Two Possible Paradoxes in Numerical Comparisons of Optimization Algorithms

Qunfeng Liu[1], Wei Chen[1], Yingying Cao[1], Yun Li[1(✉)], and Ling Wang[2(✉)]

[1] School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China
Yun.Li@ieee.org
[2] Department of Automation, TsingHua University, Beijing, China
wangling@tsinghua.edu.cn

Abstract. Comparison strategies of benchmarking optimization algorithms are considered. Two strategies, namely "C2" and "C2+", are defined. Existing benchmarking methods can be regarded as different applications of them. Mathematical models are developed for both "C2" and "C2+". Based on these models, two possible paradoxes, namely the cycle ranking and the survival of the non-fittest, are deduced for three optimization algorithms' comparison. The probabilities of these two paradoxes are calculated. It is shown that the value and the parity of the number of test problems affect the probabilities significantly. When there are only dozens of test problems, there is about 75% probability to obtain a normal ranking result for three optimization algorithms' numerical comparison, about 9% for cycle ranking, and 16% for survival of the non-fittest.

Keywords: Optimization algorithm · Benchmarking · Paradox
Survival of the non-fittest · Cycle ranking

1 Introduction

Numerous optimization algorithms have been developed to solve the following minimization problem

$$\mathrm{Min} f(x)\ s.t. x \in \Omega \tag{1}$$

where $f(x)$ is the objective function and Ω is the feasible region. If Ω is countable, then (1) is called as a discrete optimization problem, otherwise, a continuous optimization problem. If Ω comes from some constrain conditions, then problem (1) is constrained, otherwise unconstrained. Any maximization problem can be easily modeled as the above minimization problem through replacing $f(x)$ with $-f(x)$.

When the objective function $f(x)$ is nonconvex, problem (1) is often hard to solve. Therefore, in the mathematical programming community, local optimum $\hat{x}$ satisfied

$$f(\hat{x}) \leq f(x),\ \ \forall x \in B_\delta(\hat{x}) \tag{2}$$

is often seeked, where $B_\delta(\hat{x})$ is a neighborhood of $\hat{x}$. The gradient information of $f(x)$ is helpful in algorithm design and mathematical analysis. However, in the evolutionary

D.-S. Huang et al. (Eds.): ICIC 2018, LNCS 10955, pp. 681–692, 2018.
https://doi.org/10.1007/978-3-319-95933-7_77

computation community and the global optimization community, global optimum x^* satisfied

$$f(x^*) \leq f(x), \quad \forall x \in \Omega \tag{3}$$

is investigated. Global optimization is often harder than local optimization, one reason is that there is no information which guides to x^* mathematically.

Therefore, it is necessary to compare optimization algorithms' performance numerically. Firstly, there are many optimization algorithms, and which one is the "best" on some specified functions is often unclear. Numerical comparison can bring helpful insight. Secondly, there is no suitable mathematical convergence for global optimization algorithms, and numerical comparison is the only way to show their efficiency.

Extensive studies have been done on how to compare optimization algorithms numerically, especially on the design of test problems and the development of data analysis methods. Numerous test problems including many sets of benchmark functions [1–4] and hundreds of practical test problems [5–7] have been designed or modeled for numerical comparison of optimization algorithms.

Furthermore, many methods for analyzing experimental data are developed. For instance, the popular performance profiles [8, 9] and data profiles [10, 11] for comparing deterministic optimization algorithms. More methods are developed for comparing stochastic optimization algorithms, e.g., calculating means and standard deviations [12, 13], displaying the history of the found best function values [14, 15], applying statistical inferences [16, 17], employing empirical distribution functions [18–20], and visualizing confidence intervals [18, 19].

However, there are few literatures discuss the selection of comparison strategy, which relates to but is different from the analysis method. When analyze empirical data, if there are only two optimization algorithms, then comparison strategy is unnecessary. However, when algorithms exceed two, there are two basic comparison strategies, namely "C2" strategy and "C2+" strategy. In this paper, "C2" strategy means to compare two algorithms at every match and repeats several matches to obtain an aggregated ranking [12–17]. On the contrary, "C2+" strategy means to rank all algorithms through one or few grand matches [2, 8–11, 18, 19]. In other words, the main difference between "C2" and "C2+" is how many algorithms are compared in each match: two for "C2" while more than two for "C2+".

