with best function value in the initial population is regarded as the starting point x_0. The important thing is, the starting point x_0 must be the same for all algorithms.

It is also important to note that since we consider minimization problems here, if H_{upper} and H_{lower} are two different algorithms performance data, then H_{upper} performs worse than H_{lower}.

Now we begin to show how to compare two stochastic global optimization algorithms with PPMS and (or) DPMS. First, we compare the sample means to determine a winner. Then we compare the upper bound of the winner (worse than its sample means) with the lower bound of the loser (better than its sample means) to confirm the winner. In both situations, a higher profile

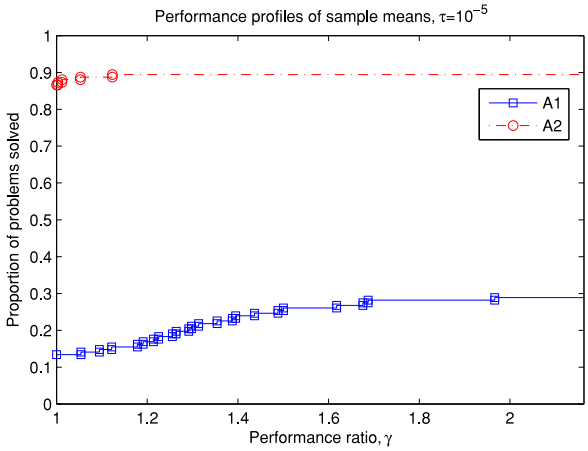


Fig. 6. Example of comparing two stochastic optimization algorithms with PPMS. The sample mean matrix $\overline{H}$ is displayed with PPM. It is clear that the algorithm A2 performs much better than A1. However, when the sample variances are taken into account, does A2 still perform better than A1? See Fig. 7 for an answer.

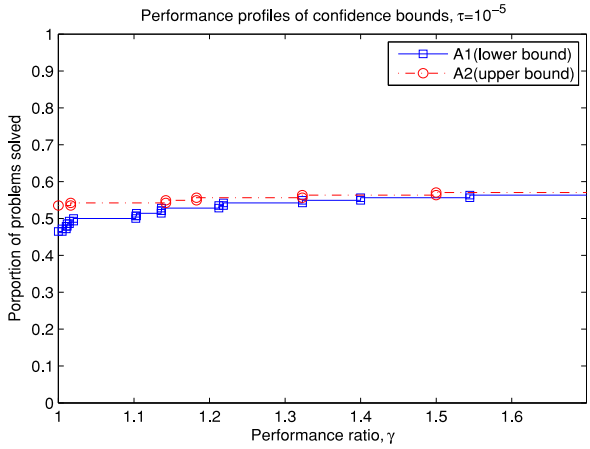


Fig. 7. Confidence upper bound for A2 and confidence lower bound for A1 are displayed with PPM. It is clear that A2 performs slightly better than A1. Combined with Fig. 6, A2 performs significantly better than A1.

Algorithm 1 Comparing Two Algorithms

Given the sample mean matrix $\overline{H}$, the confidence bound matrixes H_{upper} and H_{lower}, and each problem's dimension $D_p, p = 1, \ldots, n_p$.

* Display the sample mean matrix $\overline{H}$ with PPM or DPM, and denote the winner as the n_b-th algorithm;
* Display the confidence ***upper*** bound matrix of the winner $H_{upper}(:, :, n_b)$ together with the confidence ***lower*** bound matrix of the other algorithm $H_{lower}(:, :, n_w), n_w \neq n_b$.
  - If the n_b-th algorithm still wins, then it performs significantly better than the other algorithm.
  - If the n_b-th algorithm loses, then there is no significant performance difference between them.

Algorithm 2 Comparing More Than Two Algorithms

Given the average matrix $\overline{H}$, the confidence bound matrixes H_{upper} and H_{lower}, and each problem's dimension $D_p, p = 1, \ldots, n_p$.

* Display the sample mean matrix $\overline{H}$ with PPM or DPM, and denote the winner as the n_b-th algorithm;
* Display the confidence ***upper*** bound matrix of the winner $H_{upper}(:, :, n_b)$ together with the confidence ***lower*** bound matrices of the other algorithms $H_{lower}(:, :, n_w), n_w \neq n_b$. Then we can have the following conclusions.
  - For each algorithm, if it performs worse than the n_b-th algorithm in the figure, it implies that this algorithm performs significantly worse than n_b-th algorithm.
  - Otherwise, if it performs better than the n_b-th algorithm in the figure, it implies that there is not significant performance difference between it and the n_b-th algorithm.

curve corresponds to a better algorithm since it can solve more problems than another algorithm under given computational cost. If these profiles are crossing, then the algorithm with a higher profile value at the right endpoint is regarded as the better one, just like in the traditional methods.

The details of this method are presented in Algorithm 1.

In the first step of Algorithm 1, both algorithms' sample means are compared, and the winner is determined if its profile is higher. In the second step, confidence bounds are compared, and if the winner's profile is still higher, then it performs significantly better than the nonwinner due to Proposition 1. Otherwise, we regard that there is no significant performance difference between these two algorithms.

