Information Sciences 277 (2014) 656–679

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms

Niki Veček [*], Marjan Mernik, Matej Črepinšek

University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, 2000 Maribor, Slovenia

ARTICLE INFO

Article history:
Received 2 September 2013
Received in revised form 13 February 2014
Accepted 25 February 2014
Available online 13 March 2014

Keywords:
Evolutionary algorithm
Computational experiment
Null hypothesis significance testing
Chess rating
Ranking

ABSTRACT

The Null Hypothesis Significance Testing (NHST) is of utmost importance for comparing evolutionary algorithms as the performance of one algorithm over another can be scientifically proven. However, NHST is often misused, improperly applied and misinterpreted. In order to avoid the pitfalls of NHST usage this paper proposes a new method, a Chess Rating System for Evolutionary Algorithms (CRS4EAs) for the comparison and ranking of evolutionary algorithms. A computational experiment in CRS4EAs is conducted in the form of a tournament where the evolutionary algorithms are treated as chess players and a comparison between the solutions of two algorithms on the objective function is treated as one game outcome. The rating system used in CRS4EAs was inspired by the Glicko-2 rating system, based on the Bradley–Terry model for dynamic pairwise comparisons, where each algorithm is represented by rating, rating deviation, a rating/confidence interval, and rating volatility. The CRS4EAs was empirically compared to NHST within a computational experiment conducted on 16 evolutionary algorithms and a benchmark suite of 20 numerical minimisation problems. The analysis of the results shows that the CRS4EAs is comparable with NHST but may also have many additional benefits. The computations in CRS4EAs are less complicated and sensitive than those in statistical significance tests, the method is less sensitive to outliers, reliable ratings can be obtained over a small number of runs, and the conservativity/liberality of CRS4EAs is easier to control.

1. Introduction

Computational experiments are essential within the field of meta-heuristics [8,9]. New meta-heuristic algorithms with different exploration and exploitation abilities [16] have been proposed and their effectiveness needs to be displayed on artificial or real problems by theoretical analysis or empirical testing. Empirical testing when performing computational experiments is preferred within meta-heuristics as theoretical analysis is based on mathematical theorems and models and can be more difficult to understand. Essentially, the contributions of a new meta-heuristic algorithm should be evaluated scientifically and reported objectively. As already noted by Barr et al. [5] this is not always the case for heuristic methods. We have also observed the following problems in some meta-heuristic literature: newly developed algorithms are not described in sufficient detail nor are publicly available, the experimental settings are only partially documented thus preventing an exact

* Corresponding author. Tel.: +386 40570780.
E-mail addresses: niki.vecek@uni-mb.si (N. Veček), marjan.mernik@uni-mb.si (M. Mernik), matej.crepinsek@uni-mb.si (M. Črepinšek).

http://dx.doi.org/10.1016/j.ins.2014.02.154
0020-0255/© 2014 Elsevier Inc. All rights reserved.

replication of an experiment and fair comparison, experiments are not conducted under the same or stricter conditions as the original ones, the experimental results are not always rich enough with respect to statistics, improper use or even the absence of statistical methods, statistical significance is omitted, and the derived conclusions are too general and unsupported by experiments - these problems are also discussed in [25]. The aforementioned problems are not only typical for the field of meta-heuristics. For example, Kitchenham et al. [61] mentioned similar problems within the fields of software engineering and medicine, where it was reported that 40% of the examined medical publications (in total 164) had statistical errors; and even in another study half of the publications were impossible to evaluate due to insufficient details of the statistical methods used, whilst nearly one third of the publications contained inappropriate usages of statistics. If researchers have problems with the proper usages of statistical methods within an established discipline such as medicine, then it is unsurprising that researchers have problems in much younger disciplines such as software engineering [23] and meta-heuristics. This problem is vividly described in [61] as: *"Some problems with statistics arise because there are methodological difficulties applying standard statistical procedures to software experiments. Nonetheless, the majority of problems result from lack of statistical expertise in the empirical research community."*