In this paper, we dedicate to answer the following questions: What is the difference between the ranking results when employing the "C2" or "C2+" strategy? are the ranking results compatible? These questions are well known in the community of political elections and some other social science [21–23]. However, they are unfamiliar in the numerical optimization community, especially the evolutionary computation community.

Through considering the properties and conditions of numerical comparison of optimization algorithms, we will show that the results of "C2" and "C2+" strategy may be different and even incompatible. Specifically, two paradoxes are shown to be possible and their probabilities are calculated to determine the extent of incompatibility.

The rest of this paper is organized as follows. In next Section, the "C2" strategy and the "C2+" strategy are modeled mathematically for convenient analysis. Based on the model, possible paradoxes are deduced in Sect. 3, and probabilities of paradoxes are calculated in Sect. 4. Finally, some conclusions are summarized in Sect. 5.

2 Model the Comparison Strategies

In this section, we describe and model mathematically both the "C2" and the "C2+" strategies.

2.1 The "C2" Strategy

Under this strategy, the whole comparison is divided into several matches (sub-comparisons), and only two algorithms are considered in each match. There are two popular applications of the "C2" strategy, namely the one-play-all comparison and the all-play-all comparison.

One-Play-All Comparison. One-play-all comparison means to compare one special algorithm with all other algorithms, one by one. It is often applied when a new algorithm (including the improvement of an existing algorithm) is proposed. In this case, whether the proposed algorithm performs better than existing similar algorithms is often concerned, and therefore, popular choice is to compare the proposed algorithm with some popular existing algorithms [12, 15, 17, 18, 24, 25]. Different data analysis methods are allowable for applying the one-play-all comparison. For example, the statistical test methods [12, 15, 17, 24, 25], the cumulative distribution function methods [2, 18, 26], and the visualizing confidence intervals method [19].

All-Play-All Comparison. All-play-all comparison means to compare each algorithm with all other algorithms, one by one, and is often called as the Round-robin comparison. It is often applied in algorithms competition [3]. All-play-all comparison can be regarded as a repeated version of one-play-all comparison.

Suppose there are k algorithms, then $k-1$ matches are needed to finish a one-play-all comparison, while $\frac{k(k-1)}{2}$ matches are needed to finish an all-play-all comparison. In other words, k one-play-all comparisons are executed. To aggregate several one-play-all comparisons' ranking results, it is popular to sum up each algorithm's ranking number.

Mathematical Model of "C2" Strategy. Although different data analysis methods are allowable for applying the "C2" strategy, only two ranking results are possible in any match of two algorithms A_1, A_2: A_1 performs better than A_2, or A_1 does not perform better than A_2. In numerical optimization, there are several different standards to judge which algorithm performs better, e.g., convergence, robustness or efficiency. Any single standard is allowable in this paper. For convenience of later discussion, we select one of the most popular standards in global optimization. Specifically, given computational budget, the found best objective function values are employed to determine which algorithm performs better.

Definition 1. Given a fixed computational cost, suppose that f_{min}^{i} is the found minimal objective function value by the algorithm A_i, $i = 1, 2$. Then $A_1 \succsim A_2$ if and only if $f_{min}^{1} \leq f_{min}^{2}$. Moreover, $A_1 > A_2$ if and only if $f_{min}^{1} < f_{min}^{2}$, and $A_1 = A_2$ if and only if $f_{min}^{1} = f_{min}^{2}$.

Based on Definition 1, there are two possible ranking results of the match of algorithms A_1 and A_2 on each test problem: $A_1 \succsim A_2$ or $A_2 \succsim A_1$. Therefore, if m test problems are tested, then there are 2^m possible ranking combinations. This can be regarded as a random sampling with size m from the population with a binomial distribution, which is described in Table 1.

Table 1. Ranking distribution when comparing algorithms A_1, A_2

Ranking	$A_1 \succsim A_2$	$A_2 \succsim A_1$
Probability	p	$1 - p$

In Table 1, p measures the occurrence probability of the event "$A_1 \succsim A_2$", and it is problem dependant. If the test problem biases A_1, then p is close to 1. On the contrary, p is close to 0 if the test problem biases A_2. More details about the parameter p will be discussed in Sect. 4.

