Paradoxes in Numerical Comparison of Optimization Algorithms

Qunfeng Liu, William V. Gehrlein, Ling Wang, Yuan Yan, Yingying Cao, Wei Chen, and Yun Li, *Fellow, IEEE*

***Abstract*—Numerical comparison is often key to verifying the performance of optimization algorithms, especially, global optimization algorithms. However, studies have so far neglected issues concerning comparison strategies necessary to rank optimization algorithms properly. To fill this gap for the first time, we combine voting theory and numerical comparison research areas, which have been disjoint so far, and thus extend the results of the former to the latter for optimization algorithms. In particular, we investigate compatibility issues arising from comparing two and more than two algorithms, termed "C2" and "C2+" in this article, respectively. Through defining and modeling "C2" and "C2+" mathematically, it is uncovered and illustrated that numerical comparison can be incompatible. Further, two possible paradoxes, namely, "cycle ranking" and "survival of the nonfittest," are discovered and analyzed rigorously. The occurrence probabilities of these two paradoxes are also calculated under the no-free-lunch assumption, which shows the first justifiable use of the impartial culture assumption from voting theory, providing a point of reference to the frequency of the paradoxes occurring. It is also shown that significant influence on these probabilities comes from the number of algorithms and the number of optimization problems studied in the comparison. Further, various limiting probabilities when the number of optimization problems goes to infinity are also derived and characterized. The results would help guide benchmarking and developing optimization and machine learning algorithms.**

***Index Terms*—Cycle ranking, evolutionary computation, machine learning, numerical comparison, optimization.**

Manuscript received December 22, 2018; revised April 27, 2019 and August 23, 2019; accepted November 12, 2019. Date of publication November 22, 2019; date of current version July 30, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFD0400206, in part by the National Natural Science Fund for Distinguished Young Scholars of China under Grant 61525304, in part by the National Natural Science Foundation of China under Grant 61773119 and Grant 61873328, and in part by the Dongguan University of Technology under Grant KCYKYQD2017014 and Grant DGUT(Q)-GGB-2016005. *(Corresponding author: Yun Li.)*

Q. Liu, Y. Yan, Y. Cao, W. Chen, and Y. Li are with the Dongguan Industry 4.0 Artificial Intelligence Laboratory, Dongguan University of Technology, Dongguan 523808, China, and also with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523808, China (e-mail: yun.li@ieee.org).

W. V. Gehrlein is with the Department of Business Administration, University of Delaware, Newark, DE 19716 USA (e-mail: wvg@udel.edu).

L. Wang is with the Department of Automation, Tsinghua University, Beijing 100083, China.

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

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

Digital Object Identifier 10.1109/TEVC.2019.2955110

I. Introduction

NUMERICAL comparison has been widely used in testing and verifying the performance of optimization algorithms, especially, global optimization algorithms such as evolutionary algorithms [1]–[4]. Despite that a number of comparison methods and amalgamated scoring systems [5] exist, few studies have been reported on comparison strategies so far. In this article, we study comparison strategies and their mathematical properties. In particular, the compatibility of different comparisons results by two popular comparison strategies will be investigated in full. This article will cover numerical comparison of algorithms on multiple optimization problems.

As several matches of numerical tests can be included in a comparison, a performance metric is needed to indicate an algorithm's performance rank in each match through an appropriate comparison method. Fig. 1 shows the process of the numerical comparison of optimization algorithms, which requires numerical experiments (selecting optimization algorithms and problems, and then testing the algorithms on these problems), determination of a comparison strategy and a comparison method, analyzing metric data and obtaining comparison results. Following the comparison endeavoring, declaration of a winner (or the "best") algorithm is often desired in an algorithm competition [4], [6], [7] or algorithm development [8], [9]. The next section will outline more background on this aspect.

Extensive studies have been undertaken to evaluate how to compare or benchmark optimization algorithms numerically, especially, on the design of test or benchmark problems and the development of comparison methods [6], [7], [10], [11]. So far, however, there exist few reports on a comparison strategy. In essence, several "matches" are often included in the entire comparison, and a comparison strategy is about how many matches are needed to complete the entire comparison and which algorithms need to be compared at each match. Conversely, a comparison method deals with the analysis in a single match. In other words, a comparison strategy tries to obtain algorithms' performance ranking when the entire comparison finishes, whilst a comparison method seeks to obtain the performance rankings in a single match.

In [4], a specially designed methodology called exploratory landscape analysis was proposed to obtain a "winner" or best algorithm from the black-box optimization benchmarking (BBOB), COmparing continuous optimizers

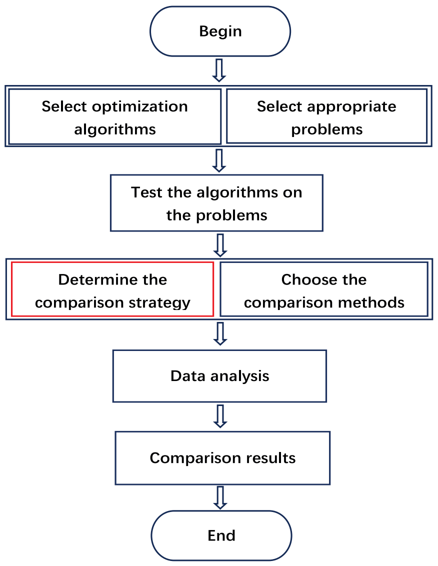

Begin
Select optimization algorithms
Select appropriate problems
Test the algorithms on the problems
Determine the comparison strategy
Choose the comparison methods
Data analysis
Comparison results
End

Fig. 1. Systematic framework and flow chart for the numerical comparison of optimization algorithms. The red box highlights the focus of this article, verifying the compatibility of comparison results to be obtained from different comparison strategies.

(COCO) 2009–2010 comparison results [6], [12]. It was concluded that it is not easy to aggregate rankings from algorithm evaluations on individual problems into a consensus, and some common pitfalls exist.

In this article, therefore, we dedicate our investigations to the comparison strategy, especially, the compatibility of different comparison results from different strategies. Two basic comparison strategies are to be mathematically defined and modeled, and two probable paradoxes are to be illustrated. The occurrence probabilities of these paradoxes are calculated and their limiting values are provided to illustrate possibly incompatible comparison results from different strategies and that the number of algorithms and problems considered can influence the results significantly.

The remainder of this article is organized as follows. In Section II, the flow chart of Fig. 1 is discussed in detail, where the comparison method and the comparison strategy are reviewed after summarizing the necessity of numerical comparison. In Section III, two popular comparison strategies are modeled mathematically to enhance analysis. Based on the model, probable paradoxes are illustrated in Section IV, and the probabilities of the paradoxes are calculated in Section V. Finally, the conclusions are summarized in Section VI.

II. Numerical Comparisons of Optimization Algorithms

Without loss of generality, many optimization algorithms have been developed to solve the minimization problem

$$\min\ f(t), \quad t \in \Omega \subseteq \mathbb{R}^n \tag{1}$$

where $f(t)$ is the objective function and t is a vector containing n decision variables. A maximization problem can be modeled as the above minimization problem through, for example, minus $f(t)$. When $\Omega = \mathbb{R}^n$, problem (1) is unconstrained, otherwise it is constrained.

When $f(t)$ is nonconvex, problem (1) is often hard to solve. Therefore, in the mathematical programming community, a local optimal point $\hat{t}$ satisfying

$$f(\hat{t}) \le f(t) \quad \forall t \in (\hat{t} - \delta,\ \hat{t} + \delta) \quad \forall \delta > 0 \tag{2}$$

is often sought, where gradient information regarding $f(t)$ is often necessary in algorithm design and analysis. However, in the evolutionary computation community and the global optimization community, the global optimal point $t^\star$ satisfying

$$f(t^\star) \le f(t) \quad \forall t \in \Omega \tag{3}$$

is often sought and investigated, regardless of the availability of gradient information.

With either local optimization or global optimization, it is often necessary to compare different algorithms' numerical performance. To do this, certain algorithms are first selected and then tested on certain optimization problems. Both a proper comparison strategy and a comparison tactic, i.e., method, are needed in analyzing the gathered test data.

In order to investigate the compatibility of different comparison results from different comparison strategies, we first describe what numerical comparison of optimization algorithms is and what the role of comparison strategy is. In this section, we shall show why numerical comparison of optimization algorithms is necessary, and shall describe its main components (Fig. 1), namely, specifying algorithms and test problems, numerical experiments, comparison strategies, comparison methods, and data analysis and comparison results.

A. Why Numerical Comparison

There exist a large number of optimization algorithms developed for various kinds of optimization problems. However, which algorithm is the best on a certain given problem is often unclear or inconclusive [4], [13], [14]. Numerical comparison can bring helpful insight to the users through verifying the performance of optimization algorithms, whether being local or global.

For local optimization algorithms, they are often compared theoretically through analyzing the convergence and the convergence rate. The higher the convergence rate is, the better the algorithm is. However, a higher convergence rate does not imply a shorter CPU time. Moreover, convergence is a property under limit conditions which may be inaccessible in practice. Therefore, numerical comparison is necessary to verify local optimization algorithms' performance before reaching the limit.

For global optimization algorithms, the convergence is hard to prove due to the absence of applicable mathematical optimal conditions. Therefore, numerical comparison is often the only way to verify the efficacy of a global optimization algorithm.

In summary, numerical comparison is necessary for both local and global optimization algorithms' design, analysis, and development. Key components of the framework for numerical comparison are detailed as follows.

B. Specifying Algorithms and Test Problems

From Fig. 1, we can see that the selection of algorithms and problems are the preconditions of numerical experiments (i.e., testing the algorithms on the problems).

The selection of optimization algorithms often depends on the specific implementations. There are two popular implementations: 1) algorithm competition and 2) development of a new algorithm. For algorithm competition, algorithms are often selected if their developers decide to attend the competition. For the development of a new algorithm, the algorithm is often selected by the developer of the new algorithm, and a common practice is to select certain existing good algorithms of the same type as the proposed new algorithm.

Numerous test problems, including benchmarking functions [6], [10], [15] and practical test problems [16]–[18] have been designed or modeled for numerical comparison of optimization algorithms. There are some popular sets of benchmark functions, for example, the series of conference of evolutionary computation (CEC) benchmark sets [7], [20], and series of BBOB sets [12].

However, there is still no comprehensive or systematic research on how to select proper optimization problems for numerical comparison. Without this guide, it is preferred to select the whole set of a certain popular optimization problem suite, and not part of it. Moreover, in this article, we shall show that the number of optimization problems and its parity have a significant impact on the final comparison result.

C. Numerical Tests

In this phase, each selected algorithm will be adopted to solve each selected problem. If the algorithm is nondeterministic or stochastic, then multiple independent runs are required to alleviate random variances as far as possible. For local optimization, numerical experiments often stop when a mathematical optimal condition is satisfied. However, there is no suitable mathematical optimal condition for global optimization. Therefore, numerical experiments often stop after a given computational budget is consumed.