As the main reason for the plethora of statistical errors is inadequate understanding of statistical methods, the question is how to alleviate this problem during computational experiments within the field of meta-heuristics, in particular evolutionary computations [4,26]. Better training might be an obvious answer but this solution would only come into effect long-term. In the meantime the proper designing, executing, and reporting of computational experiments will remain a crucial task. This paper describes a possible alternative solution in which knowledge of the Null Hypothesis Significance Testing (NHST) is not essential when comparing evolutionary algorithms. A novel Chess Rating System for Evolutionary Algorithms (CRS4EAs) method is proposed for the comparisons and rankings of evolutionary algorithms as a feasible alternative to NHST. Roughly speaking, the idea of rating chess players is incorporated within our method. Nowadays, chess ratings are very reliable regarding chess players' strengths [42,44]. Can chess ratings be used to measure the strengths of evolutionary algorithms, and hence used for algorithms' comparisons? This paper shows that CRS4EAs can indeed be used as an alternative for algorithm rankings and comparisons. The following analogies are used in CRS4EAs: 'chess player = evolutionary algorithm', 'chess game = searching for the best solution for a given problem using a pre-specified number of fitness evaluations', and 'chess tournament = comparison of evolutionary algorithms on the benchmark test suite using a pre-specified number of independent runs'. On a benchmark suite containing 20 standard numerical optimisation functions, 16 different evolutionary algorithms and their variants are compared using the newly-proposed CRS4EAs method, as well as classical NHST. It was shown that CRS4EAs has many similarities to NHST, whilst on the other hand it also has several additional benefits such as: (1) robustness to outliers, (2) a controllable mechanism for conservativity/liberality, (3) is simple to apply without the danger of misuse or misunderstanding, (4) has a simple experimental design and (5) accurate estimate of an algorithm's performance is achieved over a small number of runs (25 or less). The limitations of CRS4EAs can be seen in: (1) the number of total runs (i.e. games) depends on the number of algorithms compared, (2) the ranking list changes with each new added algorithm, (3) the means and other statistical values reported within existing publications cannot simply be re-used for algorithm comparisons, (4) a ranking-list with a slightly different experiment results must not be used identically for further comparing a new algorithm, or (5) the handling of the outliers might be less appropriate in situations that demand high success rates in all independent runs due to one-to-one run (game) comparison.

The main contributions of this paper are: the proposal of a new method CRS4EAs for comparing and ranking evolutionary algorithms, and the presentation of a computational experiment that empirically confirmed that CRS4EAs can be used for comparing evolutionary algorithms and is comparable with NHST but also contains several other benefits. We wish CRS4EAs to be used not only amongst researchers but to also assist reviewers in evaluating newly-developed algorithms.

This paper is organised as follows. Section 2 briefly reviews the classical method of evolutionary algorithms' comparisons by using various statistical methods. Section 3 presents an introduction to those chess rating systems currently in use by different Chess Federations with an emphasis on the Glicko and the Glicko-2 rating systems. The novel CRS4EAs method for ranking evolutionary algorithms is introduced in Section 4. Extensive experimental results after comparing the two approaches, CRS4EAs and NHST, for evolutionary algorithms' comparisons are presented in Section 5, followed by a discussion in Section 6. The paper concludes in Section 7 with a brief statement on proposed future directions.

2. Background

A common method for comparing the performances of different evolutionary algorithms is modern statistical hypothesis testing, which was developed by Fisher [33,34,36], and Neyman and Pearson [71]. After stating a null hypothesis that there is no difference in the results from the experiment, an appropriate statistical test is used to check whether the null hypothesis could be rejected or not. Any rejection of the null hypothesis would lead to acceptance of the alternative hypothesis that states that there would be differences in the results of the experiment. Hypothesis testing requires the specification of an acceptable level of statistical error. When the null hypothesis is falsely rejected a Type I error is made, and if the test fails to reject a false null hypothesis a Type II error is made. The probabilities of both errors are also known as α and β. The probability of correctly rejecting a false null hypothesis, i.e. $1 - \beta$, is the power of statistical test.

Usually, the goal of statistical inference in evolutionary algorithms is to compare multiple algorithms on multiple data sets and there are several statistical techniques for doing this that are divided into two types – parametric and non-parametric tests [82]. The difference between these two types is in the properties of the data sets. The first one assumes that the data emanates from a type of probability distribution and makes inferences about the parameters of the distribution, whilst the second one does not make such assumptions. If the stated assumptions are indeed correct, the parametric tests have greater power and should therefore be used. Otherwise, the results of the parametric test can be misleading and the non-parametric test would be a more appropriate choice. Statistical tests only provide information that there are differences in the results but post hoc analysis [7] is required for further investigation. Post hoc tests are also known as posteriori tests as their main goal is to look at the data after the experiment has been conducted and find patterns or relationships that were unspecified as a priori. Such patterns would otherwise remain undetected.