Given some test problems, the sampling can be described as a matrix, e.g.,

$$\begin{bmatrix} A_1 & A_2 & A_1 & \dots & A_2 \\ A_2 & A_1 & A_2 & \dots & A_1 \end{bmatrix}_{2 \times m} \tag{4}$$

one column for each problem. In this matrix, the first column $[A_1, A_2]^T$ means A_1 performs better than A_2, and the rest is similar.

2.2 The "C2+" Strategy

This strategy compares all the algorithms at a single or few matches, and it is often used to determine the winner(s) in algorithms competitions [2, 27] or new algorithm proposing [9, 10, 28].

Difference Between "C2+" and "C2". An obvious difference is that "C2+" allows to compare more than two algorithms in a single match while "C2" always compare two algorithms in each match. This brings another difference that "C2" often needs much more matches than "C2+" to finish the whole comparison.

The third but maybe the most important difference between "C2" and "C2+" is that "C2+" adopts statistical aggregation method to obtain all algorithms' ranking results directly. On the contrary, "C2" has to obtain ranking in each match firstly and then aggregate them to obtain a final ranking.

Mathematical Model of "C2+". Suppose there are k algorithms $A_i, i = 1, \ldots, k$. After testing these k algorithms on a test problem, k found best function values f^i_{min} are obtained for any fixed computational budget, where f^i_{min} is the best function value found by $A_i, i = 1, \ldots, k$. Through comparing these function values, a ranking

$$A_{i1} \succsim A_{i2} \succsim \ldots \succsim A_{ik} \tag{5}$$

is obtained, where $(i1, i2, \ldots, ik)$ is a permutation of $(1, 2, \ldots, k)$, and the relationship $\succsim$ is defined in Definition 1. Obviously, different test problem often brings different ranking.

Since there are totally $k!$ possible ranking series, we obtain a multinomial distribution. When $k = 3$, the distribution is summarized in Table 2, where the parameter $p_i, i = 1, \ldots, 6$ and satisfied $\sum_{i=1}^{6} p_i = 1$. When $k > 3$, the distribution is similar as but more complex than that in Table 2.

Table 2. Ranking distribution when comparing algorithm $A_i, i = 1, 2, 3$.

Ranking	$A_1 \succsim A_2 \succsim A_3$	$\cdots\cdots$	$A_3 \succsim A_2 \succsim A_1$
Probability	p_1	$\cdots\cdots$	p_6

If m problems are tested, then it can be regarded as a random sampling with size m from the distribution. For convenience, denote (5) as the following column vector

$$[A_{i1}, A_{i2}, \ldots, A_{ik}]^T.$$

Then a matrix

$$M_1 = \begin{bmatrix} A_{j1} & \cdots & A_{i1} \\ A_{j2} & \cdots & A_{i2} \\ \cdots & \cdots & \cdots \\ A_{jk} & \cdots & A_{ik} \end{bmatrix} \tag{6}$$

can be used to represent the random sampling with size m, each column corresponds the ranking on a test problem. Denote X_i as the number of the i-th ranking, $i = 1, \ldots, k$, then $\sum_{i=1}^{k!} X_i = m$ and the random vector $X = [X_1, X_2, \ldots, X_k]^T$ follows the multinomial distribution with parameter m and $p = [p_1, p_2, \ldots, p_{k!}]$.

Since the sampling matrix (6) of "C2+" includes the ranking information of any pair of algorithms, it contains the sample matrix (4) of "C2". Therefore, it can be adopted to analyze the relationship of ranking results from both "C2+" and "C2". Based on the matrix (6), two paradoxes are presented in Sect. 3, and their probabilities are calculated in Sect. 4 by the help of the multinomial distribution in Table 2.

3 Two Paradoxes

In this section, we adopt the majority rule, which is very popular in numerical comparisons of optimization algorithms [2, 12, 15, 17, 19], to deduce two paradoxes.

Assumption 1 (Majority rule). An algorithm performs better than other algorithms if it can perform better on more test problems than the others do.