After all the experiments finish, the large amounts of data gathered undergo post analysis. Although the data contain several metrics, such as the function values and the CPU times, the most popular metric is the series of the best solutions found as the computational cost increases.

It is observed that numerical experiments are conducted almost independently from later selections of either a comparison strategy or a comparison method. In effect, whatever the comparison strategy or the comparison method is chosen, the same implementations of numerical experiments described above are often needed. The only differences are the selection of performance metrics, e.g., the found best function values or the CPU times, and the determination of parameters, such as the computational budget and the number of dependent runs for stochastic optimization algorithms. Although these selections may have a high influence on comparison results, they do not affect the analysis on the compatibility of different comparison strategies.

D. Comparison Strategies

A comparison strategy is meaningful when three or more optimization algorithms need to be compared. In this case, several matches may be included in the comparison. A comparison strategy determines how many matches are needed and which algorithms are compared in each match. However, a comparison strategy is often omitted or overlooked partly because it is often considered as a default setting when choosing an appropriate comparison method in each match. A more important reason is that, to the best of our knowledge, there exists no research in the optimization community to inform what comparison results from different comparison strategies may be incompatible.

When three or more algorithms are compared, there exist two basic comparison strategies, namely, comparing by pairwise evaluation on each problem and comparing three or more algorithms on each problem. In this article, we term them a "C2" strategy and a "C2+" strategy, respectively, where "C" stands for "comparing" and "2+" stands for "more than 2." Although there are several choices for the "C2+" strategy, the most popular choice in practice is to compare all algorithms on each problem. Therefore, in the latter of this article, "C2+" is a strategy in which all participating algorithms are compared on each problem, unless otherwise specified.

Both "C2" and "C2+" are popular in the optimization community, e.g., the "C2" strategy is adopted in algorithm design or development in [9] and [21]–[25] and in the CEC algorithm competition [20], while "C2+" is adopted in other algorithm competitions, such as the BBOB COCO [6], [12], the IOHprofiler [26], and the black-box optimization competition (BBComp) [27], and also in algorithm design or development in [11] and [28]–[32].

There exist two popular implementations of the "C2" strategy, namely, the one-plays-all comparison and the all-play-all comparison, both comparing two players at one time. One-plays-all is often applied in the development of a new algorithm, where the main concern is the determination of whether or not the proposed algorithm performs better than existing similar algorithms [9], [21], [24], [32]–[34]. All-play-all is often called a round-robin comparison, i.e., it is a round one-plays-all, usually applied in algorithm competition [15]. If there are k algorithms, then $k-1$ matches are needed to finish a one-plays-all comparison, while $k(k-1)/2$ matches are needed to complete an all-play-all comparison.

Fig. 2 shows a diagram of the "C2+" strategy where three algorithms A, B, and C are compared by their data profiles [11], [31]. Specifically, as the increase of computational cost, the proportions of problems solved by these algorithms are displayed by cumulative distribution functions. This strategy compares all the algorithms at a single match or a few matches. Therefore, it is popular in algorithm competition [6], [26], [27], [35], [36] and the development of new algorithms [29], [30], [37].

It is clear that "C2" often needs many more matches than "C2+" to finish the whole comparison. Besides that, the most important difference between them is that "C2+" often adopts a certain kind of statistical aggregation method, such as

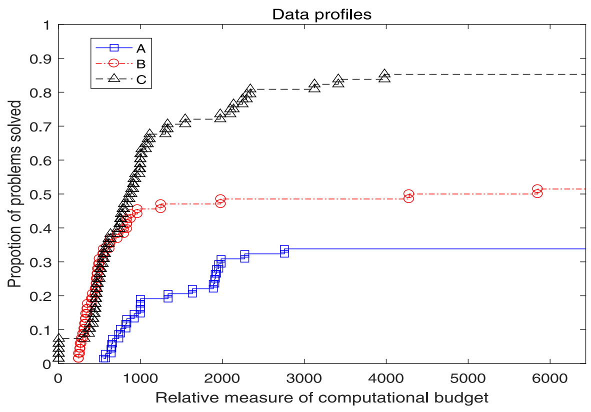


Fig. 2. Example of the "C2+" strategy, where three algorithms A, B, and C are compared simultaneously [11].

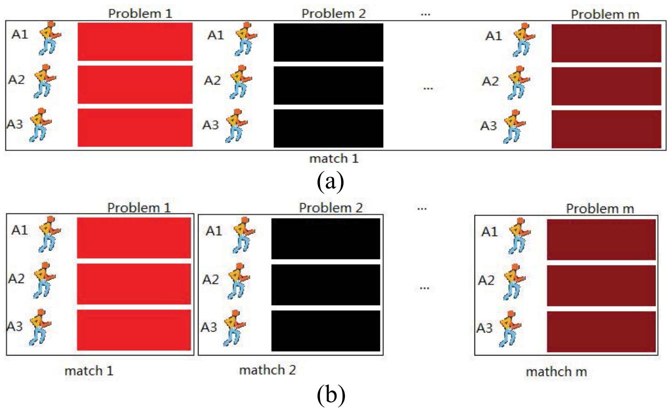


Fig. 3. Two "C2+" strategies when comparing three algorithms A_1, A_2, and A_3 on m optimization problems are illustrated under the analogy of running competitions. (a) In the upper figure, only one single match is included in the entire comparison. Comparison methods which can deal with three or more algorithms and several problems simultaneously are allowable, e.g., the performance profiles [28], the data profiles [31], and the visualizing confidence intervals [11]. (b) In the lower figure, m matches are included. Comparison methods which can deal with three or more algorithms but only one single problem is allowable, e.g., analysis of variance [38]. In practice, the strategy in (a) is much more popular than that in (b).

the cumulative distribution function, to obtain all algorithms' ranking results directly. On the contrary, "C2" has to obtain ranking in each match first and then aggregate them to obtain a final ranking.

In order to make clearer the relationships of the above concepts, e.g., a comparison, a match, a comparison strategy, and a comparison method, it is interesting to consider the numerical comparison of optimization algorithms as running competitions. Under this analogy, optimization algorithms are regarded as runners and each optimization problem corresponds to a competition. Since there are several optimization problems, therefore, the runners have to compete several times.

In Fig. 3, two "C2+" strategies when comparing three algorithms A_1, A_2, and A_3 on m optimization problems are illustrated under the analogy of running competitions, where 1 match shown in Fig. 3(a) and m matches shown in Fig. 3(b) are contained in the entire comparison, respectively. Three algorithms are compared on each problem. Similarly, in Fig. 4, two "C2" strategies are illustrated under the same analogy, where

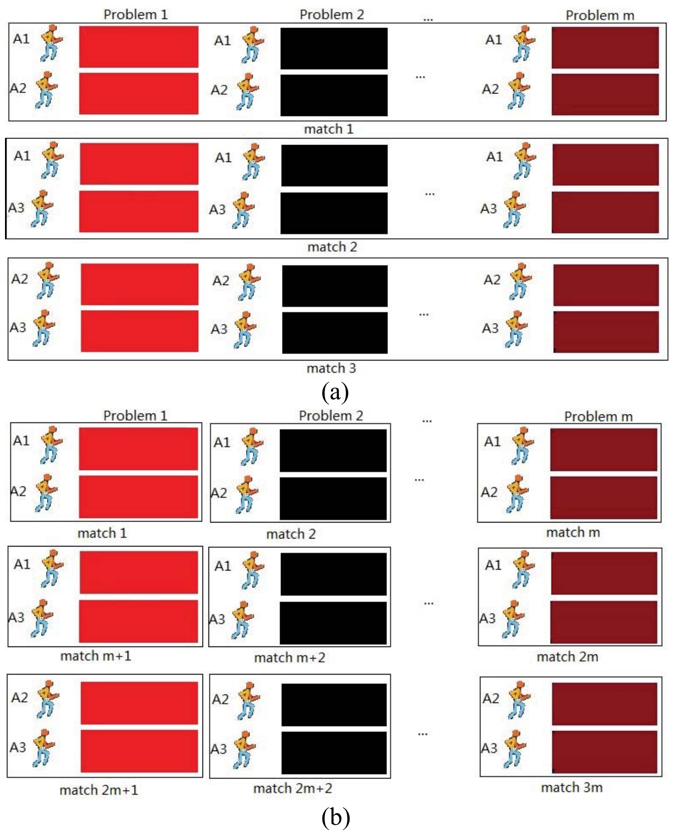


Fig. 4. Two "C2" strategies when comparing three algorithms A_1, A_2, and A_3 on m optimization problems are illustrated under the analogy of competitive running. (a) In the upper figure, three matches are included in the entire comparison. Comparison methods which can deal with two algorithms and several problems are allowable, e.g., the performance profiles [28], the data profiles [31], and the visualizing confidence intervals [11]. (b) In the lower figure, $3m$ matches are included. Comparison methods which deal with two algorithms and only one single problem are allowable, e.g., Student's t-test and the Wilcoxon's rank-sum test [39].

the entire comparison includes 3 and $3m$ matches, respectively. However, only two algorithms are compared on each problem. In both Figs. 3 and 4, possible comparison methods are provided in the captions.

E. Comparison Methods

After the comparison strategy is determined, a proper comparison method can be chosen for each single match. Many methods for analyzing the experimental data gathered during the experiments have been developed. We can classify them into three kinds of methods, namely, static comparison, dynamic ranking, and dynamic comparison by cumulative distribution functions, whose main characters are provided below, referring to [11] for more details.

Static comparison methods often care about the final state of numerical experiments, e.g., the best function values found. The means and standard deviations are often calculated [21], [22], and certain statistical inferences, e.g., student's t-test and Wilcoxon's rank-sum test [39], are often applied [9], [25]. In BBComp [27], the static comparison method is adopted, however, only one single run is allowed even for stochastic algorithms.

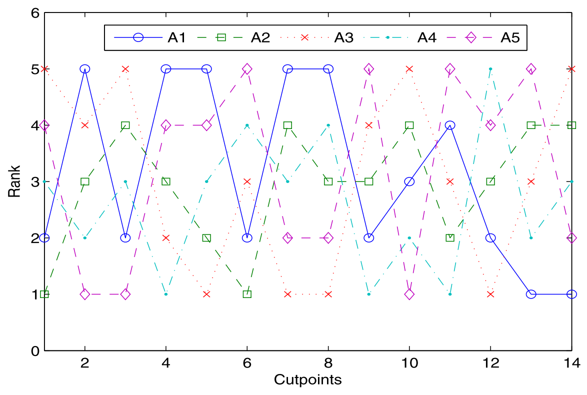


Fig. 5. Illustration of the ranking results of the CEC algorithm competition, where five algorithms are compared and 14 groups of rankings are provided according to different computational costs [11].