Remark 5: It is clear that the upper bound matrix H_{upper} is almost always greater than the lower bound matrix H_{lower} (for any given k, j, i). Therefore, if we consider them as two different algorithms' performance data, then H_{upper} performs worse than H_{lower} since we consider the minimization problem. In the graph of both data profiles and performance profiles, the vertical axis is not real value but proportion of problems solved. Therefore, H_{upper}'s profile is often lower than that of H_{lower} since H_{upper} performs worse than H_{lower}.

Figs. 6 and 7 show an example of comparing two stochastic optimization algorithms with PPMS. First, the sample mean matrix $\overline{H}$ is displayed in Fig. 6, and we see that the algorithm A2 performs much better than A1. However, when the sample variances are taken into account, does A2 still performs better than A1? In Fig. 7, the confidence upper bound of A2 and confidence lower bound of A1 are displayed together. It is clear that A2 still performs better than A1. Therefore, we can conclude that A2 performs significantly better than A1.

Figs. 8 and 9 adopt DPMS to compare the same stochastic optimization algorithms A1 and A2. The same results are obtained.

E. Comparing More Than Two Algorithms

Now we consider the case that $n_s \geq 3$. The details are presented in the following Algorithm 2.

If our purpose is to find the best algorithm, then one run of Algorithm 2 is enough, like some nonparametric test methods [31]. However, it is usually needed to perform at least $n_s - 1$ significance tests for traditional parametric significance test-based methods. Furthermore, if we need to rank all

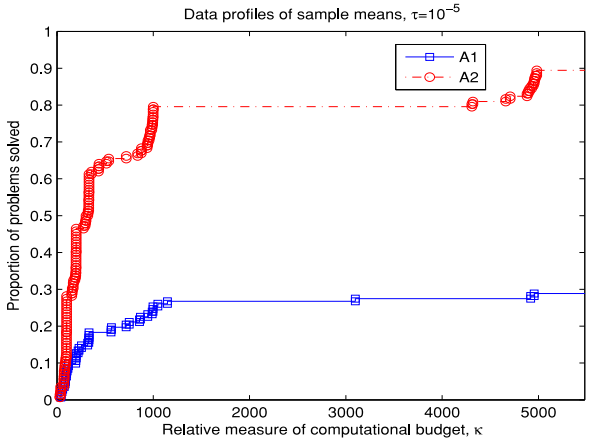


Fig. 8. Example of comparing two stochastic optimization algorithms with DPMS. The sample mean matrix $\overline{H}$ is displayed with DPM. It is clear that the algorithm A2 performs much better than A1. However, when the sample variances are taken into account, does A2 still perform better than A1? See Fig. 9 for an answer.

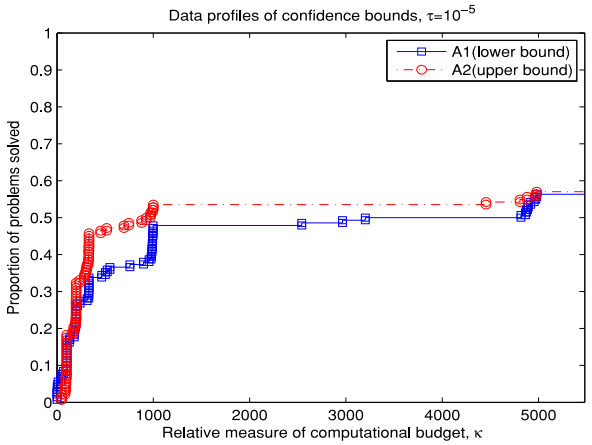


Fig. 9. Confidence upper bound for A2 and confidence lower bound for A1 are displayed with DPM. It is clear that A2 still performs better than A1. Combined with Fig. 8, A2 performs significantly better than A1.

algorithms, repeat Algorithm 2 and drop the best algorithm(s) obtained in the previous run.

Figs. 10 and 11 show an example of comparing five stochastic optimization algorithms with DPMS. In Fig. 10, the sample mean matrix $\overline{H}$ is displayed, and the algorithm A2 performs much better than the other algorithms. However, when the sample variances are taken into account, does A2 still perform better than the other algorithms? Fig. 11 displays the confidence upper bound matrix of A2 and the confidence lower bound matrices of the other algorithms. It is clear that A2 still performs better than the other algorithms except A3. Combined with Fig. 10, A2 got the best data profile of the mean values and it is statistically different from the results of the other algorithms, except for A3.

F. Discussion

Compared with the traditional parametric significance-test-based methods, our method possesses several advantages.

1) It is distribution-free, i.e., it does not depend on nor make use of any distribution assumption of the population. In other words, whatever the population distribution is, the proposed method can be used to compare stochastic optimization algorithms.
2) It is convenient for a large set of benchmark problems.
3) It needs only one comparison to select the best algorithm.
4) It is convenient to compare stochastic optimization algorithms with the deterministic optimization algorithms. What we need to do is just to replace the sample means and confidence bounds with the found best function values for the deterministic optimization algorithms.
5) It provides more information including the ranking. From each profile, we can see how many problems are solved during any given relative measure of computational budget or performance ratio. Moreover, the performance profiles present how many problems each algorithm can perform the best, i.e., the value $\rho(0)$.