For simplicity, only 3 algorithms (A_1, A_2, A_3) are considered in this and next sections, and it is enough for our purpose. In this case, there are 6 possible ranking series, and its ranking distribution is the multinomial distribution listed in Table 2.

Given any problem and the computational budget, test these 3 algorithms on it, and we can obtain a ranking result. According to the discussions in Sect. 2, it can be regarded as a random sampling from Table 2. Specifically, testing on a problem is regarded as a random sampling from the distribution in Table 2. Repeat such process until a desired number of problems are tested, then we obtain a sampling matrix, e.g.,

$$M_1 = \begin{bmatrix} A_2 & A_3 & A_2 & A_3 & A_1 \\ A_1 & A_1 & A_1 & A_2 & A_3 \\ A_3 & A_2 & A_3 & A_1 & A_2 \end{bmatrix}, \quad M_2 = \begin{bmatrix} A_2 & A_3 & A_2 & A_3 & A_1 \\ A_1 & A_1 & A_1 & A_1 & A_2 \\ A_3 & A_2 & A_3 & A_2 & A_3 \end{bmatrix}. \tag{7}$$

Both M_1 and M_2 have 3 rows and 5 columns, indicating that 3 algorithms have been tested on 5 problems. The first column means $A_2 \succsim A_1 \succsim A_3$, i.e., A_2 performs better than or similarly as A_1 on this problem, and A_1 performs better than or similarly as A_3 on this problem. The rest is similar.

Paradox from "C2": Cycle Ranking. Suppose that there are totally 5 test problems, and the ranking results are given by the matrix M_1 in (7). If we adopt the "C2" strategy to compare these 3 algorithms, then $A_2 \succsim A_1$ since 3 problems bias A_2 and 2 bias A_1. Similarly, $A_1 \succsim A_3$ since 3 problems bias A_1 and 2 bias A_3, $A_3 \succsim A_2$ since 3 problems bias A_3 and 2 bias A_2. As a result, we obtain a cycle ranking $A_2 \succsim A_1 \succsim A_3 \succsim A_2$, and we cannot tell which algorithm performs the best.

The cycle ranking paradox is also called as Condorcet paradox [29, 30], which is very popular in voting theory and was found firstly by Marquis de Condorcet in the 18th century when he investigated a voting system. In next section, we will discuss the occurrence probability of cycle ranking, and how the number of test problems affect the probability.

Paradox from "C2+": Survival of the Non-fittest. Suppose that the ranking results are given by the matrix M_2 in (7). If we adopt the "C2+" strategy to compare these 3 algorithms, then A_2 or A_3 is the winner since both perform the best on 2 problems while A_1 only performs the best on the fifth problem.

However, if we compare A_1 and A_2 alone, then A_1 performs better than A_2 on 3 problems (2nd, 4th and 5th) while worse only on 2 problems (1st and 3rd). Therefore, A_1 performs better than A_2 on the whole test set. Similarly, A_1 performs better than A_3 on the whole test set, too. Therefore, A_1 is the winner of the "C2" strategy.

In other words, the winner of the "C2+" strategy do not perform well in "C2" comparisons. Such phenomenon is called as the survival of the non-fittest in this paper,

which is also called as the Borda paradox, it was also found in the 18th century [30]. We will discuss its occurrence probabilities in next section and show how the number of test problems affect the probability.

4 Probability Analysis

To calculate the occurrence probabilities of cycle ranking and survival of the non-fittest, the parameters in Table 2 should be determined firstly. In this paper, we adopt the following No Free Lunch (NFL) assumption.

Assumption 2 (The NFL assumption). For any given 3 optimization algorithms and any given test problem, all 6 possible rankings of these algorithms on this problem are equally likely, i.e., $p_i = \frac{1}{6}$, $i = 1, \ldots, 6$ in Table 2.

The NFL assumption is a direct application of the No Free Lunch theorem in optimization [31]. According to the NFL theorem, if the test problems are selected randomly from all possible problems, then the average performance of any algorithm is equal. In other words, any ranking in Table 2 has the same occurrence probability, and therefore $p_i = \frac{1}{6}$.