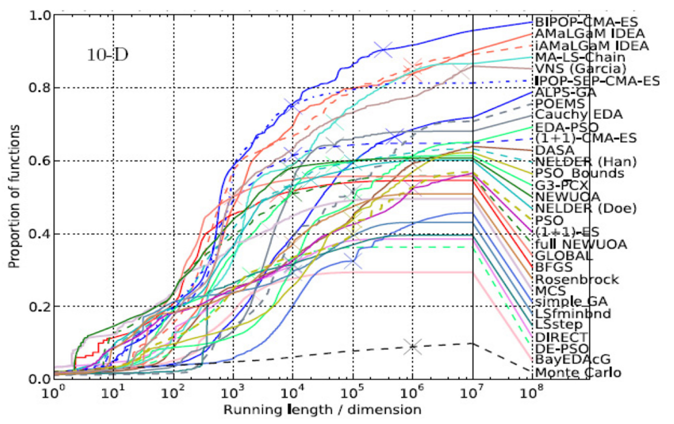


Fig. 6. Comparison results of the BBOB COCO 2009 algorithms competition, where 31 algorithms are compared in a single match [6].

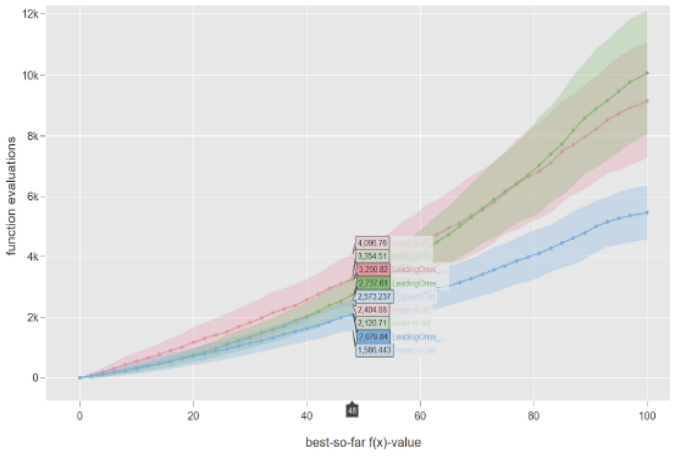


Fig. 7. Example of the comparison results of the IOHprofiler algorithms competition [26].

Dynamic ranking methods tend to obtain more information about the process states. A popular method is to display the history of the best function values found for each problem [23], [24]. Another way is to rank several (or many) times during the process [15], [40], in other words, carrying out the static comparison many times. For instance, In the CEC algorithm competition [15], 14 static comparisons were implemented to obtain dynamic rankings (see Fig. 5 for an illustration).

Dynamic ranking is not suitable when the number of problems is large since it is often carried out for each problem. The third kind of method overcomes this issue through employing cumulative distribution functions. Specifically, the proportion of problems solved by different algorithms are cumulated from 0 to 1 according to the increase of the computational cost (see Fig. 2 for an example).

Since the third kind of methods are convenient for comparing several algorithms simultaneously, they become more and more popular in the optimization community. For instance, the performance profiles method [28], [29] and the data profiles method [30], [31] for comparing deterministic optimization algorithms are almost necessary in mathematical programming community (see Fig. 2). Recently, such a technique has been extended to comparing stochastic optimization algorithms [11], [26], [32], [41]. Comparing the average behaviors [32], [41] is a direct application. In BBOB COCO [41], 100 test instances were averaged for each problem and the average behaviors were compared (Fig. 6), just like the data profiles [31] for deterministic algorithms. More techniques were developed to deal with stochastic errors, e.g., visualizing confidence intervals [11], [26]. In IOHprofiler [26], both sample means and confidence bounds are displayed in a single figure (see Fig. 7 for an example). However, two figures are needed to display them sequentially in [11].

F. Data Analysis and Comparison Result

After the comparison strategy and the comparison methods are determined, they are applied in the data analysis stage on the experimental data obtained. The main task of this stage is to complete the various calculations needed for the comparison methods. The outcome of these calculations is the comparison result, which includes certain ranks. Within these ranks, the most important information is which algorithm is the winner.

It is natural to seek a winner in an algorithm competition. In algorithm designing or development, a winner is also desired from numerical comparisons since it can provide information about which operator in the algorithm works better. Such information is helpful in further improving this algorithm or even this kind of algorithms.

III. Modeling Comparison Strategies

When the numerical experiments are finished, the experimental data become an objective existence. Since both “C2” and “C2+” are popular strategies in analyzing the experimental data, it is of interest to investigate whether their comparison results are always compatible. If not, what makes them incompatible? Further, what is the occurrence probability of incompatibility?

In order to answer these questions, in this section, we model both “C2” and “C2+” strategies mathematically to pursue rigorous analysis. To start, we first model the comparison of optimization algorithms with voting, as in an election.

A. Voting: Analogy to Comparison

The comparison of optimization algorithms is the same as an election in the sense of a mathematical model. Here, the algorithms and the problems are regarded as the candidates and the voters, respectively.

Now, the "selection" of candidates and voters is an analogy of the selection of algorithms and problems, and the voting process is an analogy of the numerical experiments. The comparison method is thus reduced to a counting method. However, a comparison strategy is still important in an election, and both "C2" and "C2+" are also popular in practice.

Under this analogy, it is natural and helpful to adopt the existing voting theory and its results to analyze our questions.

B. Mathematical Models of the Comparison Strategies

Although different comparison methods are allowable for applying the comparison strategy, only two ranking results are possible in any match of two algorithms $A_1, A_2 : A_1 \succeq A_2$ (implying A_1 does not perform worse than A_2) or $A_2 \succeq A_1$ (A_1 does not perform better than A_2). In numerical optimization, the best objective function values found are often employed to determine which algorithm performs better. For convenience in later discussions, the following definition is introduced.

Definition 1: Given a problem P, a fixed computational cost and allowable accuracy, suppose that $f^i_{\min}$ is the found optimality, or the minimal objective function value, by algorithm $A_i, i = 1, 2$. Then A_1 is said to be no worse than A_2 on P, i.e., $A_1 \succeq A_2$, if and only if $f^1_{\min} \leq f^2_{\min}$. Moreover, $A_1 \succ A_2$ if and only if $f^1_{\min} < f^2_{\min}$, and $A_1 = A_2$ if and only if $f^1_{\min} = f^2_{\min}$.

Definition 1 can be easily extended to more general cases. Suppose there are k algorithms $A_i, i = 1, \dots, k$. After adopting them to solving any given problem, k best function values $f^i_{\min}$ are found (for any fixed computational budget), where $f^i_{\min}$ is the best function value found by $A_i, i = 1, \dots, k$. Through comparing these function values, a ranking

$$A_{i_1} \succeq A_{i_2} \succeq \cdots \succeq A_{i_k} \tag{4}$$

is obtained, where $i_1, \dots, i_k$ is a permutation of $1, \dots, k$, and the relationship $\succeq$ is defined in Definition 1.

Note that a different ranking in (4) may occur on a different problem, different run or different computational budget. In theory, all $k!$ rankings are possible. Therefore, it can be regarded as random sampling from a multinomial distribution for which m optimization problems are tested. For convenience, such a sampling can be expressed as the following matrix:

$$\begin{bmatrix} p_1 & \cdots & p_{k!} \\ x_1 & \cdots & x_{k!} \\ A_1 & \cdots & A_k \\ A_2 & \cdots & A_{k-1} \\ \cdots & \cdots & \cdots \\ A_k & \cdots & A_1 \end{bmatrix}_{(k+2)\times k!} \tag{5}$$

where $p_i \in (0, 1)$ and $x_i \in [0, m]$ for $i = 1, \dots, k!$ and satisfy

$$\sum_{i=1}^{k!} p_i = 1, \quad \sum_{i=1}^{k!} x_i = m. \tag{6}$$

Model (5) is a probabilistic model of "C2+," where the first column implies that $A_1 \succeq A_2 \succeq \cdots \succeq A_k$ occurs with a probability p_1 on each problem, actually occurring on x_1 among m problems, and the rest is similar. In other words, the random vector $x = (x_1, \dots, x_{k!})^{\mathrm{T}}$ satisfies the multinomial distribution with parameters m and $p = (p_1, \dots, p_{k!})$.

The winner of the probabilistic model (5) is determined by the following plurality rule, which is also called the relative majority rule and is very popular in numerical comparisons of optimization algorithms [6], [9], [11], [21], [24].

Assumption 1 (Plurality Rule): An algorithm performs better than the other algorithms if it performs better on more problems than the others do.

Specifically, according to the probabilistic model (5), if $x_i \geq x_j, j = 1, 2, \dots, k!$, then the algorithm lies in the ith column and the third row is the winner of the comparison.

It is not difficult to deduce a probabilistic model of "C2" for any pair of algorithms from model (5) of "C2+." For instance, if we consider the comparison of A_1 and A_2, then the model can be expressed as follows:

$$\begin{bmatrix} q & 1-q \\ Y & m-Y \\ A_1 & A_2 \\ A_2 & A_1 \end{bmatrix} \tag{7}$$

where q and Y are the sum of p_i and x_i, respectively, for those i with which $A_1 \succeq A_2$ occurs at the ith column. It is clear that Y satisfies the binomial distribution $B(m, q)$.

Since (5) is the probabilistic model of "C2+" and includes implicitly all the models of "C2," it is convenient for our purposes of rigorously studying numerical comparison strategies. For example, in (7), we say $A_1 \succeq A_2$ if and only if $Y \geq m-Y$.

IV. Ranking and Survival Paradoxes

In this section, we adopt the probabilistic model (5) to illustrate two paradoxes. For simplicity, only three algorithms (A_1, A_2, and A_3) are considered in this section, and it is enough for our purpose. In this case, there are $3! = 6$ possible rankings.

A. "Cycle Ranking" Paradox: From "C2"

The following matrix is an example of the probabilistic model (5), where three algorithms are tested on 25 problems and the parameters p_i satisfy condition (6):

$$\mathcal{M} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\ 0 & 5 & 10 & 0 & 5 & 5 \\ A_1 & A_1 & A_2 & A_2 & A_3 & A_3 \\ A_2 & A_3 & A_1 & A_3 & A_1 & A_2 \\ A_3 & A_2 & A_3 & A_1 & A_2 & A_1 \end{bmatrix}. \tag{8}$$

The second column of $\mathcal{M}$ implies that $A_1 \succeq A_3 \succeq A_2$ occurs on five problems, and the rest is similar.