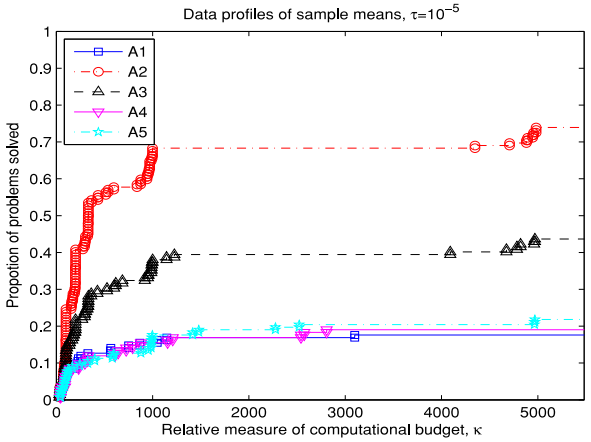


Fig. 10. Example of comparing more than two stochastic optimization algorithms with DPMS. The sample mean matrix $\overline{H}$ is displayed with DPM. It is clear that the algorithm A2 performs much better than the other algorithms. However, when the sample variances are taken into account, does A2 still perform better than the other algorithms? See Fig. 11 for an answer.

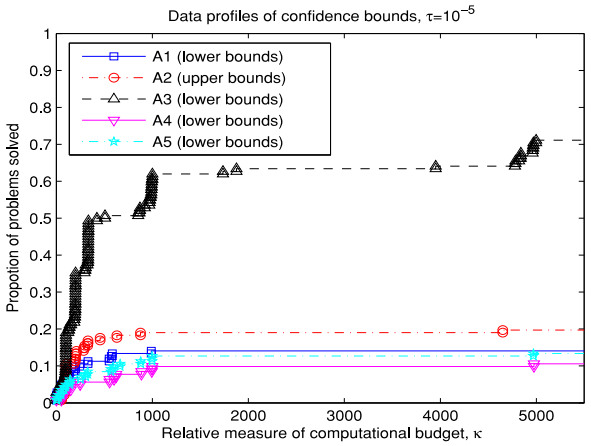


Fig. 11. Confidence upper bounds for A2 and confidence lower bounds for the other algorithms are displayed with DPM. It is clear that A2 still performs better than the other algorithms except A3. Combined with Fig. 10, A2 performs significantly better than the other algorithms except for A3.

Compared with most nonparametric test-based methods, our method possesses the second, the fourth, and the fifth advantage, especially the second one and the fifth one.

Actually, most of the above advantages are also possessed by the BBOB approach. However, the latter obtains these advantages through considering only the sample means while ignoring the sample variances, which we believe is not considerate for stochastic algorithms. For example, in Fig. 10, A2 performs much better than A3 from the view of sample means. However, if the sample variances are taken into account, there is no significant difference between A2 and A3 [under the considered confidence level of the confidence intervals (see Fig. 11)]. Therefore, compared with the BBOB method, our method takes sample variances into account without depending on any distribution assumptions.

To sum up, our benchmarking method is distribution-free, confidence interval-based and CDF-based. It is convenient for benchmarking optimization algorithms (stochastic, or deterministic or both) by visualizing confidence intervals.

It is necessary to point out that we do not know the exact significance level of our result due to the absence of the population's distribution. However, we do know how to increase the confidence level—simply adopting a larger constant λ in the confidence interval (6). The larger λ is, the higher the significance level is.

Another question we need to point out is the selection of PPMS and DPMS. If we care about how the ranking values and performance differences change as computational cost increases, then both PPMS and DPMS are necessary since they provide complementary information. However, if only the final results are concerned, then either PPMS or DPMS is enough since they always provide the same final results (see Remark 3).

Finally, when applying our benchmarking method, a large enough number of (no less than 30) *independent* runs are needed for each benchmark function. If these runs are not really independent, then the obtained confidence intervals will be unreliable due to the problem of pseudoreplication [46]–[49].

V. Numerical Experiments

In this section, we provide two kinds of numerical comparisons. First, we benchmark the following three population-based stochastic optimization algorithms on the CEC-2014 test suite [26].

1) *GA:* The GA [14] implemented in the MATLAB global optimization toolbox.
2) *DE:* The DE algorithm [17]. In our experiments, we select a basic DE solver—DeMat implemented in [50].
3) *SPSO:* The standard PSO [51], which is an implement of the original PSO [15], [16].

Second, we adopt our method to analyze the BBOB-2009 data. The main purpose is to provide some real examples to show how our benchmarking method can be used to compare stochastic optimization algorithms.