If these 3 algorithms have been tested on m problems, and the i-th ranking in Table 2 has appeared X_i times, $i = 1, \ldots, 6$, then the random vector $X = [X_1, X_2, \ldots, X_6]^T$ satisfies the following multinomial distribution.

$$P(X_1 = x_1, \ldots, X_6 = x_6) = \frac{m!}{x_1! \ldots x_6!} \frac{1}{6^m}, \tag{8}$$

where $x_i \in [0, m], i = 1, \ldots, 6$ and satisfy $\sum_{i=1}^{6} x_i = m$. In this paper, $P(A)$ is denoted as the probability of a random event A.

Then we calculate the occurrence probabilities of cycle ranking and survival of the non-fittest based on the NFL assumption.

4.1 Division of the Sample Space

Firstly, we give some definitions below, which define possible random events when benchmarking optimization algorithms.

Definition 2. When we adopt the "C2" strategy, if the ranking of these 3 algorithms form a cycle, i.e., $A_1 \succsim A_2 \succsim A_3 \succsim A_1$ or $A_3 \succsim A_2 \succsim A_1 \succsim A_3$, then we say that the random event of cycle ranking happens, or random event C happens for simplicity.

Definition 3. If the final winner of the "C2+" strategy is not the final winner of the "C2" strategy, then we say that the random event of survival of the non-fittest happens, or random event S happens for simplicity.

Definition 4. The final winner of the "C2+" strategy is exactly the final winner of the "C2" strategy, then we say that the random event N happens.

Then we have the following theorem, whose proof is not presented in this paper due to the limitation of space.

Theorem 1. Only three random events C, S, N are possible in the numerical comparisons of three optimization algorithms.

Theorem 1 implies that

$$\mathrm{P}(C) + P(S) + P(N) = 1. \tag{9}$$

Therefore, in later subsections, we will calculate the probabilities of $P(C)$ and $P(S)$, and then calculate the probability of $P(N)$ indirectly.

4.2 Probabilities of the Random Event C

Theorem 2. When comparing 3 optimization algorithms on m test problems, the probability of random event C is

$$\mathrm{P}(C) = \frac{1}{6^m}\left(2\sum\nolimits_{\{x_i\}\in C_1}\frac{m!}{x_1!\ldots x_6!} - \sum\nolimits_{\{x_i\}\in C_2}\frac{m!}{x_1!\ldots x_6!}\right), \tag{10}$$

where C_1, C_2 are determined as follows.

$$C_1: \begin{cases} x_1+\ldots+x_6 = m \\ x_1+x_2+x_5 \geq \frac{m}{2} \\ x_1+x_3+x_4 \geq \frac{m}{2} \\ x_4+x_5+x_6 \geq \frac{m}{2} \\ x_i = 0,1,\ldots,m, i = 1,\ldots,6, \end{cases} \quad C_2: \begin{cases} x_1+\ldots+x_6 = m \\ x_1+x_2+x_5 = \frac{m}{2} \\ x_1+x_3+x_4 = \frac{m}{2} \\ x_4+x_5+x_6 = \frac{m}{2} \\ x_i = 0,1,\ldots,m, i = 1,\ldots,6. \end{cases} \tag{11}$$

Theorem 2's proof is omitted in this paper partly due to the limitation of space.

It is clear from (11) that C_2 is the border of C_1, and it is empty when m is odd. Furthermore, $\log_{m\to\infty} P(C_2) = 0$. Therefore, in the literatures of calculating Condorcet paradox's probabilities, only odd m are often considered [30].

Given the number of test problems m, we can calculate the probability $P(C)$ through formula (10) in Theorem 2. Figure 1 shows the numerical results of $P(C)$ for $m = 1, 2, \ldots, 100$. From Fig. 1 we found that $P(C) = 0$ when $m = 1$ and $P(C) = 0.5$ when $m = 2$. As m increases, $P(C)$ changes zigzagged, and there are two opposite trends of $P(C)$. When m is even, $P(C)$ decreases from 0.5 to near 0.13 as m increases. On the contrary, when m is odd, $P(C)$ increases from 0 to near 0.9 as m increases. These results are the same as those reported in [30] when m is odd. However, we provide the probabilities when m is even, which bring helpful insight.