If we adopt the "C2" strategy to compare these three algorithms, then the following three matrixes can be derived from

$\mathcal{M}$:

$$\begin{bmatrix} p_{12} & 1-p_{12} \\ 10 & 15 \\ A_1 & A_2 \\ A_2 & A_1 \end{bmatrix}, \begin{bmatrix} p_{13} & 1-p_{13} \\ 15 & 10 \\ A_1 & A_3 \\ A_3 & A_1 \end{bmatrix}, \begin{bmatrix} p_{23} & 1-p_{23} \\ 10 & 15 \\ A_2 & A_3 \\ A_3 & A_2 \end{bmatrix} \tag{9}$$

where $p_{12} = p_1 + p_2 + p_5, p_{13} = p_1 + p_2 + p_3, p_{23} = p_1 + p_3 + p_4$, and which are the probabilistic models of "C2" when comparing (A_1, A_2), (A_1, A_3), and (A_2, A_3), respectively.

From the first matrix of (9) we conclude that $A_2 \succeq A_1$ since A_2 performs better than A_1 on more problems ($15 > 10$). Similarly, the left matrices imply that $A_1 \succeq A_3$ and $A_3 \succeq A_2$, respectively. Thus, the comparison results $A_2 \succeq A_1$, $A_1 \succeq A_3$, and $A_3 \succeq A_2$ form a cycle, and we cannot tell which algorithm performs the best. This phenomenon is termed a "cycle ranking" paradox in this article, and its definition in general cases is provided as follows.

Definition 2: When adopting the "C2" strategy to compare optimization algorithms $A_i, i = 1, \ldots, k$, a cycle ranking is defined by a situation in which there is no A_i such that $A_i \succeq A_j$ for all $1 \le j \le k$. In other words, if no winner algorithm exists, then we say a cycle ranking occurs, or the random event $\mathbb{C}$ occurs for simplicity.

B. *"Survival of the Nonfittest" Paradox: From "C2+"*

The following matrix is another example of the probabilistic model (5), where three algorithms are tested on 50 problems and parameters p_i satisfy condition (6):

$$\mathcal{N} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\ 10 & 0 & 20 & 0 & 20 & 0 \\ A_1 & A_1 & A_2 & A_2 & A_3 & A_3 \\ A_2 & A_3 & A_1 & A_3 & A_1 & A_2 \\ A_3 & A_2 & A_3 & A_1 & A_2 & A_1 \end{bmatrix}. \tag{10}$$

If we adopt the "C2+" strategy to compare these three algorithms, then A_2 or A_3 is the winner since both of them perform the best on 20 problems, while A_1 only performs the best on ten problems.

However, if we adopt the "C2" strategy, then we have the following three matrices:

$$\begin{bmatrix} P_{12} & 1-p_{12} \\ 30 & 20 \\ A_1 & A_2 \\ A_2 & A_1 \end{bmatrix}, \begin{bmatrix} p_{13} & 1-p_{13} \\ 30 & 20 \\ A_1 & A_3 \\ A_3 & A_1 \end{bmatrix}, \begin{bmatrix} p_{23} & 1-p_{23} \\ 30 & 20 \\ A_2 & A_3 \\ A_3 & A_2 \end{bmatrix}. \tag{11}$$

Therefore, $A_1 \succeq A_2$, $A_1 \succeq A_3$, and $A_2 \succeq A_3$, respectively. In other words, A_1 is the winner under the "C2" strategy.

As a result, a different winner would be selected if a different comparison strategy were to be adopted. Specifically, the loser (A_2 or A_3) under the "C2" strategy will be the winners under the "C2+" strategy. Such phenomenon is termed "survival of the nonfittest" in this article, whose definition in general cases is provided as follows.

Definition 3: When comparing optimization algorithms, if one of the losers under "C2" becomes the winner under "C2+," i.e., the winner under "C2+" is not the winner under "C2," then we say that the survival of the nonfittest occurs, or the random event $\mathbb{S}$ occurs for simplicity.

When comparing $k > 3$ optimization algorithms, all choices of the "C2+" strategy imply that at least three algorithms are compared on each problem. Therefore, survival of the nonfittest paradox is possible for all choices of "C2+."

C. *Paradoxes Seen in Practice*

In this section, we illustrate cycle ranking and survival of the nonfittest with real data. The data were gathered when comparing three global optimization algorithm functions in MATLAB 2016b, namely, "particleswarm" (particle swarm optimization), "ga" (genetic algorithm), and "simulannealbnd" (simulate anneal). Twelve benchmark problems in the Hedar test suite [42] were used in the tests, and 50 independent runs were executed for each problem. At each run, the algorithms stopped when 20 000 function evaluations were consumed. The means and standard deviations of the found best function values are provided in the supplementary material.

Tables I and II show the rankings under "C2+" of these algorithms on 12 benchmark problems. For each problem in Tables I and II, the ranking number "1" indicates that the corresponding algorithm performs the best on this problem, and "3" performs the worst.

If we adopt the "C2" strategy to analyze the rankings in Table I, it is easy to find that "ga" $\succeq$ "particleswarm," "particleswarm" $\succeq$ "simulannealbnd," and "simulannealbnd" $\succeq$ "ga" (each winning three times versus twice), i.e., cycle ranking occurs. If we adopt the "C2" strategy to analyze the rankings in Table II, then we have "particleswarm" $\succeq$ "simulannealbnd" (5 versus 4), "particleswarm" $\succeq$ "ga" (7 versus 2), and "simulannealbnd" $\succeq$ "ga" (5 versus 4). Therefore, "particleswarm" is the winner under "C2." However, if we adopt "C2+" to analyze Table II, then "simulannealbnd" is the winner by the plurality rule since it wins most of the times (4 versus 3 versus 2). As a result, the survival of the nonfittest occurs, i.e., "simulannealbnd" is not the winner under "C2," but is under "C2+."

TABLE I
CYCLE RANKING PARADOX IN PRACTICE

Function name	particleswarm	ga	simulannealbnd
Griewank(5D)	2	1	3
Hump(2D)	1	3	2
Michalewics(10D)	2	1	3
Rosenbrock(20D)	2	3	1
Rastrigin(5D)	3	2	1

TABLE II
SURVIVAL OF THE NONFITTEST PARADOX IN PRACTICE

Function name	particleswarm	ga	simulannealbnd
Beale(2D)	1	3	2
Easom(2D)	1	2	3
Levy(2D)	1	2	3
Michalewics(10D)	2	1	3
Powersum(4D)	2	3	1
Rosenbrock(20D)	2	3	1
Schwefel(2D)	2	1	3
Shekel7(4D)	2	3	1
Sphere(20D)	2	3	1
Perform the best	3	2	4

TABLE III
TWO POSSIBLE PARADOXES ARISING FROM TWO DIFFERENT COMPARISON STRATEGIES

Comparison strategy	Some applications		Possible paradox
	Algorithm developing	Algorithm competition	
C2	Ref.[21], [22], [23], [24], [25], [9]	CEC [20]	Cycle ranking
C2+	Ref.[28], [29], [30], [31], [32], [11]	BBOB COCO [12], [6],IOHprofiler [26],BBComp [27]	Survival of the nonfittest

Finally, we report details of how the above paradoxes were discovered. To begin, we simply select four ("ps" besides the three algorithms mentioned above) global optimization algorithms in MATLAB 2016b and the whole Hedar test suite which includes 27 problems (68 if consider different dimensions). After numerical experiments finish, "ps" is excluded partly because it is deterministic while all the other three algorithms are stochastic, and partly because we wish to reduce complexity. Through trivial statistical tests, rankings of these three algorithms on each problem are obtained, and problems with tie rankings are excluded. Finally, we select 12 problems which provide enough evidences of paradoxes (see Tables I in the supplementary material for the raw data).

D. Discussion and the Impact of Paradoxes

Table III summarizes these two possible paradoxes that come from two different comparison strategies, relevant references on algorithm development and four algorithms competitions. Here, we provide a brief analysis about these four algorithms competitions.

The "C2" strategy is adopted in the CEC algorithms competition, in which several matches are needed to compare all participating algorithms since only two algorithms are compared in each match. Statistical test are adopted in each match to determine the winner. Multiple groups of rankings are illustrated according to different computational costs, referring to Fig. 5 for an example. From the analysis in Section IV-A, a cycle ranking paradox may occur in the CEC competition.

The "C2+" strategy is adopted in the BBOB COCO, IOHprofiler, and BBComp competitions, and only one single match is enough to compare all participating algorithms. The main difference among them is how to deal with the randomness of stochastic algorithms. For BBComp, random errors are neglected. Specifically, the stochastic algorithms are treated the same as the deterministic ones since only one single run is allowed for each algorithm. For BBOB COCO, random errors are "averaged" through 100 independent runs, and in this sense, the stochastic algorithms are treated as the deterministic ones. For IOHprofiler, confidence intervals are calculated to measure random errors, and a range defined by confidence intervals is displayed dynamically for each algorithm. Such comparison method is comprehensive, but makes the comparison results, e.g., Fig. 7, difficult to comprehend if the number of participating algorithms is relatively large. According to the analysis of Section IV-B, the survival of the nonfittest paradox may occur in all these three algorithms competitions.

The existence of cycle ranking and survival of the nonfittest show that the two comparison strategies ("C2, C2+") are not always compatible in benchmarking the optimality performance of optimization algorithms. These phenomena are examples of the well-known statement: a collective choice may be irrational, even if all the individual choices are rational [43]. In many social problems and economic problems, such irrational collective choices are well known, and the election is a typical example [44].

In fact, since the 1950s, research on several paradoxes has been in the mainstream of the voting theory, where both "C2" and "C2+" are widely studied. The most well-known paradox may be the Condorcet paradox (i.e., the cycle ranking paradox), which is pointed out by Marquis de Condorcet in 1785 and tells that the collective preference on all the candidates may be cyclic [44]. The cycle ranking paradox proposed in this article is an extension of the Condorcet paradox to numerical comparison of optimization algorithms.

Moreover, the survival of the nonfittest paradox is an extension of the strong Borda paradox and the strict Borda paradox [44]. The strong Borda paradox requires the worst of "C2" be selected as the winner of "C2+." Besides this requirement, the strict Borda paradox requires the ranking be exactly reversed, i.e., the worst of "C2" is the winner of "C2+," the second-worst of "C2" is the second winner of "C2+," and so on. The survival of the nonfittest paradox only requires that a nonwinner of "C2" be selected as the winner of "C2+."

An implication of the paradoxes illustrated in this section is that we have to become more conservative when interpreting comparison results and in benchmarking. At least, we should realize the significant impact of the comparison strategy on the comparison results.

A natural question regarding the existence of resultant paradoxes is: are they deterministic or stochastic? If the latter, what affects their occurrence probabilities and how does the triplet (m, k, p_i) influence the probabilities and benchmarking as a whole?

V. OCCURRENCE PROBABILITIES OF PARADOXES

In this section, we calculate the occurrence probabilities of the two paradoxes (cycle ranking and survival of the nonfittest). We are particularly interested in how the probabilities change as the number of problems m and/or the number of algorithms k increase.