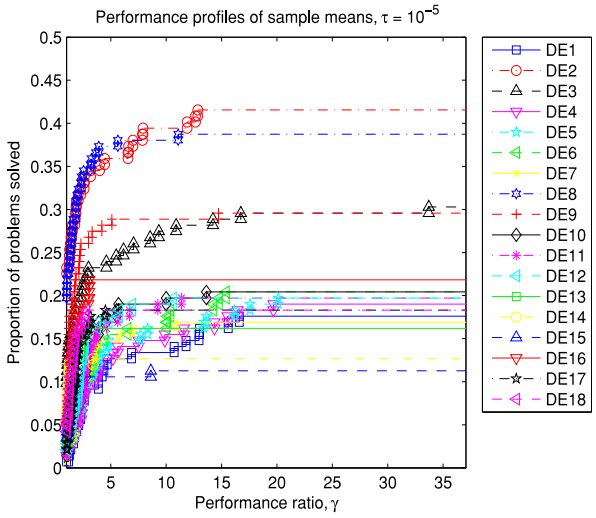


Fig. 12. Sample means matrices of 18 different DEs are displayed with PPM. DE2 performs the best. However, we select DE8 to compare with GA and SPSO.

A. Comparison of GA, DE, and SPSO

The population size for all these three algorithms are set to be 20. For our purpose, the stopping conditions of these algorithms are modified so that they will stop only after $\mu_f = 10\,000$ function evaluations are consumed. All other parameters are set to be their default values. Since there are no default values for DE's two parameters: the strategy s and the crossover probability constant F_CR, we try to optimize their values by our benchmarking method.

The CEC-2014 test set [26] is adopted for our comparison, which is extended from the popular benchmark suit CEC-2005 [52] for evolutionary computation, whose codes can be downloaded at http://www.ntu.edu.sg/home/epnsugan/. For each benchmark problem, we have tested five different scales of dimensions $D_p = 2, 10, 30, 50, 100$ except for eight problems (F17–F22, F29, and F30) which are not defined for $D_p = 2$. Totally, we have tested 142 problems. For each benchmark problem, $n_r = 50$ runs were executed for GA, DE, and SPSO.

1) Comparison of DEs: In this section, we benchmark 18 DE solvers which have the same parameter setting except the strategy s and F_CR.

1) DEi : $s = i$, $F\_\mathrm{CR} = 0.1$, $i = 1, 2, \ldots, 6$.
2) DE$(i + 6)$: $s = i$, $F\_\mathrm{CR} = 0.5$, $i = 1, 2, \ldots, 6$.
3) DE$(i + 12)$: $s = i$, $F\_\mathrm{CR} = 1$, $i = 1, 2, \ldots, 6$.

The first purpose in this section is to provide a real example which shows how our benchmarking method can be used to compare dozens of (or even more) stochastic optimization algorithms. The second purpose is to select the best DE solver to compare with GA and SPSO in the next section.

Fig. 12 displays the sample mean matrix with PPM. We can see that DE2 and DE8 perform much better than the other DEs. Although DE2 performs slightly better than DE8, we decide to select DE8 to compare with GA and SPSO in the next section. The reasons lie in two facts: first, it got the second best mean matrix in our experiments; second, with $F\_\mathrm{CR} = 0.5$ in DE8,

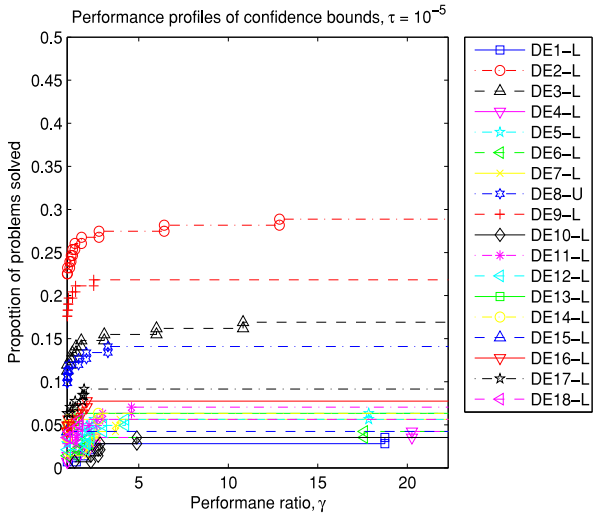


Fig. 13. Confidence upper bound matrix of DE8 and the confidence lower bound matrices of the other 17 DEs are displayed with PPM. All DEs perform worse than DE8 except DE2, DE3, and DE9. Combined with Fig. 12, DE8 performs significantly better than the other DEs, except for DE2, DE3, and DE9.

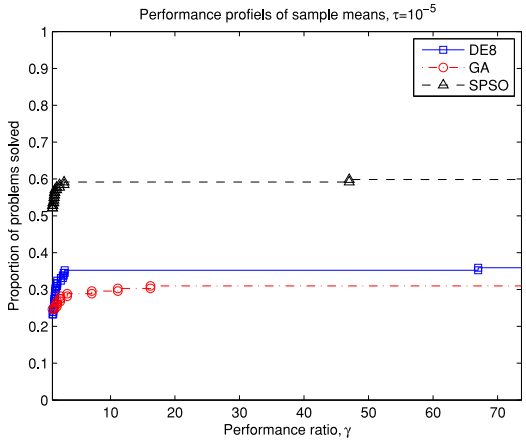


Fig. 14. Sample mean matrix is displayed with PPM. SPSO performs the best, and GA performs the worst. However, when the sample variances are taken into account, does the ranking remain the same? See Fig. 15 for an answer.

it does not try to make use of the composition structure of the CEC-2014 test set, just like GA and SPSO.