A common statistical method for comparing multiple algorithms on multiple data sets is ANOVA or Analysis of Variance [35]. ANOVA is the generalisation of a t-test [86] for more than two samples and is a parametric test. The null hypothesis states that all results have the same mean, and that the observed differences between the means are due to chance. The total variability is divided into the variabilities between algorithms, variabilities between data sets and error variabilities. If an error variability is significantly smaller than the variability between algorithms the null hypothesis is rejected, thus implying that there were differences in the results of the observed algorithms. The rejection of a null hypothesis is followed by post hoc analysis. The more common post hoc tests used with ANOVA are the Tukey test [89] and the Dunnett test [22].

The second method for multiple comparison is the Friedman test [37,38], which is a non-parametric alternative to ANOVA. The Friedman test ranks the algorithms by data sets. Let N be the number of problems, k the number of algorithms, and $r_{i,j}$ the rank of the jth algorithm on the ith problem, then the average rank of jth algorithm is $R_j = \frac{1}{N}\sum_i^N r_{i,j}$. The null hypothesis states that the observed algorithms provide the same result, and therefore have the same rank. The Friedman statistic (Eq. (1)) is distributed according to chi-square with $(k-1)$ degrees of freedom, when N and k are big enough (as a rule of thumb, $N > 10$ and $k > 5$).

$$\chi_F^2 = \frac{12N}{k(k+1)}\left(\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right) \tag{1}$$

Iman and Davenport [58] derived a less conservative alternative with statistic distributed according to F-distribution (Eq. (2)) with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom.

$$F_F^2 = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{2}$$

Critical values can be found in corresponding statistical tables [66]. The rejection of the null hypothesis is again followed by the post hoc analysis. The more common post hoc test for the Friedman statistic is the Nemenyi test [68]. The goal of the Nemenyi test is to find which of the k average ranks of the observed algorithms differ for the critical difference $CD = q_\alpha\sqrt{k(k+1)/6N}$. The critical values q_α are based on the Studenised range statistic with ∞ degrees of freedom divided by $\sqrt{2}$, and can again be found in the statistical tables [66]. The more commonly used post hoc methods are also the Bonferroni–Dunn test [21], the Holm procedure [54], the Shaffer procedure [81], the Hochberg procedure [51], and the Hommel procedure [55], that sequentially test the hypotheses ordered by their significances. These methods are used for comparison using a control algorithm - that is in those cases when a new algorithm is suggested and compared with other $k-1$ similar approaches. For each of the $k-1$ comparisons, the p_i values are calculated and ordered, so that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_{k-1}$. The test statistics for comparing the i-th and j-th algorithms during these procedures is $z = (R_i - R_j)/\sqrt{k(k+1)/6N}$. The Holm step-down procedure, for example, starts with the most significant (smallest) p value, i.e. p_1. If p_1 is smaller than $\alpha/(k-1)$, the first hypothesis is rejected. If the second smaller p value p_2 is smaller than $\alpha/(k-2)$, the second hypothesis is also rejected. This procedure continues by rejecting those hypotheses for which the p_i values are smaller than $\alpha/(k-i)$, until the spot j is found where this is not the case. All the other hypotheses for which the p_i values are greater than $\alpha/(k-i)$ are retained. A common method for also comparing two algorithms over multiple data sets is the Wilcoxon signed-rank test [91]. The Wilcoxon test ranks the differences in performances of two algorithms for each problem by absolute values and then compares the ranks between the positive and negative differences. The difference between two algorithms on the ith problem is denoted by d_i. The R^+ is the sum of the ranks for problems on which the second algorithm outperformed the first (where the detected difference in performances is positive), the R^- is the sum of the ranks for problems on which the first algorithm outperformed the second (where the detected difference in performances is negative). The ranks of differences that equal 0 are split evenly amongst the sums (Eq. (3)).

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \text{ and } R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{3}$$

The statistics $z = (T - N(N+1)/4)/\sqrt{(N(N+1)(2N+1)/24}$, where T is the smaller of the sums, $T = \min(R^+, R^-)$, is approximately normally distributed for a N of more than 25. Differences d_i should not be rounded to one or two decimals as that would decrease the power of the test [18,19]. The Wilcoxon signed rank test is a widely used method (e.g., [12,13,47,57,59,69]) for comparing two algorithms over multiple data sets.

There are also a few more advanced alternatives to the Friedman test such as the Multiple Sign-test [84], the Friedman Aligned Ranks [52], or the Quade test [78]. These methods are described in detail by García et al. [39], with several suggestions for post hoc analysis such as the Holland procedure [53], the Finner procedure [32], the Rom procedure [77], or the Li procedure [65]. All these tests are used during comparisons using the control algorithm. The detected differences in the powers of these post hoc tests were observed as being small [39], though in some cases the powers differed depending on the circumstances.