First, we need to determine the parameters $p_i, i = 1, \dots, k!$ in the probabilistic model (5). From a theoretical viewpoint, the no-free-lunch (NFL) theorem in optimization [45] is suitable for our purpose. According to the NFL theorem, all possible (infinite) optimization problems as a whole are unbiased on any optimization algorithm. In other words, for any given set of optimization algorithms, each algorithm has the same probability to perform the best on the full test set. In this article, we adopt the following NFL assumption, which is a direct application of the NFL theorem.

Assumption 2 (NFL Assumption): For given k optimization algorithms and a set of test problems, all $k!$ possible rankings of these algorithms are equally likely, i.e., all occurring with probability p_i given by

$$p_i = \frac{1}{k!}, \ i = 1, 2, \dots, k!. \tag{12}$$

This NFL assumption appears similar to but inequivalent to the NFL theorem of [45] which assumes that an infinite set of all possible optimization problems are unbiased toward a particular optimization algorithm. Our NFL assumption only requires that a given problem set (finite in practice) is unbiased toward a given optimization algorithm. In this regard, Assumption (12) is an application of the NFL theorem. Moreover, the NFL assumption is the same as and the first justifiable actual use of the impartial culture (IC) assumption (or condition) in voting theory [44].

We should highlight two facts here. The first is that the NFL theorem does not change the desire for seeking a winner on a numerical comparison of optimization algorithms. The reason is that the NFL theorem only holds under a limit condition, while the numerical comparison is popular in practice for verifying the performance of optimization algorithms before reaching the limit. The second fact is that the NFL assumption does not affect the existence of paradoxes. The reason is that the NFL assumption only influences the probabilities $P_i, i = 1, \dots, k!$ in the mathematical model (5). However, the existence of paradoxes does not depend on the exact values of $P_i, i = 1, \dots, k!$ (referring to Section IV).

In effect, the role of the NFL assumption in this article is only to provide a convenient "platform" for the calculation of occurrence probabilities of the paradoxes. Without the NFL assumption, these calculations would be much harder, and the values of the occurrence probabilities would be different.

A. Comparison Strategy Theorems

In this section, we provide several theorems to set the scene for our later analysis.

1) Existence and Uniqueness of Winners:

Theorem 1: When comparing optimization algorithms,

1) a winner under "C2+" always exists and may not be unique;
2) a winner under "C2" may not exist, and if it exists, then it may not be unique.

Proof: 1) According to the plurality assumption, the winner under "C2+" is the algorithm which lies in the ith column and the third row of the matrix (5), where i satisfies $x_i = \max\{x_1, \dots, x_{k!}\}$. Since i always exists and may not be unique, the winner under "C2+" exists and may not be unique.

2) From Definition 2, the existence of a cycle ranking paradox implies that the winner under "C2" may not exist. If a "C2" winner exists, then it may not be unique due to the following two reasons. First, there may exist a tie between all pairs of the best algorithms when m is even. Second, when the number of algorithms exceeds 3, it is possible that the best algorithms form a cycle, and each of them performs better than the other algorithms. ■

Theorem 2: If there is an x_i in matrix (5) that satisfies $x_i \geq m/2$, then both cycle ranking and survival of the nonfittest are impossible.

Proof: According to the plurality assumption, since $x_i \geq m/2$, the algorithm which lies in the ith column and the third row of matrix (5) must be the winner under "C2+." Therefore, a cycle is impossible. Moreover, this winner under "C2+" is also the winner under "C2" since it votes $Y \geq x_i$ [referring to matrix (7)]. Hence, survival of the nonfittest is also impossible. ■

2) Division of the Sample Space: For the convenience of expression, we present an additional definition below, of a "normal" case when comparing optimization algorithms.

Definition 4: If one of the winners under the "C2+" strategy is also the winner under the "C2" strategy, we say that the random event $\mathbb{N}$ occurs.

Then, we have the following theorem, which is an extension of [46, Th. 1].

Theorem 3: The random events $\mathbb{C}, \mathbb{N}$, and $\mathbb{S}$ is a partition of the sample space when adopting the "C2" and "C2+" strategies to compare optimization algorithms.

Proof: When adopting the "C2" strategy to compare optimization algorithms $A_i, i = 1, 2, \dots, k$, there are $2^{k(k-1)/2}$ possible basic events. Some of these events form cycles, the others do not. According to Definition 2, the former basic events imply that the event $\mathbb{C}$ occurs, while the latter imply that $\mathbb{C}$ does not occur.

Among those latter basic events, the winner under "C2" always exists. According to Theorem 1, the winner under "C2+" always exists, too. If all the winners under "C2+" are not the winner under "C2," then random event $\mathbb{S}$ occurs. Otherwise, at lease one of the winners under "C2+" is the winner under "C2," and therefore, $\mathbb{N}$ occurs.

Since there are no other basic events left, the proof is thus complete. ■

The following corollary is a direct application of Theorem 3, which is helpful for our later calculations.

Corollary 1: When adopting the "C2" and "C2+" strategies to compare optimization algorithms, the probabilities of the random events $\mathbb{C}, \mathbb{S}$, and $\mathbb{N}$ satisfy

$$P(\mathbb{C}) + P(\mathbb{S}) + P(\mathbb{N}) = 1. \tag{13}$$

B. Occurrence Probability of "Cycle Ranking"

In this section, we calculate the occurrence probabilities of cycle ranking. Only three algorithms ($k = 3$) are considered first, and then the general case $k > 3$ is analyzed.

We have presented that the cycle ranking paradox studied in this article is the same as Condorcet's paradox in voting theory. Certain results of Condorcet's paradox are thus applied directly to cycle ranking in this section. However, a new method and new results are provided.

1) Case I [Comparing Three Algorithms ($k = 3$)]: The calculation of occurrence probabilities of paradoxes is often hard, especially, when k is large. Therefore, we consider the case $k = 3$ first. The following theorem is reproduced from our recent conference paper [46], where the proof is absent due to the limitation of space.

Theorem 4: When comparing three optimization algorithms on m problems, the occurrence probability of random event $\mathbb{C}$ is given by

$$P(\mathbb{C}) = \frac{1}{6^m}\left(2\sum_{\{x_i\}\in C_1}\frac{m!}{x_1!\cdots x_6!} - \sum_{\{x_i\}\in C_2}\frac{m!}{x_1!\cdots x_6!}\right) \quad (14)$$

where C_1 and C_2 are determined, respectively, by

$$\begin{cases} x_1+\cdots+x_6=m \\ x_1+x_2+x_5\geq\frac{m}{2} \\ x_1+x_3+x_4\geq\frac{m}{2} \\ x_4+x_5+x_6\geq\frac{m}{2} \\ x_i\in\{0,1,\dots,m\}\end{cases},\quad \begin{cases} x_1+\cdots+x_6=m \\ x_1+x_2+x_5=\frac{m}{2} \\ x_1+x_3+x_4=\frac{m}{2} \\ x_4+x_5+x_6=\frac{m}{2} \\ x_i\in\{0,1,\dots,m\}\end{cases},\quad i=1,\dots,6. \quad (15)$$

Moreover, when m is odd, C_2 is empty.

Proof: When comparing only three optimization algorithms (say, A_1, A_2, A_3) by the "C2" strategy, all possible basic random events are given

$$\begin{array}{llll} (G_1) & A_1\succeq A_2 & A_2\succeq A_3 & A_1\succeq A_3 \\ (G_2) & A_1\succeq A_2 & A_3\succeq A_2 & A_3\succeq A_1 \\ (G_3) & A_1\succeq A_2 & A_3\succeq A_2 & A_1\succeq A_3 \\ (G_4) & A_2\succeq A_1 & A_2\succeq A_3 & A_3\succeq A_1 \\ (G_5) & A_2\succeq A_1 & A_2\succeq A_3 & A_1\succeq A_3 \\ (G_6) & A_2\succeq A_1 & A_3\succeq A_2 & A_3\succeq A_1 \\ (G_7) & A_2\succeq A_1 & A_3\succeq A_2 & A_1\succeq A_3 \\ (G_8) & A_1\succeq A_2 & A_2\succeq A_3 & A_3\succeq A_1. \end{array} \quad (16)$$

Both G_7 and G_8 imply that the occurrence of the random event $\mathbb{C}$, and all the other six basic random events bring clear final winners. Therefore,

$$P(\mathbb{C}) = P(G_7) + P(G_8) - P(A_1 = A_2, A_2 = A_3, A_3 = A_1).$$

Due to the symmetry, $P(G_7) = P(G_8)$. Thus,

$$P(C) = 2P(G_8) - P(A_1 = A_2, A_2 = A_3, A_3 = A_1).$$

In this case, the following matrix:

$$\begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ A_1 & A_1 & A_2 & A_2 & A_3 & A_3 \\ A_2 & A_3 & A_1 & A_3 & A_1 & A_2 \\ A_3 & A_2 & A_3 & A_1 & A_2 & A_1 \end{bmatrix} \quad (17)$$

is the mathematical model we need, where x_i satisfies

$$\sum_{i=1}^{6} x_i = m, \quad x_i\in[0,m], \quad i=1,\dots,6. \quad (18)$$

Since $(x_1,\dots,x_6)^T$ satisfies the multinomial distribution with parameters m and $(1/6,\dots,1/6)$, we have, therefore,

$$P(G_8) = \sum_{\{x_i\}\in C_1}\frac{m!}{x_1!\cdots x_6!}\frac{1}{6^m}$$

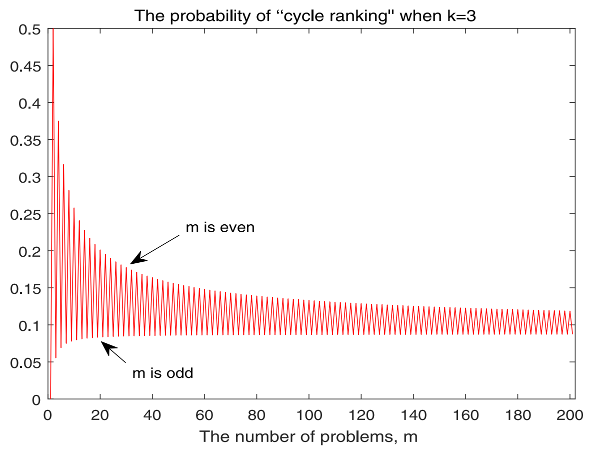


Fig. 8. How the probability of cycle ranking $P(\mathbb{C})$ changes as the number of problems m increases, where the number of algorithms $k = 3$ is fixed.

where $C_1 = \{(x_1,\dots,x_6)|A_1\succeq A_2, A_2\succeq A_3, A_3\succeq A_1\}$ is defined by

$$\begin{cases} x_1+\cdots+x_6=m & \\ x_1+x_2+x_5\geq\frac{m}{2} & [A_1\succeq A_2] \\ x_1+x_3+x_4\geq\frac{m}{2} & [A_2\succeq A_3] \\ x_4+x_5+x_6\geq\frac{m}{2} & [A_3\succeq A_1] \\ x_i\in\{0,1,\dots,m\}, i=1,\dots,6. & \end{cases} \quad (19)$$

In (19), $A_1\succeq A_2$ requires that $x_1+x_2+x_5\geq x_3+x_4+x_6$, which is equivalent to $x_1+x_2+x_5\geq m/2$ due to condition (18). Similarly, $A_2\succeq A_3$ requires that $x_1+x_3+x_4\geq m/2$, and $A_3\succeq A_1$ requires that $x_4+x_5+x_6\geq m/2$.