Then the confidence upper bound matrix of DE8 are compared with the confidence lower bound matrices of the other DEs. Fig. 13 shows the resulted performance profiles. From Fig. 13, we see that DE8 performs better than 14 DEs except DE2, DE3, and DE9. Therefore, we can conclude that DE8 got the second best performance profile of the mean values on the CEC-2014 test set, and it performs statistically different from the other algorithms, except DE2, DE3, and DE9.

2) Comparison of DE8, GA, and SPSO: Fig. 14 displays the sample mean matrix with PPM. We can see that SPSO performs very well for almost any performance ratio or relative measure of computational budget, DE8 performs better than GA but worse than SPSO.

Fig. 15 displays the confidence upper bound matrix of SPSO and the confidence lower bound matrix of GA and DE8 with

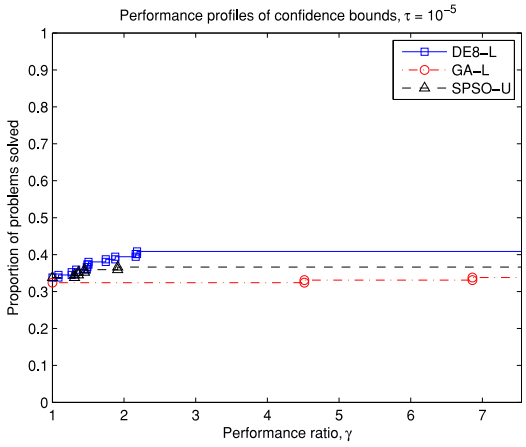


Fig. 15. Confidence upper bound matrix of SPSO together with the confidence lower bound matrices of DE8 and GA are displayed with PPM. DE8 performs better than SPSO, and GA performs worse than SPSO. Combined with Fig. 14, SPSO performs significantly better than GA, but its performance is not statistically different from DE8s.

TABLE I
RESULTS OF WILCOXON RANK SUM TESTS (HOW MANY FUNCTIONS THE ROW ALGORITHM WINS WHEN IT IS COMPARED WITH THE COLUMN ALGORITHM, TOTAL NUMBER OF FUNCTIONS IS 142, AND THE SIGNIFICANCE LEVEL IS 95%)

Algorithm	GA	DE	SPSO
GA	0	35	34
DE	96	0	42
SPSO	100	75	0

performance profiles. We can see from Fig. 15 that, DE8 performs better than SPSO, and SPSO performs better than GA. Therefore, combining with Fig. 14, we conclude that there is no significant difference between DE8 and SPSO while GA performs the worst. The supplementary material provides the comparison results by DPMS. We have tried other values for the precision level τ, and similar results are obtained. It seems that the conclusions given above are insensitive to τ.

In order to provide a comparison between our method and the significance-test-based method, we adopt the Wilcoxon rank sum test to analyze the same data. Table I shows how many functions the row algorithm wins when it is compared with the column algorithm. For instance, when we compare GA and DE, 142 Wilcoxon rank sum tests (one for each function) are performed. The results show that GA performs better than DE on 35 functions, while DE performs better than GA on 96 functions (they perform similarly on 11 functions). Similarly, SPSO performs better than GA on 100 functions while worse on 34 functions (they perform similarly on eight functions), and SPSO performs better than DE on 75 functions while worse on 42 functions (they perform similarly on 25 functions).

Compared with our method, the Wilcoxon test method provides similar results: DE performs better than GA while worse than SPSO. However, our method shows that DE performs worse than SPSO only in the sense of means, and it does not hold when variances are taken into account. The reasons may lie in the following two facts. First, the Wilcoxon test cares about the median but our method focuses on the mean.

Second, our method is often stricter than significant test-based methods which is implied in Proposition 1.

Although our method does not depend on nor make use of any distribution information of the population, it does not mean that the comparison results are always the same when different populations are adopted. The supplementary material (part A) provides two examples where different seeds are adopted when comparing DE8, GA, and SPSO. Results show that SPSO performs the best, which is different from previous results.

Finally, it is necessary to point out that these results are based on only 10 000 function evaluations, which is not enough to find the global optimum for most of the CEC-2014 problems. Moreover, when the computational budget increases, different results may appear. As we have mentioned at the beginning of this section, our main purpose is not to compare these three algorithms completely but to provide some real comparison examples with our benchmarking method. Further assessment of the performance of the algorithms under bigger budgets of function evaluation is not considered in this paper.