Based on these calculations, we conclude that odd number of test problems is a good choice for numerical comparisons of optimization algorithms, since it decreases the occurrence probability of cycle ranking. Under this choice, the occurrence probability of cycle ranking is less than 9%. In other words, cycle ranking is only occasionally happened.

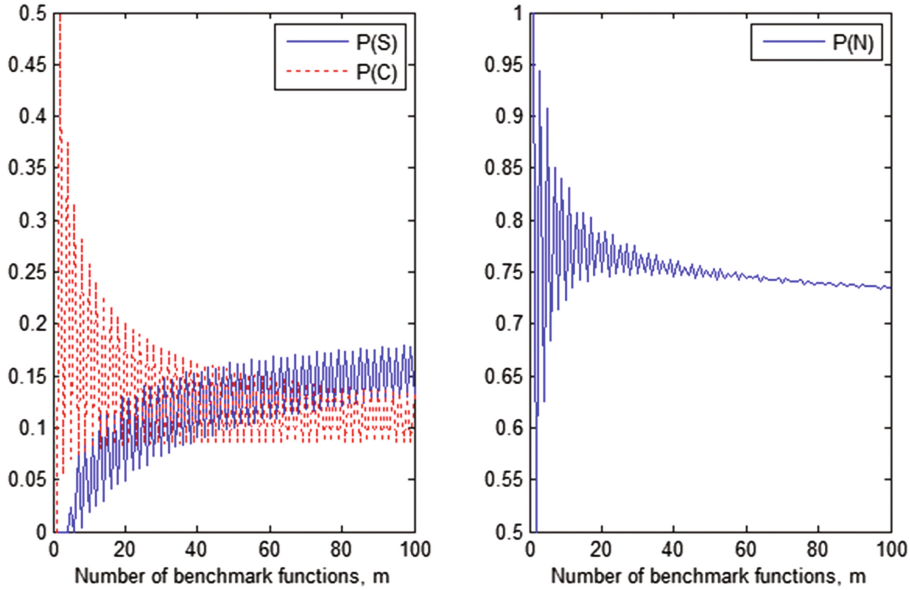

Fig. 1. Probabilities of $P(C), P(S)$ and $P(N)$.

4.3 Probabilities of the Random Event S

Although there are several probability calculations [30] of the Condorcet paradox (i.e.,$P(C)$), to our knowledge, there is no published results of $P(S)$. In [32, 33], the occurrence probabilities of the strict Borda paradox and the strong Borda paradox were analyzed, however, which are significantly different from $P(S)$.

We present the theoretical formula of $P(S)$ as the following theorem, whose proof is not included here due to the limitation of space.

Theorem 3. When comparing 3 optimization algorithms on m test problems, the probability of random event S is given by

$$\mathrm{P}(S) = \frac{3}{6^m}\left(\sum\nolimits_{\{x_i\}\in S_1} \frac{m!}{x_1!\ldots x_6!} - \sum\nolimits_{\{x_i\}\in S_2} \frac{m!}{x_1!\ldots x_6!}\right), \tag{12}$$

where the dominant S_1, S_2 are defined as follows.

$$S_1 : \begin{cases} x_1+\ldots+x_6 = m \\ x_1+x_2+x_5 > \frac{m}{2} \\ x_1+x_2+x_3 > \frac{m}{2} \\ \max(x_3+x_4, x_5+x_6) > x_1+x_2 \\ x_i = 0,1,\ldots,m, i = 1,\ldots,6, \end{cases} \quad S_2 : \begin{cases} x_1+\ldots+x_6 = m \\ x_1+x_2+x_5 = \frac{m}{2} \\ x_1+x_2+x_3 > \frac{m}{2} \\ x_1+x_3+x_4 > \frac{m}{2} \\ x_5+x_6 > x_1+x_2 \\ x_5+x_6 > x_3+x_4 \\ x_i = 0,1,\ldots,m, i = 1,\ldots,6. \end{cases} \tag{13}$$

Given the number of test problems m, we can calculate the probability $P(S)$ through formula (12) in Theorem 3. Figure 1 shows the numerical results of $P(S)$ for $m = 1, 2, \ldots, 100$.