Consequently, we have

$$P(A_1 = A_2, A_2 = A_3, A_3 = A_1) = \sum_{\{x_i\}\in C_2}\frac{m!}{x_1!\cdots x_6!}\frac{1}{6^m}$$

where $C_2 = \{(x_1,\dots,x_6)|A_1 = A_2, A_2 = A_3, A_3 = A_1\}$ is defined by

$$\begin{cases} x_1+\cdots+x_6=m & \\ x_1+x_2+x_5=\frac{m}{2} & [A_1=A_2] \\ x_1+x_3+x_4=\frac{m}{2} & [A_2=A_3] \\ x_4+x_5+x_6=\frac{m}{2} & [A_3=A_1] \\ x_i\in\{0,1,\dots,m\}, \quad i=1,\dots,6. & \end{cases}$$

Hence, the probability $P(\mathbb{C})$ is given by (14). It is obvious C_2 is empty when m is odd since $A_1 = A_2$ is impossible. ■

Although probability $P(\mathbb{C})$ is expressed explicitly in Theorem 4, it is still hard to calculate its exact value, especially, when the number of problems m is relatively large. In [46], $P(\mathbb{C})$ is calculated numerically for $m = 1, 2, \dots, 100$.

In this article, more probabilities are calculated numerically. Specifically, all possible $(x_1, x_2, \dots, x_6)$ satisfying condition (18) and Theorem 2 (i.e., $x_i < m/2$ for all i) are enumerated first. If (15) is satisfied, then they are summed up through (14) to calculate $P(\mathbb{C})$. Fig. 8 shows the probability for $m = 1, 2, \dots, 201$. Many of these values have been reported in voting theory, especially, the probabilities for odd m [44].

From Fig. 8, we can see two opposite trends. When m is even, $P(\mathbb{C})$ decreases from 0.5 to near 0.1187 as m increases. On the contrary, when m is odd, $P(\mathbb{C})$ increases from 0 to

TABLE IV
OCCURRENCE PROBABILITY OF CYCLE RANKING $P(\mathbb{C})$ WHEN COMPARING k ALGORITHMS ON m PROBLEMS

$k \setminus m$	5	15	25	35	45	∞
3	.0694	.0820	.0843	.0853	.0855	.0877
4	.1389	.1640	.1686	.1706	.1710	.1755
5	.1995	.2350	.2417	.2444	.2460	.2513
6	.2514	.2952	.3034	.3068	.3104	.3152
7	.2958	.3463	.3556	.3570	.3600	.3692
8	.3339	.3900	.4003	.3942	.3903	.4151
9	.3676	.4260	.4370	.4420	.4450	.4545
10	.3986	.4517	.4633	.5479	.6160	.4886
11	.4232	.4890	.5010	.5060	.5090	.5187

near 0.0873 as m increases. An interesting question is what the value of $P(\mathbb{C})$ is when $m \to \infty$. The answer is derived as follows:

$$\lim_{m\to\infty} P(\mathbb{C}) = \frac{1}{2\pi}\arccos\left(\frac{23}{27}\right) \approx 0.0877 \qquad (20)$$

which is a direct application of the IC-based limiting probability of Condorcet's paradox [44] with the NFL assumption. Several methods have been reported to calculate this important value [44], [47]. In [47], a special technique, a pictogram, for vote vector with a cycle is introduced, and the Condorcet paradox is discussed by means of a pictogram. Through abstruse mathematical calculations, the result (20) is obtained. Refer to [44] and [47] for more technical details.

Therefore, we can conclude that the occurrence probability of cycle ranking is about 8.77% when only three algorithms are compared and the number of optimization problems approaches infinity. An odd number of optimization problems is preferred since it brings a much lower occurrence probability than its even neighbors.

2) Case II (Comparing More Than Three Algorithms): As the number of algorithms k increases, the calculations of probabilities becomes harder and harder. In this section, we reproduce the results of Condorcet's paradox under the IC assumption. Since the NFL assumption is an example of the IC assumption, these results are thus also true for cycle ranking under the NFL assumption.

Table IV shows the occurrence probabilities of cycle ranking when comparing $k = 3, 4, \ldots, 11$ algorithms on $m = 5, 15, \ldots, 45, \infty$ problems. When k is odd, the results are the complementary probabilities of those listed in [44, Table 4.6]. These values were calculated through designing some subtle computer enumeration procedure, refer to [44] and references therein for details.

From Table IV, we can see that $P(\mathbb{C})$ increases as m or k increases. Specifically, it increases slowly as $m > 15$ increases. However, it increases rapidly (for any given m) as k increases. The limiting probabilities of $P(\mathbb{C})$ are very important theoretically, and the last column of Table IV shows that they grow as k increases, as shown Fig. 9.

These results imply that cycle ranking may be frequent in a large competition of optimization algorithms (i.e., when k is large). Although $k < 10$ is more often the case in practice or in optimization research, the occurrence probability of cycle ranking is still too large to be ignored, especially, in benchmarking.

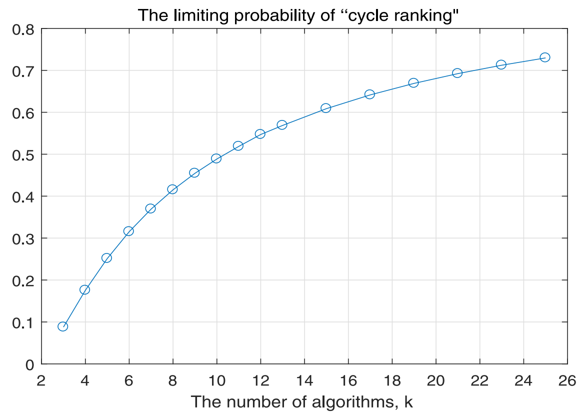


Fig. 9. How the limiting probability of cycle ranking changes as the number of algorithms k increases.

C. Occurrence Probability of "Survival of the Nonfittest"

It is harder to calculate the occurrence probability of survival of the nonfittest paradox or the random event $\mathbb{S}$ than that of $\mathbb{C}$. One reason is that the "nonfittest" contains many possibilities especially when k is large. In this section, we first consider the case of only three algorithms, and then analyze the general case. For the former case, we calculate $P(\mathbb{S})$ directly. For the latter case, some results of the voting theory are adopted to calculate $P(\mathbb{S})$ indirectly.

1) Case I (Comparing Three Algorithms): The following theorem is reproduced from our recent conference paper [46] with a minor revision correcting an error there. Its proof is omitted in [46] due to the limitation of space.

Theorem 5: When comparing three optimization algorithms on m test problems, the occurrence probability of a random event $\mathbb{S}$ is given by

$$P(\mathbb{S}) = \frac{3}{6^m}\left(\sum_{\{x_i\}\in S_1}\frac{m!}{x_1!\cdots x_6!} + \sum_{\{x_i\}\in S_2}\frac{m!}{x_1!\cdots x_6!}\right) \qquad (21)$$

where dominant S_1 and S_2 are defined as

$$S_1\colon \begin{cases} x_1+\cdots+x_6 = m \\ x_1+x_2+x_5 > \frac{m}{2} \\ x_1+x_2+x_3 > \frac{m}{2} \\ \max\{x_3+x_4, x_5+x_6\} > x_1+x_2 \\ x_i \in \{0,1,\ldots,m\}, \ i = 1,\ldots,6 \end{cases}$$

$$S_2\colon \begin{cases} x_1+\cdots+x_6 = m \\ x_1+x_2+x_5 = \frac{m}{2} \\ x_1+x_2+x_3 > \frac{m}{2} \\ x_1+x_3+x_4 > \frac{m}{2} \\ x_5+x_6 > \max\{x_1+x_2, x_3+x_4\} \\ x_i \in \{0,1,\ldots,m\}, \ i = 1,\ldots,6. \end{cases}$$

Moreover, when m is odd, S_2 is empty.

Proof: From (16), the basic random events G_1–G_6 may result in the occurrence of random event $\mathbb{S}$ or $\mathbb{N}$. If the winner of "C2+" is not the winner of "C2+," then random event $\mathbb{S}$ occurs. Otherwise, random event $\mathbb{N}$ occurs.

Since the intersection of some of these six basic random events is not empty, we repartition them as the following random events, any two of which are empty intersected:

$$\begin{aligned}
&(G_1') \quad A_1 \succ A_2 \quad A_1 \succ A_3\\
&(G_2') \quad A_2 \succ A_1 \quad A_2 \succ A_3\\
&(G_3') \quad A_3 \succ A_1 \quad A_3 \succ A_2\\
&(G_4') \quad A_1 = A_2 \quad A_2 \succ A_3 \quad A_1 \succ A_3\\
&(G_5') \quad A_2 = A_3 \quad A_2 \succ A_1 \quad A_3 \succ A_1\\
&(G_6') \quad A_3 = A_1 \quad A_1 \succ A_2 \quad A_3 \succ A_2.
\end{aligned} \tag{22}$$

If we denote $A^*_{\text{C2+}}$ as the winner algorithm under the "C2+" strategy, then the random event $\mathbb{S}$ can be partitioned as follows:

$$\begin{aligned}
&(H_1) \quad A_1 \succ A_2 \quad A_1 \succ A_3 \quad A^*_{\text{C2+}} \neq A_1\\
&(H_2) \quad A_2 \succ A_1 \quad A_2 \succ A_3 \quad A^*_{\text{C2+}} \neq A_2\\
&(H_3) \quad A_3 \succ A_1 \quad A_3 \succ A_2 \quad A^*_{\text{C2+}} \neq A_3\\
&(H_4) \quad A_1 = A_2 \quad A_2 \succ A_3 \quad A_1 \succ A_3 \quad A^*_{\text{C2+}} = A_3\\
&(H_5) \quad A_2 = A_3 \quad A_2 \succ A_1 \quad A_3 \succ A_1 \quad A^*_{\text{C2+}} = A_1\\
&(H_6) \quad A_3 = A_1 \quad A_1 \succ A_2 \quad A_3 \succ A_2 \quad A^*_{\text{C2+}} = A_2.
\end{aligned} \tag{23}$$

Here, $H1$ implies that A_1 is the winner under "C2," while it is not the winner under "C2+." The other random events can be explained similarly. Therefore,

$$P(\mathbb{S}) = \sum_{j=1}^{6} P(H_j) = 3(P(H_1) + P(H_4)). \tag{24}$$

The symmetry is adopted to reduce the second equality.