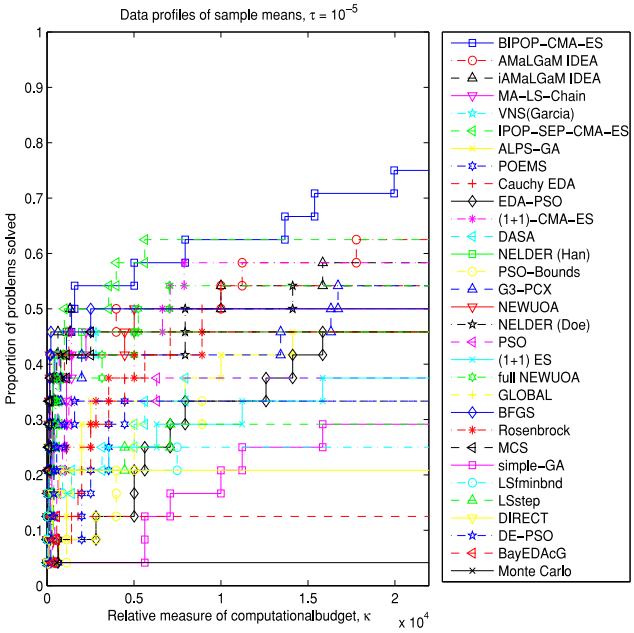


Fig. 16. Sample mean matrix generated from BBOB-2009 data is displayed with DPM. Similar results are obtained as those in [32], e.g., BIPOP-CMA-ES is the best algorithm while Monte Carlo is the worst one.

B. Application on the BBOB-2009 (10-D) Data

In this section, we apply our method to analyze the BBOB-2009 data [32], which can be downloaded from http://coco.gforge.inria.fr/doku.php?id=start. Moreover, for a direct comparison with Fig. 3, only the 10-D data is considered.

First, we have to point out that the BBOB-2009 data is not very suitable for our method because there are only 15 independent runs for each problem. In [32], an interesting simulation method was designed to generate 100 instances of runtime from these 15 trials. However, we do not know clearly whether such a method is free of pseudoreplication [46]–[49] or not. Therefore, we can only rely on the 15 independent trials. Since our method requires at least 30 independent runs, so the results below are not trustable sufficiently, although they are interesting.

The BBOB competition requires to find solutions very close to the global optimum, so large (even huge) number of function evaluations are needed. For instances, in the BBOB-2009 data, tens of millions of function evaluations are often needed. Thus the memory required maybe too large to be satisfied, since we need to deal with 3-D matrices. To make it simple, we generated three matrices ($\bar{H}$, H_{upper}, and H_{lower}) with size $200\,000\times24\times31$ from the BBOB-2009 data. In other words, a history of 200 000 best function values found during the optimization process was generated for each of the 31 algorithms on each of the 24 problems.

Fig. 16 shows the data profiles and the performance profiles of the sample means for these 31 algorithms, respectively. From Fig. 16, we can see that BIPOP-CMA-ES and IPOP-SEP-CMA-ES are the best two algorithms when the computational cost is less than about 140 000 function evaluations, and after that BIPOP-CMA-ES surpasses the latter and the other algorithms, too. BIPOP-CMA-ES, IPOP-SEP-CMA-ES, and AMal GaM IDEA are the best three algorithms which solve the most problems finally. Their proportions of solved problems are about 75%, 63%, and 63%, respectively. Similarly, Monte Carlo, BayEDAcG and DIRECT are the worst three algorithms. Their proportions are about 4%, 12%, and 21%, respectively. The performance profiles of the sample means for these 31 algorithms are provided in Fig. S4. For the limitation of paper length, Figs. S4–S6 below are provided in the supplementary material (part B).

Roughly speaking, the results obtained from the data profiles in Fig. 16 are similar to those obtained in Fig. 3. Especially, they reveal the same conclusion that which are the best algorithm (BIPOP-CMA-ES) and the worst algorithm (Monte Carlo). The other algorithms' ranking values are somewhat different, but generally similar. However, the performance profiles in Fig. S4 reveal more information for us. In [32], it was shown that both MCS and NEWUOA perform well on problems with low dimensions (2-D and 3-D). However, we have shown in Fig. S4 that they perform well even on larger dimension as long as the computational cost is not large. Such result is also supported in [53].

Then we take the sample variances into account to see whether BIPOP-CMA-ES performs significantly better than the other algorithms. Unfortunately, the answer is no. When the upper bound matrix of BIPOP-CMA-ES and the lower bound matrices of the other 30 algorithms are displayed through DPM, we can see from Fig. S5 that most profiles are low (below 20%), and BIPOP-CMA-ES cannot solve any problem, i.e., its profile is gone! Therefore, the result that BIPOP-CMA-ES performs better than all the other 30 algorithms is not significant statistically, at least under the significance level (95%, if the population is normally distributed) corresponding to $\lambda = 2$ in the confidence intervals (6).

Finally, we tried $\lambda = 1$ to generate the confidence intervals (6), whose confidence level is about 68% if the

population is normally distributed. Fig. S6 shows the data profiles, where the upper bound matrix of BIPOP-CMA-ES and the lower bound matrices of the other 30 algorithms are displayed. We can see that BIPOP-CMA-ES performs well in this case, better than all the other algorithms when the computational cost is larger than about 140 000. Thus we can only conclude in a moderate confidence level that BIPOP-CMA-ES performs significantly better than the other 30 algorithms.