From Fig. 1 we found that $P(S) = 0$ until $m \geq 5$, and changes zigzagged as m increases. Roughly speaking, $P(S)$ increases from almost 0 to about 0.18 as $m = 5, 7, 9, \ldots$ increases, and increases from 0 to about 0.13 as $m = 6, 8, 10, \ldots$ increases.

To conclude, the probability $P(S)$ is larger than $P(C)$. What is more, $P(S)$ increases as m increases, whatever m is odd or even. Therefore, survival of the non-fittest is not too rare, and should be taken it seriously when adopting the "C2+" strategy.

4.4 Probabilities of the Random Event *N*

Given the number of test problems m, we can calculate the probability $P(N)$ according to (9), where $P(C)$ and $P(S)$ are calculated through formulas (10) and (12), respectively. Figure 1 shows the numerical results of $P(N)$ for $m = 1, 2, \ldots, 100$.

From Fig. 1 we found that $P(N)$ zigzagged violently when m is small and decreases roughly as m increases. Finally, $P(N)$ becomes less than 0.75 when $m > 80$.

5 Conclusions and Future Work

Numerical comparisons of optimization algorithms are analyzed through considering is as a selection, where optimization algorithms are regarded as candidates while test problems are regarded as voters. Two popular comparison strategies of benchmarking optimization algorithms are discussed, namely the "C2" strategy and the "C2+" strategies.

It was shown that two paradoxes, cycle ranking and survival of the non-fittest, are possible. Their probabilities are calculated when only three optimization algorithms are compared. It was shown that the value and the parity of the number of test problems m affect the probabilities significantly.

To decrease the probability of paradox, we suggest adopting an odd m test problems to implement a "C2" comparison, while an even m for a "C2+" comparison. However, our calculations show that it is impossible to eliminating both paradoxes except $m = 1$, which is impractical.

Roughly speaking, there is about 9% probability to find a cycle ranking when adopting the "C2" strategy, about 16% to find a survival of the non-fittest when adopting "C2+", and about 75% to obtain a normal ranking result for three optimization algorithms' numerical comparison. Therefore, "C2" is more suitable than "C2+" from the view of bringing less probability of paradox.

Although only three optimization algorithms are considered in this paper, the paradoxes happen in more general cases, and the probability calculation are ongoing. Several relevant issues are necessary to investigate.

Acknowledgment. This work was supported by National Key R&D Program of China (No. 2016YFD0400206), NSF of China (No. 61773119) and NSF of Guangdong Province (No. 2015A030313648).