Following the mathematical model (17) and the multinomial distribution with parameters m and $(1/6, \ldots, 1/6)$, we have:

$$P(H_1) = \sum_{\{x_i\} \in S_1} \frac{m!}{x_1! \cdots x_6!} \frac{1}{6^m} \tag{25}$$

where $S_1 = \{(x_1, \ldots, x_6) | H_1\}$ is defined as follows:

$$\begin{cases}
x_1 + \cdots + x_6 = m & \\
x_1 + x_2 + x_5 > \frac{m}{2} & [A_1 \succ A_2]\\
x_1 + x_2 + x_3 > \frac{m}{2} & [A_1 \succ A_3]\\
\max\{x_3 + x_4, x_5 + x_6\} > x_1 + x_2 & [A^*_{\text{C2+}} \neq A_1]\\
x_i \in \{0, 1, \ldots, m\}, \quad i = 1, \ldots, 6. &
\end{cases} \tag{26}$$

Similarly, we have

$$P(H_4) = \sum_{\{x_i\} \in S_4} \frac{m!}{x_1! \cdots x_6!} \frac{1}{6^m} \tag{27}$$

where $S_4 = \{(x_1, \ldots, x_6) | H_4\}$ is defined as follows:

$$\begin{cases}
x_1 + \cdots + x_6 = m & \\
x_1 + x_2 + x_5 = \frac{m}{2} & [A_1 = A_2]\\
x_1 + x_2 + x_3 > \frac{m}{2} & [A_1 \succ A_3]\\
x_1 + x_3 + x_4 > \frac{m}{2} & [A_2 \succ A_3]\\
x_5 + x_6 > \max\{x_1 + x_2, x_3 + x_4\} & [A^*_{\text{C2+}} = A_3]\\
x_i \in \{0, 1, \ldots, m\}, \quad i = 1, \ldots, 6. &
\end{cases} \tag{28}$$

Therefore, the probability $P(\mathbb{S})$ is given by (21), and it is obvious that S_2 is empty when m is odd since $A_1 = A_2$ is impossible. ■

Based on Theorem 5, for any given m, we can calculate the probability $P(\mathbb{S})$ by enumeration. However, it is computationally expensive. Fig. 10 shows the numerical results of $P(\mathbb{S})$ for $m = 1, 2, \ldots, 201$, which includes many more values than those provided [46].

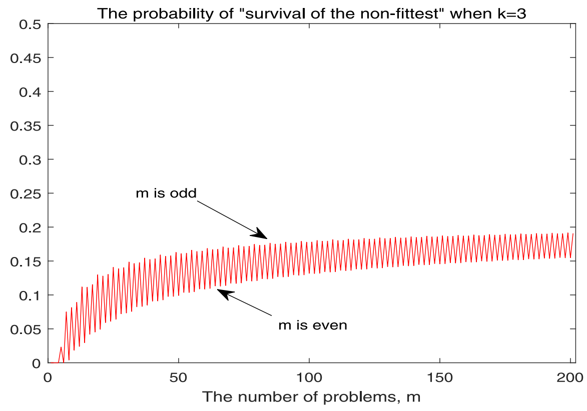


Fig. 10. How the occurrence probability of survival of the nonfittest $P(\mathbb{S})$ changes as m increases, where the number of algorithms $k = 3$ is fixed.

From Fig. 10, we can observe that $P(\mathbb{S})$ increases slowly as $m > 20$ increases, whether m is even or odd. However, $P(\mathbb{S})$ is always smaller for even m than its odd neighbors.

An interesting and important question is whether there is a limit of $P(\mathbb{S})$ when $m \to \infty$? The answer is yes, and the limiting probability is given as follows:

$$\lim_{m \to \infty} P(\mathbb{S}) = 0.2215. \tag{29}$$

See the next subsection for the reason.

2) Case II (Comparing More Than Three Algorithms): The calculation of $P(\mathbb{S})$ becomes more and more difficult as the number of algorithms k increases. In this section, we introduce certain results of "Condorcet efficiency" in voting theory, and then calculate $P(\mathbb{S})$ indirectly.

The "Condorcet efficiency" is a conditional probability which tells the chance whether a voting rule will select the Condorcet winner if it exists [44]. The Condorcet winner is a candidate which can beat all the other candidates under the "C2" comparison strategy. In other words, if the winner under "C2" exists, the "Condorcet efficiency of plurality rule" is the conditional probability that a winner under "C2" is also the winner under "C2+."

According to Definition 2 and Theorem 3, there are only three random events $\mathbb{C}, \mathbb{N}$, and $\mathbb{S}$ in a comparison of optimization algorithms, and the winner under "C2" exists if and only if when a cycle ranking does not occur. Therefore, the "Condorcet efficiency of plurality rule" P_e can be rewritten as

$$P_e = \frac{P(\mathbb{N})}{1 - P(\mathbb{C})} = 1 - \frac{P(\mathbb{S})}{1 - P(\mathbb{C})}. \tag{30}$$

Thus, we have

$$P(\mathbb{S}) = (1 - P_e)(1 - P(\mathbb{C})). \tag{31}$$

Since $P(\mathbb{C})$ has been calculated in previous section and several results of P_e are known in voting theory, then (31) can be used to calculate $P(\mathbb{S})$ indirectly.

Table V shows how the limiting probability of P_e, $P(\mathbb{C})$ and $P(\mathbb{S})$ change as k increases. The values of P_e in the second row are reproduced from [48], where a computer simulation

TABLE V
HOW THE LIMITING PROBABILITIES OF P_e, $P(\mathbb{C})$, $P(\mathbb{S})$, AND $P(\mathbb{N})$ CHANGE AS k INCREASES. ALL VALUES WITH ONLY THREE DIGITS ARE SIMULATION ESTIMATES

k	3	4	5	6	7	8	9
P_e	.7572	.646	.571	.521	.440	.420	.393
$P(\mathbb{C})$	.0877	.1755	.2513	.3152	.3692	.4151	.4545
$P(\mathbb{S})$	.2215	.292	.321	.328	.353	.339	.331
$P(\mathbb{N})$	.6908	.533	.428	.357	.278	.246	.215

process was developed-based upon the representation of the Condorcet efficiencies as multivariate normal positive orthant probabilities [48]. The values of $P(\mathbb{C})$ in Table V are reproduced from Table IV in this article, and the values of $P(\mathbb{S})$ are calculated according to (31). Since the values of P_e are estimated from simulations except when $k = 3$, all values with only three digits are therefore simulation estimations. It is suggested that the simulated values of P_e possess a maximum error about 0.025 [48]. According to (31), the values of $P(\mathbb{S})$ possess almost the same error.

Since P_e and $P(\mathbb{C})$ possess different monotonicity as k increases, the monotonicity of $P(\mathbb{S})$ is unclear according to (31). From Table V, we can see that $P(\mathbb{S})$ increases when $k = 3, 4, 5, 6$, and 7, and then decreases when $k = 8$ and 9. Therefore, $P(\mathbb{S})$ obtains its maximal value 35.3% when $k = 7$.

D. Occurrence Probability of the Random Event $\mathbb{N}$

According to (13) or (30), we can calculate the probability $P(\mathbb{N})$ through

$$P(\mathbb{N}) = 1 - P(\mathbb{C}) - P(\mathbb{S}) = P_e(1 - P(\mathbb{C})). \tag{32}$$

Fig. 11 shows the values of $P(\mathbb{N})$ for $k = 3$ and $1 \leq m \leq 201$, and the fifth row of Table V shows the limiting values of $P(\mathbb{N})$ for $k = 3, 4, \ldots, 9$.

From Fig. 11, we have found that $P(\mathbb{N})$ zigzags violently when m is small, and the amplitude becomes very small when $m > 60$. Approximately, $P(\mathbb{N})$ decreases slowly as $m > 30$ increases, and, it finally converges to about 0.6908 (Table V).

From Table V, we can see that the limiting values of $P(\mathbb{N})$ decreases as k increases. This is arises from (32) since both P_e and $1 - P(\mathbb{C})$ decrease as k increases. In other words, the more algorithms that are compared, the smaller the probability that the winner under "C2+" is also the winner under "C2." For example, when $k = 3, 5$, and 7, the probabilities are about 69%, 43%, and 28%, respectively.

E. Discussion

The analysis and calculations proposed in this article show that both "C2" and "C2+" can have problems with their use. The "C2" strategy may bring the occurrence of cycle ranking, and therefore no winner algorithm will be elected. On the other hand, "C2+" may cause the survival of the nonfittest, i.e., the winner under "C2+" will be defeated in at least one match of "C2."

Although the occurrence probability of cycle ranking is small (less than 9% in the sense of limiting probability) when there are only three algorithms being compared, it increases significantly as the number of algorithms increases.

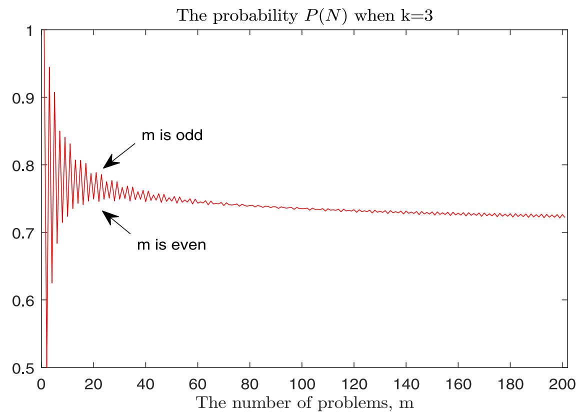


Fig. 11. When only three algorithms are compared, how $P(\mathbb{N})$ changes as m increases.

For instance, the limiting probability becomes larger than 30% when there are six algorithms, and larger than 40% when there are eight algorithms (Table V). On the contrary, the occurrence probability of survival of the nonfittest increases slowly as the number of algorithms increases. Moreover, it begins to decrease when the number of algorithms is greater than 7 (Table V). Therefore, "C2+" is more preferred than "C2" when the number of algorithms is relatively large.

Despite that cycle ranking may occur on a "C2" strategy, it is self-verified. Specifically, after the whole comparison of "C2," it is clear whether the cycle ranking paradox happens or not. If not, then the result is certain. On the other hand, "C2+" is not self-verified and thus its results are doubtful. In effect, only when both "C2+" and "C2" are adopted, can it be known whether survival of the nonfittest happens or not. In this sense, "C2" is more preferred than "C2+."