VI. Conclusion

A distribution-free, confidence interval-based benchmarking method has been proposed to compare stochastic optimization algorithms. Through introducing a general confidence interval into DPM and PPM, the proposed method can compare stochastic algorithms with CDFs in graph. Moreover, both sample means and sample variances are considered in our method, and therefore it is better than some methods where only sample means are taken into account. Finally, our method is perfectly suitable for large set of benchmark problems.

The evolutionary computing community has witnessed impressive advances in the last decades, but the proper evaluation of the performance of numerous optimization algorithms remains an issue largely inappropriately addressed. We hope our distribution-free, confidence interval-based method can provide a formal utility for researchers to conduct large-scale benchmarking studies and draw sensible conclusions.

Acknowledgment

The authors would like to thank M. Clerc for the codes of SPSO 2011, and R. Storn for the DeMat codes.

References

[1] C. A. Floudas and C. E. Gounaris, "A review of recent advances in global optimization," *J. Glob. Optim.*, vol. 45, no. 1, pp. 3–38, 2009.

[2] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, 3rd ed. Berlin, Germany: Springer, 1996.

[3] T. Weise, *Global Optimization Algorithms—Theory and Applications*, 3nd ed. 2011. [Online]. Available: http://www.it-weise.de/projects/book.pdf

[4] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *J. Optim. Theory Appl.*, vol. 79, no. 1, pp. 157–181, 1993.

[5] D. E. Finkel and C. T. Kelley, "Additive scaling and the direct algorithm," *J. Glob. Optim.*, vol. 36, no. 4, pp. 597–608, 2006.

[6] W. Huyer and A. Neumaier, "Global optimization by multilevel coordinate search," *J. Glob. Optim.*, vol. 14, no. 4, pp. 331–355, 1999.

[7] Q. Liu and J. Zeng, "Global optimization by multilevel partition," *J. Glob. Optim.*, vol. 61, no. 1, pp. 47–69, 2015.

[8] Q. Liu, J. Zeng, and G. Yang, "MrDIRECT: A multilevel robust direct algorithm for global optimization problems," *J. Glob. Optim.*, vol. 62, no. 2, pp. 205–227, 2015.

[9] G. Liuzzi, S. Lucidi, and V. Piccialli, "A partition-based global optimization algorithm," *J. Glob. Optim.*, vol. 48, no. 1, pp. 113–128, 2010.

[10] W.-N. Chen *et al.*, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.

[11] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimization," *Inf. Sci.*, vol. 381, pp. 142–160, Mar. 2017.

[12] Q. Yang *et al.*, "Adaptive multimodal continuous ant colony optimization," *IEEE Trans. Evol. Comput.*, to be published.

[13] Q. Yang *et al.*, "Multimodal estimation of distribution algorithms," *IEEE Trans. Cybern.*, to be published.

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. Cambridge, MA, USA: MIT Press, 1992.

[15] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. IEEE 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, 1995, pp. 39–43.

[16] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. IV*, Piscataway Township, NJ, USA, 1995, pp. 1942–1948.

[17] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[18] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[19] M. Jiang, Y. P. Luo, and S. Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Inf. Process. Lett.*, vol. 102, no. 1, pp. 8–16, 2007.

[20] Q. Liu, "Order-2 stability analysis of particle swarm optimization," *Evol. Comput.*, vol. 23, no. 2, pp. 187–216, Jun. 2015.

[21] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317–325, 2003.

[22] W.-N. Chen *et al.*, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[23] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[24] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[25] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[26] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session and competition on real-parameter optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China and Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, Jan. 2013.

[27] E. Brandstätter, "Confidence intervals as an alternative to significance testing," *Methods Psychol. Res. Online*, vol. 4, no. 2, pp. 33–46, 1999.

[28] D. H. Johnson, "The insignificance of statistical significance testing," *J. Wildlife Manag.*, vol. 63, no. 3, pp. 763–772, 1999.

[29] R. S. Nickerson, "Null hypothesis significance testing: A review of an old and continuing controversy," *Psychol. Methods*, vol. 5, no. 2, pp. 241–301, 2000.

[30] J. S. Armstrong, "Significance tests harm progress in forecasting," *Int. J. Forcasting*, vol. 23, no. 2, pp. 321–327, 2007.

[31] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[32] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009," in *Proc. 12th Annu. Conf. Companion Genet. Evol. Comput.*, Portland, OR, USA, 2010, pp. 1689–1696.

[33] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, 2002.

[34] J. Moré and S. Wild, "Benchmarking derivative-free optimization algorithms," *SIAM J. Optim.*, vol. 20, no. 1, pp. 172–191, 2009.

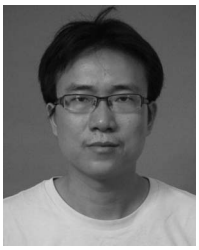
[35] J. Vesterstrøm and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, Portland, OR, USA, 2004, pp. 1980–1987.

[36] Y.-J. Gong *et al.*, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[37] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2238–2251, Oct. 2016.

[38] Q. Yang *et al.*, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, to be published.