The choice of test depends upon what it is intended to study. As Demšar [18] showed, non-parametric tests are safer and more appropriate than parametric tests for comparisons between two or more algorithms on multiple data sets. Due to its theoretical and practical advantages the Friedman test should be preferred over ANOVA, as has already been the case in many works (e.g., [1,63,67]). Any further analysis using post hoc tests is for the researcher to decide. The Nemenyi test is an appropriate choice for $k \times k$ pairwise comparison, whilst for comparison with a control algorithm, the Holm procedure is powerful enough and easier to apply. For the fuzzy-environment Alizadeh et al. [2] developed fuzzy-hypothesis testing based on the likelihood ratio test. Bayesian hypothesis testing is used for more theoretical hypothesis testing, where direct probability calculations are allowed. However, Bayesian hypothesis testing is not widely used because it requires prior probability of hypotheses, which is usually undetermined objectively [56]. Shilane et al. [83] presented bootstrap-based multiple hypothesis testing.

However, significance tests should be conducted with great care and by comprehensive understanding of statistical methods. This is because the powers of post hoc tests are dependent and because these tests are often misused, many statisticians [14,49,80] believe that significance tests should not be performed at all. Levine et al. [64] highlighted four of the more common criticisms of NHST: sensitivity to sample size, the null hypothesis is almost always false (at least in social sciences), unacceptable Type II error rates, and allegations that the null-hypothesis testing is frequently misunderstood and abused. The problems with such analyses are also in the measurements from different data sets such as questions about the normalities of their distributions and the homogeneity of variance. Many statisticians [41,61] have also noticed a problem in the way statistical inference is conducted, stating that many researchers (in all fields, not just evolutionary algorithms) have problems performing empirical studies, make misinterpretations or are victims of specific flaws in the logic of synthesis. But, do we have an alternative approach?

3. Chess rating system

This section introduces the chess rating system for comparing the absolute powers of the real chess players and the formulas for updating their powers. Chess is a strategic game [72] of (usually) two players with three possible outcomes – win, lose, or draw. A set of multiple games between multiple players is called a tournament and happens periodically, e.g. under the Chess Federation organisation. Each such organisation has a list of its players, ranked according to their rating points. Players gain rating points by winning games during organised tournaments.

Different chess federations have their own rules and formulas for updating their players' ratings. In 1948 Hoesslinger designed the Ingo system [50], which then inspired many other rating systems. The more common chess rating system in use today is Elo [27], the formula of which is derived from the Bradley–Terry model [10] for pairwise comparison. Although, the Elo rating system [27] is famous for its simplicity and wide usage, it has a few drawbacks such as properly setting the value of the factor that controls the change in rating, an inaccurate distribution model, or unreliable rating. The main concern about unreliable rating is the possibility of a player winning the game but losing rating points, or losing the game yet gaining rating points. The second problem with unreliable rating shows up in games between players with the same ratings, when one of them has not played for years and the other plays constantly. The winner would gain the same amount of points the loser would lose, however, that should not be the case. A more unreliable rating for a player who has not played for years, and a more reliable rating for a player who plays all the time is expected. Therefore, if the first player wins, it is expected that his rating would go up more than would the rating of the second player go down.

Furthermore, nothing is divulged about a player's gaming behaviour or the reliability of his/her power, by knowing only his/her rating. This is why Glickman [42] introduced a new chess rating system in 1995, which is used today by several organisations such as the prestigious Australian Chess Federation. In the Glicko system [43] each player gets his/her rating R and rating deviation value RD. The rating deviation denotes how reliable a player's rating is. If a player has a small rating deviation he/she plays quite often and his/her rating is reliable. If the player's rating deviation is high, his/her rating is unreliable. Glickman suggested using a more informative way of summarising a player's strength by reporting a 95% confidence interval. It can be said with 95% probability that the player's rating R is within an interval $[R - 2RD, R + 2RD]$. The empirical 68–95–99.7 rule [73] – which states that the 68% of the values lie within one standard deviation of the mean, at 95% of the values lie within two standard deviations of the mean, and at 99.7% of the values lying within three standard deviations of the mean - can be used. It can be said with 99.7% probability that the player's rating R is within an interval $[R - 3RD, R + 3RD]$.