All the probabilities are derived under the NFL assumption, which is a strong condition. Therefore, the probabilities derived in this article should not be used unrestricted for any given set of optimization problems. The main effect of these probabilities is to provide a point of reference to the frequency of possible paradoxes. Further, it is a useful reference to future calculations without the NFL assumption.

VI. CONCLUSION

This article has investigated compatibility issues arising from comparing optimization algorithms and has mathematically defined and modeled the popular "C2" and "C2+" comparison strategies for the first time. To the best of our knowledge, few studies have considered the differences between these two strategies. This article has rigorously illustrated that the results can be incompatible. In particular, two paradoxes, namely, cycle ranking and survival of the nonfittest, are discovered probable in numerical comparison of optimization algorithms.

Then the occurrence probabilities of the paradoxes are derived based on the NFL assumption. It is also shown that significant influence on these probabilities comes from the number of algorithms and the number of optimization problems studied in the comparison. Further, the limiting probabilities when the number of optimization problems goes to infinity are also derived and characterized. Based on the analysis in this article, we recommend adopting "C2" first. If cycle ranking occurs, then consider "C2+." Under this strategy, the ranking of algorithms is certain if cycle ranking does not occur; otherwise, the ranking is still compatible.

In this article, an analogy has been drawn between voting theory and optimization comparison. This is to help users and developers of optimization algorithms understand the cycle ranking and survival of the nonfittest paradoxes more systematically as similar paradoxes have been studied rigorously in voting theory over the past decades. Some of the results in voting paradoxes are extended to optimization theory in this article. In this sense, our findings can be regarded as an application of those results to numerical comparison in optimization and have shown the first justifiable actual use of the IC condition in voting theory.

Optimization attracts significant attention in scientific research and engineering applications. As numerical comparison is key to designing and analyzing optimization and machine learning algorithms, rigorous studies and theories on comparison strategies of these algorithms are necessary and desirable. It is hoped that this article will lay down a steppingstone to guided development of future algorithms. Prior to this, two research directions could arise from this work. First, it is possible to calculate probabilities when the NFL assumption is violated. Second, designing a better comparison strategy than "C2" and "C2+" with a theoretical foundation is also attractive and could offer guidance to benchmarking optimization algorithms more rigorously.

References

[1] K. Schittkowski, C. Zillober, and R. Zotemantel, "Numerical comparison of nonlinear programming algorithms for structural optimization," *Struct. Optim.*, vol. 7, nos. 1–2, pp. 1–19, 1994.

[2] K. Miettinen, M. M. Mäkelä, and J. Toivanen, "Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms," *J. Glob. Optim.*, vol. 27, no. 4, pp. 427–446, 2003.

[3] D. Simon, R. Rarick, M. Ergezer, and D. Du, "Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms," *Inf. Sci.*, vol. 181, no. 7, pp. 1224–1248, 2011.

[4] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, and C. Weihs, "Analyzing the BBOB results by means of benchmarking concepts," *Evol. Comput.*, vol. 23, no. 1, pp. 161–185, 2015.

[5] L. Li, A. A. F. Saldivar, Y. Bai, Y. Chen, Q. Liu, and Y. Li, "Benchmarks for evaluating optimization algorithms and benchmarking MATLAB derivative-free optimizers for practitioners' rapid access," *IEEE Access*, vol. 7, pp. 79657–79670, 2019.

[6] 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.*, 2010, pp. 1689–1696.

[7] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization," School Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, Rep., Nov. 2016.

[8] 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.

[9] Q. Yang *et al.*, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2896–2910, Sep. 2017.

[10] M. Gaviano, D. E. Kvasov, D. Lera, and Y. D. Sergeyev, "Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization," *ACM Trans. Math. Softw.*, vol. 29, no. 4, pp. 469–480, 2003.

[11] Q. Liu *et al.*, "Benchmarking stochastic algorithms for global optimization problems by visualizing confidence intervals," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2924–2937, Sep. 2017.

[12] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *arXiv:1603.08785*, 2016.

[13] T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation: The New Experimentalism*. Heidelberg, Germany: Springer, 2006.

[14] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, *Experimental Methods for the Analysis of Optimization Algorithms*. Heidelberg, Germany: Springer, 2010.

[15] 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, Rep. 201212, Jan. 2013.

[16] M. Gong, Z. Wang, Z. Zhu, and L. Jiao, "A similarity-based multiobjective evolutionary algorithm for deployment optimization of near space communication system," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 878–897, Dec. 2017.

[17] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.

[18] Y. Wang, B. Xu, G. Sun, and S. Yang, "A two-phase differential evolution for uniform designs in constrained experimental domains," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 665–680, Oct. 2017.

[19] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization," Nanyang Technol. Univ., Singapore, Rep., Nov. 2016.

[20] Q. Chen, B. Liu, Q. Zhang, J. J. Liang, P. N. Suganthan, and B. Y. Qu, "Problem definition and evaluation criteria for CEC 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, and Nanyang Technol. Univ., Singapore, Rep., Nov. 2014.

[21] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

[22] M. Yang *et al.*, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 493–505, Aug. 2017.

[23] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[24] 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.

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

[26] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, "IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics," *arXiv e-prints:1810.05281*, Oct. 2018. [Online]. Available: https://arxiv.org/abs/1810.05281
C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, "IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics," *arXiv:1810.05281*, 2018.

[27] *Black Box Optimization Competition (BBComp)*. Accessed: Aug. 22, 2019. [Online]. Available: https://bbcomp.ini.rub.de/

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

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

[30] 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.

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

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

[33] Y. H. Li, Z.-H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Inf. Sci.*, vol. 293, pp. 370–382, Feb. 2015.

[34] 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.

[35] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," Inria, Rocquencourt, France, Rep. RR-6829, Feb. 2010.

[36] 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.

[37] R. Paulavičius, Y. D. Sergeyev, D. E. Kvasov, and J. Žlinskas, "Globally-biased DISIMPL algorithm for expensive global optimization," *J. Glob. Optim.*, vol. 59, nos. 2–3, pp. 545–567, 2014.

[38] D. A. Freedman, R. Pisani, and R. Purves, *Statistics*, 4th ed. New York, NY, USA: Norton, 2007.

[39] M. Taboga, *Lectures on Probability Theory and Mathematical Statistics*, 2nd ed. North Charleston, SC, USA: Amazon CreateSpace, 2012.

[40] C. García-Martínez, F. J. Rodríguez, and M. Lozano, "Arbitrary function optimisation with metaheuristics: No free lunch and real-world problems," *Soft Comput.*, vol. 16, no. 2, pp. 2115–2133, 2012.

[41] N. Hansen, A. Auger, D. Brockhoff, D. Tušar, and T. Tušar, "COCO: Performance assessment," *arXiv:1605.03560*, 2016.

[42] A.-R. Hedar. *Global Optimization Test Problems*. [Online]. Available: http:// www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm

[43] K. J. Arrow, *Social Choice and Individual Values*, 2nd ed. New Haven, CT, USA: Yale Univ. Press, 1963.

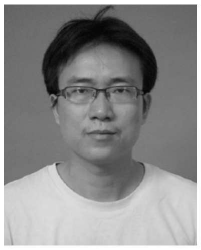
[44] W. V. Gehrlein, *Condorcet's Paradox*, (Theory and Decision Library, Series C: Game Theory, Mathematical Programming and Operations Research). Berlin, Germany: Springer, 2006.

[45] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[46] Q. Liu, W. Chen, Y. Cao, Y. Li, and L. Wang, "Two possible paradoxes in numerical comparisons of optimization algorithms," in *Proc. Int. Conf. Intell. Comput.*, 2018, pp. 681–692.

[47] E. Stensholt, "Circle pictograms for vote vectors," *SIAM Rev.*, vol. 38, no. 1, pp. 96–119, 1996.

[48] W. V. Gehrlein, "Condorcet efficiency of constant scoring rules for large electorates," *Econ. Lett.*, vol. 19, no. 1, pp. 13–15, 1985.



Qunfeng Liu received the B.S. degree in applied mathematics and the M.S. degree in probability and mathematical statistic from the Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in computational mathematics from Hunan University, Changsha, China, in 2011.

He is currently an Associate Professor with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan, China. He have authored more than 40 refereed papers. His current research interests include global optimization, evolutionary computation, and machine learning.

William V. Gehrlein received the M.S. degree in physics and the Ph.D. degree in business administration from Pennsylvania State University, State College, PA, USA, where he taught for 35 years.

He is currently a Professor Emeritus with the University of Delaware, Newark, DE, USA. With the significant help of many coauthors, he has published numerous articles and has written or edited seven books on various topics in the areas of operations management, mathematics, statistics, and voting theory.

Ling Wang received the B.Sc. degree in automation and the Ph.D. degrees in control theory and engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively.

Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a Full Professor in 2008. He has authored five academic books and more than 260 refereed papers. His current research interests include intelligent optimization and production scheduling.

Prof. Wang is a recipient of a number of awards in China. He is the Editor-in-Chief of the *International Journal of Automation and Control*, and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Yuan Yan received the bachelor's degree from the North University of China, Taiyuan, China, in 2018. She is currently pursuing the graduation degree in computer science with the Dongguan University of Technology, Dongguan, China.

Her current research interests include computational intelligence and its applications.

Yingying Cao received the bachelor's and Ph.D. degrees from the School of Mathematics and Computing Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2012, respectively.

From 2011 to 2012, she was a Postgraduate Visiting Researcher with the City University of Hong Kong, Hong Kong. She is currently a Lecturer with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan, China. Her current research interests include computational intelligence, optimization algorithms, and their applications.

Wei Chen received the bachelor's and M.S. degrees from Henan University, Kaifeng, China, in 2006 and 2009, respectively, and the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2014.

He is currently a Research Fellow with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan, China. His current research interests include global optimization, evolutionary computation, quantum computation, and quantum evolutionary computation.

Yun Li (S'87–M'90–SM'17–F'20) received the B.S. degree in electronics science from Sichuan University, Chengdu, China, in 1984, the M.E. degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1987, and the Ph.D. degree in parallel computing and control from the University of Strathclyde, Glasgow, U.K., in 1990.

From 1991 to 2018, he was a Lecturer, a Senior Lecturer, and a Professor with the University of Glasgow, Glasgow, U.K. He is currently the Founding Director of Dongguan Industry 4.0 Artificial Intelligence Laboratory and a Distinguished Professor with the Dongguan University of Technology, Dongguan, China. He has published over 260 papers, one of which is the most popular article every month in the IEEE TCST and another among the top five in the IEEE TSMC-B. His current research interest includes artificial and computational intelligence for Industry 4.0.

Dr. Li is an Associate Editor of the IEEE TEVC, TNNLS and TETCI. He is an FRSA and has co-led the U.K. EPSRC Industrial Systems in the Digital Age Network Plus.