References

1. Gaviano, M., Kvasov, D., Lera, D., Sergeyev, Y.D.: Algorithm 829: software for generation of classes of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. **9**, 469–480 (2003)
2. Hansen, N., Auger, A., Ros, R., Finck, S. and Pošík P.: Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: Proceedings of the 12th annual conference companion on genetic and evolutionary computation, pp. 1689–1696 (2010)
3. Awad, N.H., Ali, M.Z., Liang, J.J., Qu, B.Y., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC2017 special session and competition on single objective bound constrained real parameter numerical optimization. Nanyang Technological University, Singapore, Technical report, November 2016
4. Hansen, N., Auger, A., Mersmann, O., Tušar, T., Brockhoff, D.: Coco: A platform for comparing continuous optimizers in a black-box setting. ArXiv e-prints arXiv:1603.08785 (2016)
5. Gong, M., Wang, Z., Zhu, Z., Jiao, L.: A similarity-based multiobjective evolutionary algorithm for deployment optimization of near space communication system. IEEE Trans. Evol. Comput. **21**, 878–897 (2017)
6. Valle, Y., Venayagamoorthy, G.K., Mohagheghi, S., Hernandez, J.-C., Harley, R.G.: Particle swarm optimization: Basic concepts, variants and applications in power systems. Inf. Sci. **12**, 171–195 (2008)
7. Wang, Y., Xu, B., Sun, G., Yang, S.: A two-phase differential evolution for uniform designs in constrained experimental domains. IEEE Trans. Evol. Comput. **21**, 665–680 (2017)
8. Dolan, E.D., Moŕe, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
9. Liu, Q., Zeng, J.: Global optimization by multilevel partition. J. Glob. Optim. **61**, 47–69 (2015)
10. Liu, Q., Zeng, J., Yang, G.: MrDIRECT: a multilevel robust DIRECT algorithm for global optimization problems. J. Glob. Optim. **62**, 205–227 (2015)
11. Moŕe, J., Wild, S.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**, 172–191 (2009)
12. Omidvar, M.N., Yang, M., Mei, Y., Li, X., Yao, X.: Dg2: A faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans. Evol. Comput. **21**, 929–942 (2017)
13. Yang, M., Omidvar, M.N., Li, C., Li, X., Cai, Z., Kazimipour, B., Yao, X.: Efficient resource allocation in cooperative co-evolution for large-scale global optimization. IEEE Trans. Cybern. **21**, 493–505 (2017)
14. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. **16**, 210–224 (2012)
15. Qin, Q., Cheng, S., Zhang, Q., Li, L., Shi, Y.: Particle swarm optimization with interswarm interactive learning strategy. IEEE Trans. Cybern. **46**, 2238–2251 (2015)
16. Gong, Y.-J., Li, J.-J., Zhou, Y., Li, Y., Chung, H.S.-H., Shi, Y.-H., Zhang, J.: Genetic learning particle swarm optimization. IEEE Trans. Cybern. **46**, 2277–2290 (2016)
17. Yang, Q., Chen, W.-N., Gu, T., Zhang, H., Deng, J.D., Li, Y., Zhang, J.: Segment-based predominant learning swarm optimizer for large-scale optimization. IEEE Trans. Cybern. **47**, 2896–2910 (2017)
18. Liu, Q.: Order-2 stability analysis of particle swarm optimization. Evol. Comput. **23**, 187–216 (2015)

19. Liu, Q., Chen, W.-N., Deng, J.D., Gu, T., Zhang, H., Yu, Z., Zhang, J.: Benchmarking stochastic algorithms for global optimization problems by visualizing confidence intervals. IEEE Trans. Cybern. **47**, 2924–2937 (2017)
20. Hansen N., Auger A., Brockhoff D., Tušar D., and Tušar T.: Coco: Performance assessment. ArXiv e-prints arXiv:1605.03560 (2016)
21. Maassen, H., Bezembinder, T.: Generating random weak orders and the probability of a Condorcet winner. Soc. Choice Welf. **19**, 517–532 (2002)
22. Dwork C., Kumar R., Naor M., and Sivakumar D.: Rank aggregation methods for the web. In: Proceedings of the 10th International Conference on World Wide Web, pp. 613–622. ACM (2001)
23. Cucuringu, M.: Sync-rank: Robust ranking, constrained ranking and rank aggregation via eigenvector and SDP synchronization. IEEE Trans. Netw. Sci. Eng. **3**, 58–79 (2016)
24. Li, Y.H., Zhan, Z.-H., Lin, S.J., Zhang, J., Luo, X.N.: Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems. Inf. Sci. **293**, 370–382 (2015)
25. Chen, W.-N., Zhang, J., Lin, Y., Chen, N., Zhan, Z.-H., Chung, H.S.-H., Li, Y., Shi, Y.-H.: Particle swarm optimization with an aging leader and challengers. IEEE Trans. Evol. Comput. **17**, 241–258 (2013)
26. Liu, Q., Wei, W., Yuan, H., Zhan, Z.-H., Li, Y.: Topology selection for particle swarm optimization. Inf. Sci. **363**, 154–173 (2016)
27. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**, 1247–1293 (2013)
28. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žlinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. J. Glob. Optim. **59**, 545–567 (2014)
29. Deemen, A.V.: On the empirical relevance of condorcet's paradox. Pub. Choice **158**, 311–330 (2014)
30. Gehrlein, W.V.: Condorcet's Paradox. Springer, Berlin (2006)
31. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
32. Diss, M., Gehrlein, W.V.: Borda's Paradox and weighted scoring rules. Soc. Choice Welf. **38**, 121–136 (2012)
33. Gehrlein, W.V., Lepelley, D.: On the probability of observing Borda's paradox. Soc. Choice Welf. **35**, 1–23 (2015)