In 2012, Glickman introduced an improved version of the Glicko rating system, that is Glicko-2 [46]. Players in the Glicko-2 rating system are now presented with rating R, rating deviation RD, and rating volatility σ. The rating volatility indicates the degree of expected fluctuation in a player's rating. High rating volatility indicates that the player's rating has suddenly increased or decreased, whilst it was constant before that. Low rating volatility informs that the player's rating is reliable and expected. The reliable and expected rating means that we can trust the player's absolute power. If two players have reliable and expected ratings and the first has a high rating while the second has a low rating, it is expected with high probability that the former will outperform the latter.

Multiple games between multiple players within a rating period (tournament) are treated in order to apply the rating algorithm. The ratings, rating deviations, and rating volatilities for all players have to be set before the tournament starts. If this player is an established player, this data is already known but if the player is new, his/her performance rating has to be defined first. Performance rating is a player's first rating and differs from organisation to organisation. Some chess organisations set the performance rating after the player finishes his/her first n games to make the player's first rating more reliable. Glickman recommended setting rating R to 1500, rating deviation RD to 350, and rating volatility σ to 0.06. Since rating deviation affects the reliablity of rating, it should not be too big. Glickman suggested applying the following formula between rating periods:

$$RD' = \min\left(\sqrt{RD^2 + c^2 t}, 350\right), \tag{4}$$

where t is the number of tournaments that have passed since the player last participated, and c is the constant that regulates the increase of unreliability over time. If a player participated at the latest tournament then t equals 1. The constant c is set by each organisation individually and determines how many rating periods would need to pass before a rating for a typical player becomes as unreliable as that of a non-rated player. For example, suppose that a typical player has an RD of 30 and the rating periods happen every 2 months. Assuming that after 3 years (36 months) of a player's inactivity his rating deviation falls back to 350, constant c can be set by solving equation $350 = \sqrt{30^2 + c^2 * 18}$, since the time that must have passed would be $t = 18$ rating periods. In this case $c = 82.2$ would be used. The Glicko-2 system introduces another new constant τ, which plays a similar role to t and is used later in Eq. (10). Glickman recommended choosing τ from interval $(0.3, 1.2)$.

The Glicko-2 system uses a different scale than the Glicko system. Therefore, the rating R and rating deviation value RD have to be converted from the Glicko to the Glicko-2 system. The value of rating volatility σ is already in Glicko-2 scale and does not need to be converted. The converted value of rating R is denoted by μ and the converted value of rating deviation RD is denoted by ϕ. The conversions are performed by the following formulas:

$$\mu = \frac{R - 1500}{173.7178} \text{ and } \phi = \frac{RD}{173.7178} \tag{5}$$

When all the players have their own Glicko-2 ratings, rating deviations, and rating volatilities, the tournament can start. Each participant of the tournament plays k games, one game with each of the k opponents with ratings $R_1, R_2, \ldots, R_k$ and rating deviations $RD_1, RD_2, \ldots, RD_k$. The scores he/she gains are noted by $S_1, S_2, \ldots, S_k$, where S_i equals 1 for win, 0 for lose, and 0.5 for draw.

The estimated variance v of the player's rating based only on game outcomes is calculated using the formula in Eq. (6).

$$v = \left(\sum_{i=1}^{k} g(\phi_i)^2 E(\mu, \mu_i, \phi_i)(1 - E(\mu, \mu_i, \phi_i))\right)^{-1} \tag{6}$$

The gravity factor $g(\phi)$ (Eq. (7)) plays the same role as the factor K in the Elo system [27] but depends on the rating deviation of the player. The expected score $E(\mu, \mu_i, \phi_i)$ (Eq. (8)) is derived from the Bradley–Terry model for the pairwise comparison [10] and is similar to the expected score in the Elo system [27].

$$g(\phi) = \frac{1}{\sqrt{1 + 3\phi^2/\Pi^2}} \tag{7}$$

$$E(\mu, \mu_i, \phi_i) = \frac{1}{1 + 10^{-g(\phi_i)(\mu - \mu_i)}} \tag{8}$$

Calculation of the gravity factor $g(\phi)$ and the expected score $E(\mu, \mu_i, \phi_i)$ is followed by calculation of the estimated improvement in rating Δ (Eq. (9)). The pre-period rating μ is compared to the performance rating μ_i based only on the game outcomes S_i.

$$\Delta = v \sum_{i=1}^{k} g(\phi_i)(S_i - E(\mu, \mu_i, \phi_i)) \tag{9}$$

A new rating volatility σ' is found when using the Illinois algorithm [20], a variant of the regula falsi method. The function used to achieve this is

$$f(x) = \frac{e^x(\Delta^2 - \phi^2 - v - e^x)}{2(\phi^2 + v + e^x)^2} - \frac{x - a}{\tau^2} \tag{10}$$

where $a = \ln(\sigma^2)$ and with an accuracy of up to 6 decimals. The regula falsi method is used for finding zeros. Once the zero x_0 of the function above is found, $\sigma\prime$ is set to $e^{x_0/2}$, and the pre-rating period value ϕ^* (Eq. (11)) is calculated.