[39] W.-J. Yu *et al.*, "Differential evolution with two-level parameter adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080–1099, Jul. 2014.
[40] S. Ahn, S. H. Park, and K. H. Lee, "How to demonstrate similarity by using noninferiority and equivalence statistical testing in radiology research," *Radiology*, vol. 267, no. 2, pp. 328–338, 2013.
[41] A. LaTorre, S. Muelas, and J.-M. Peña, "A comprehensive comparison of large scale global optimizers," *Inf. Sci.*, vol. 316, pp. 517–549, Sep. 2015.
[42] C. García-Martínez, F. J. Rodriguez, and M. Lozano, "Arbitrary function optimisation with metaheuristics: No free lunch and real-world problems," *Soft Comput.*, vol. 16, no. 12, pp. 2115–2133, 2012.
[43] C. García-Martínez, F. Glover, F. J. Rodriguez, M. Lozano, and R. Martí, "Strategic oscillation for the quadratic multiple knapsack problem," *Comput. Optim. Appl.*, vol. 58, no. 1, pp. 161–185, 2014.
[44] A. R. Conn, K. Scheinberg, and L. N. Vincente, *Introduction to Derivative-Free Optimization*. Philadelphia, PA, USA: SIAM, 2009.
[45] M. Taboga, *Lectures on Probability Theory and Mathematical Statistics*, 2nd ed. Lexington, KY, USA: Amazon CreateSpace, 2012.
[46] M. Birattari, "On the estimation of the expected performance of a metaheuristic on a class of instances: How many instances, how many runs?" Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2004-001, 2004.
[47] S. E. Lazic, "The problem of pseudoreplication in neuroscientific studies: Is it affecting your analysis?" *BMC Neurosci.*, vol. 11, no. 5, pp. 397–407, 2004.
[48] R. B. Millar and M. J. Anderson, "Remedies for pseudoreplication," *Fisheries Res.*, vol. 70, no. 2, pp. 397–407, 2004.
[49] J. C. Schank and T. J. Koehnle, "Pseudoreplication is a pseudoproblem," *J. Comp. Psychol.*, vol. 123, no. 4, pp. 421–433, 2009.
[50] K. Price, R. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Heidelberg, Germany: Springer, 2006.
[51] M. Clerc. (2011). *Standard Particle Swarm Optimization: From 2006 to 2011*. [Online]. Available: http://clerc.maurice.free.fr/pso/
[52] J. J. Liang, "Novel particle swarm optimizers with hybrid, dynamic & adaptive neighborhood structures," Ph.D. dissertation, School Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, 2008.
[53] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: A review of algorithms and comparison of software implementations," *J. Glob. Optim.*, vol. 56, no. 3, pp. 1247–1293, 2013.



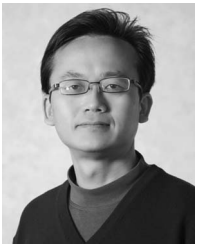
Qunfeng Liu received the Bachelor's degree and the M.S. degree from the Huazhong University of Science and Technology, in 1999 and 2002, respectively, and the Ph.D. degree from Hunan University, Changsha, China, in 2011.

He is currently an Associate Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. His current research interests include global optimization and swarm intelligence algorithms.

Wei-Neng Chen (S'07–M'12) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has published 60 papers in international journals and conferences. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society Outstanding Dissertation Award in 2016, and also the National Science Fund for Excellent Young Scholars in 2016.



Jeremiah D. Deng (M'01) received the B.Eng. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 1989, and the M.Eng. and D.Eng. degrees from the South China University of Technology (SCUT), Guangzhou, China, in 1992 and 1995, respectively.

He was a Research Assistant with the Department of Computer Science, University of Hong Kong, Hong Kong, from 1993 to 1995. He joined SCUT, as a Lecturer, in 1995. He later joined the University of Otago, Dunedin, New Zealand, as a Post-Doctoral Research Fellow, in 1999, where he is currently an Associate Professor with the Department of Information Science. He has been a Visiting Professor with Sun Yat-sen University, Guangzhou. He has published over 100 refereed research papers in international journals and conference proceedings. His current research interests include machine learning, pattern recognition, and stochastic modeling and optimization of computer networks.

Tianlong Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Bentley, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth, WA, Australia. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Huaxiang Zhang received the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 2004.

He was an Associated Professor with the Department of Computer Science, Shandong Normal University, Jinan, China, from 2004 to 2005, where he is currently a Professor with the School of Information Science and Engineering. He has authored over 100 journal and conference papers and holds eight invention patents. His current research interests include machine learning, pattern recognition, evolutionary computation, and Web information processing.

Zhengtao Yu received the Ph.D. degree in computer application technology from the Beijing Institute of Technology, Beijing, China, in 2005.

He is currently a Professor with the School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. His current research interests include natural language processing, information retrieval, and machine learning.

Jun Zhang (M'02–SM'08–F'16) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Professor with the South China University of Technology, Guangzhou, China. He was also appointed as the Changjiang Chair Professor in 2013. His current research interests include computational intelligence, cloud computing, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 50 IEEE TRANSACTIONS papers in the above areas.

Prof. Zhang was a recipient of the National Science Fund for Distinguished Young Scholars in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection and the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters. He is the Founding and Current Chair of ACM Guangzhou Chapter.