$$\phi^* = \sqrt{\phi^2 + \sigma'^2} \tag{11}$$

The rating deviation ϕ' (Eq. (12)) and rating μ' (Eq. (13)) are updated.

$$\phi' = \frac{1}{\sqrt{\frac{1}{(\phi^*)^2} + \frac{1}{v}}} \tag{12}$$

$$\mu' = \mu + \phi' \sum_{i=1}^{k} g(\phi_i)(S_i - E(\mu, \mu_i, \phi_i)) \tag{13}$$

Finally, these two values are converted from the Glicko-2 to the Glicko system using the formulas in Eq. (14). The new rating is denoted with R' and new rating deviation with RD'.

$$R' = 173.7178\mu' + 1500 \text{ and } RD' = 173.7178\phi' \tag{14}$$

Glickman also suggested that rating deviation RD should not fall below the previously-determined threshold (e.g., 30 rating points) as that would make a player's improvement harder, which is the viable's reasoning. The procedure of saving wins/losses/draws and updating ratings, rating deviations, and rating volatilities has quadratic complexity. The ratings, rating deviations, and rating volatilities are updated after each tournament.

It was shown in [44,45] that the Glicko-2 rating system is usable and accurate during dynamic pairwise comparison. Its benefits show especially in situations where new players are constantly entering and applying for tournaments, and when we want to summarise not only the player's power but also his/her game behaviour. It is also appropriate for measuring the absolute powers of players and in situations where those powers change over time (with new, better players coming into contention). It is because of these assets that we suggest applying the Glicko-2 chess rating system as a method for analysing the performances of evolutionary algorithms and comparing the results obtained during such experiments. The formulas from the Glicko-2 rating system are used in the proposed method. From now on, evolutionary algorithms represent the chess players.

4. CRS4EAs: chess rating system for evolutionary algorithms

A new approach of evolutionary algorithm performance evaluation inspired by the Glicko-2 chess rating system, is proposed and explained in this section. Computational experiments are conducted in the following manner [5]:

1. experimental context and design,
2. experiment execution,
3. analysis and presentation of results,
4. interpretation of results.

Section 2 highlights some drawbacks of NHST, which was one of the main motivations in our search for a new alternative. Most mistakes happen because of poor experimental design (step 1) and false interpretations of collected results (steps 3 and 5). Our approach takes place in the same order, by focusing on these vulnerable steps. Whilst the goal of computational experiments within the field of meta-heuristics, particularly in evolutionary computations, is to show that one algorithm statistically outperforms others significantly, the goal of our approach is to evaluate the algorithms' results and then rank the algorithms according to their powers (ratings). Significant differences are detected with observed differences in ratings, which is explained in detail in the continuation.

4.1. Experimental context and design

Every experiment should have a clear purpose (e.g., to compare the performances of various algorithms within the same test-suite), as defined prior to the actual execution of an experiment. In general, a newly proposed algorithm should be compared against the currently better available algorithms over a given set of problems. Unfortunately, this is often not the case and newly-proposed algorithms are often compared to those algorithms or their variants for which the newly-proposed algorithm outperforms them. Using the newly-proposed CRS4EAs method each algorithm has its own ratings for selected benchmark problems and it is easy to check what the ratings are of those particular algorithms used in comparison. Justification for using algorithms with low-ratings for comparison would be harder to explain.

Furthermore, during experimental design performance measurements and factors (controllable variables during an experiment that influence the result) involved need to be defined. Typical performance measurements include solution quality, computational effort, and robustness. Many factors can influence the results such as problem factors (e.g., size of the problem), algorithm factors (e.g., control parameters), and environment factors (e.g., computer configuration, programmers' skills). The experimental design of the CRS4EAs is in many respects similar to the experimental design in classical empirical experimentalism [6] but with additional emphasise on problem-algorithm-environment factors to avoid inappropriate and unfair comparison. In particular, the experiments in our approach:

- are run on the same set of N optimisation problems $\{F_1, F_2, \ldots, F_N\}$, by avoiding the omissions of some problems,
- consist of algorithms written in the same programming language, avoiding comparison between computational experiments written in different programming languages (e.g., Java, Matlab),
- have the same termination conditions (maximum number of evaluations), avoiding different stopping conditions (e.g., the number of generations vs. the number of fitness evaluations),
- are initialised with the same random seed, avoiding the seed factor, and
- are run under the same hardware configuration, avoiding the environment factor.

Whilst performance measurements (e.g., solution quality) are built into the tool Evolutionary Algorithm Rating System (EARS) [28] that supports CRS4EAs, misuses or miscalculations of these measurements are also less likely. These conditions allow us to avoid some possible threats to the validities of the conclusions.

4.2. Experiment execution

An experiment regarding CRS4EAs is conducted in the form of tournament. This tournament consists of a set of k algorithms $\{a_1, a_2, \ldots, a_k\}$, N optimisation problems, and is conducted over n independent runs. Each algorithm returns the best solution for each optimisation problem over each run ($k * N * n$ results), which are then compared between each other. A set $\{a_i, a_j\}_{l,m}$, where $i,j \in \{1, \ldots, k\}, i \neq j, l \in \{1, \ldots, N\}$, and $m \in \{1, \ldots, n\}$ is understood as one comparison or one game of algorithms a_i and a_j on optimisation problem F_l over run m. The solutions y_i and y_j of algorithms a_i and a_j for the same problem F_l over run m are compared, and those algorithms the solutions of which got closer to the optimum of F_l win. If the difference between solutions is less than the predefined draw limit ϵ, the result of the game is a draw. Hence, the tournament consists of $(k * (k-1)/2) * N * n$ games. The results from the experiment showed that the order of games does not significantly affect the outcome (see Table 4). After the tournament is concluded the results are gathered in the forms of wins, losses, and draws. The ratings, rating deviations' values, and rating volatilities are updated at the end of each tournament as the Glicko-2 rating system suggests (see Section 3). All the gathered data is saved for the next tournament and shown on a leaderboard. This procedure is illustrated in Fig. 1.

The Evolutionary Algorithm Rating System (EARS) has been developed for the purpose of demonstrating and promoting the CRS4EAs method. EARS is an open-source framework [29] developed with Java 1.6, with the goal of providing fair and

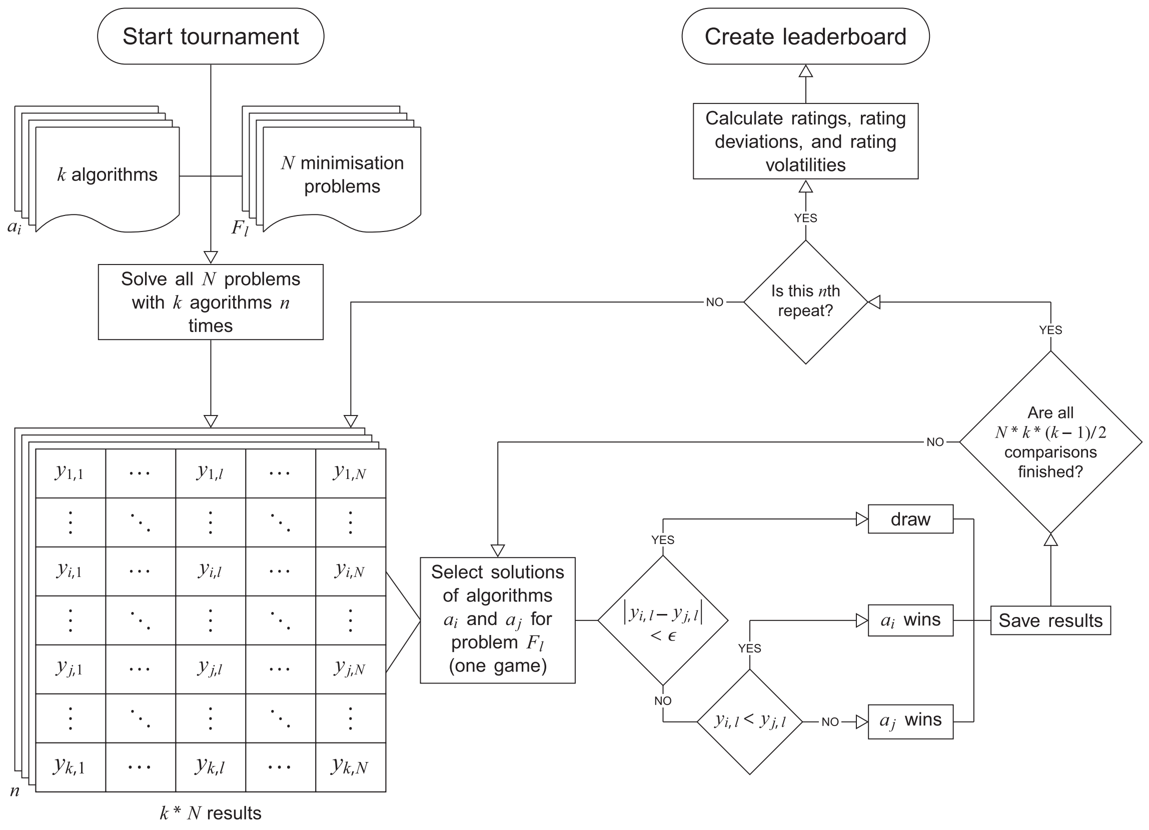

Start tournament
k algorithms
a_i
N minimisation problems
F_l
Solve all N problems with k algorithms n times
$y_{1,1}$ ⋯ $y_{1,l}$ ⋯ $y_{1,N}$
$y_{i,1}$ ⋯ $y_{i,l}$ ⋯ $y_{i,N}$
$y_{j,1}$ ⋯ $y_{j,l}$ ⋯ $y_{j,N}$
$y_{k,1}$ ⋯ $y_{k,l}$ ⋯ $y_{k,N}$
n
$k * N$ results
Select solutions of algorithms a_i and a_j for problem F_l (one game)
$|y_{i,l} - y_{j,l}| < \epsilon$
YES
NO
$y_{i,l} < y_{j,l}$
YES
NO
draw
a_i wins
a_j wins
Save results
Are all $N * k * (k-1)/2$ comparisons finished?
NO
YES
Is this nth repeat?
NO
YES
Calculate ratings, rating deviations, and rating volatilities
Create leaderboard

Fig. 1. Flowchart of experiment execution for the suggested CRS4EAs approach.

easy comparisons between evolutionary algorithms. As described in Fig. 1 the inputs to EARS are a benchmark test suit and evolutionary algorithms that also include other parameters needed for precisely describing problems and algorithms. A benchmark is defined by a set of optimisation problems/functions implemented in Java and by the number of fitness evaluations that serve as stopping conditions for all participating algorithms. The concrete algorithm must be implemented as inherited class from abstract class *Algorithm* that overrides method *run(BenchmarkProblem)*. The tournament parameters (number of independent runs n and draw limit ϵ) have to be determined when starting a tournament, besides selecting the algorithms and the benchmark test suite. A leaderboard with ratings and different statistical reports is provided at the end of the tournament. Algorithms can only apply to the benchmark already created by EARS, and have to compete with all other contestants of this benchmark on all optimisation problems within the benchmark.

4.3. Analysis and presentation of results

The results in CRS4EAs are presented on a predefined form of leaderboard (e.g., Table 3), where algorithms' ratings, rating deviations, rating volatilities, and rating intervals are displayed. An algorithm's rating R is the power of the algorithm after all executed tournaments. Note that it is unnecessary for every algorithm to participate in every tournament but once an algorithm applies to a specific tournament it has to compete on all benchmark functions included in this particular tournament. Rating deviation RD, rating volatility σ, and rating interval RI are measurements as suggested by the Glicko-2 rating system with known explanations. We decided on a 99.7% confidence interval $[R - 3RD, R + 3RD]$ to make sure that any possible error was minimal. The leaderboard is designed as a classical leaderboard with the best algorithm on the top with a rank of 1. A graphical presentation of results is created in two ways. The first graph represents a multiset of rating intervals on the natural line with a scale of 100 for all algorithms participating in the tournament. Minimum, maximum, and actual ratings are specially marked at the intervals (e.g., Fig. 2). The second presentation is made with the help of graph theory. One node presents one algorithm and two nodes are connected if their rating intervals do not overlap, meaning that these two algorithms are significantly different (e.g., Fig. 4(a)). Both graphical presentations allow for straightforward interpretation and analysis of the results without fear of misunderstanding or misinterpretation.

4.4. Interpretation of results

The objective interpretation of the results is made by the observation of ratings and rating intervals. The rating itself shows the absolute power of the algorithm over other algorithms, however, the rating interval should also be considered. It is known that when confidence intervals do not overlap algorithms provide significantly different results, whilst the converse is not necessarily true [62]. However, confidence intervals can be used as a quick and relatively rough method for exploratory data analysis [79]. Our approach uses confidence intervals for interpretations of the results with respect to the presented rule. Graphical presentation of rating/confidence intervals improves understanding and correct interpretation of the results.

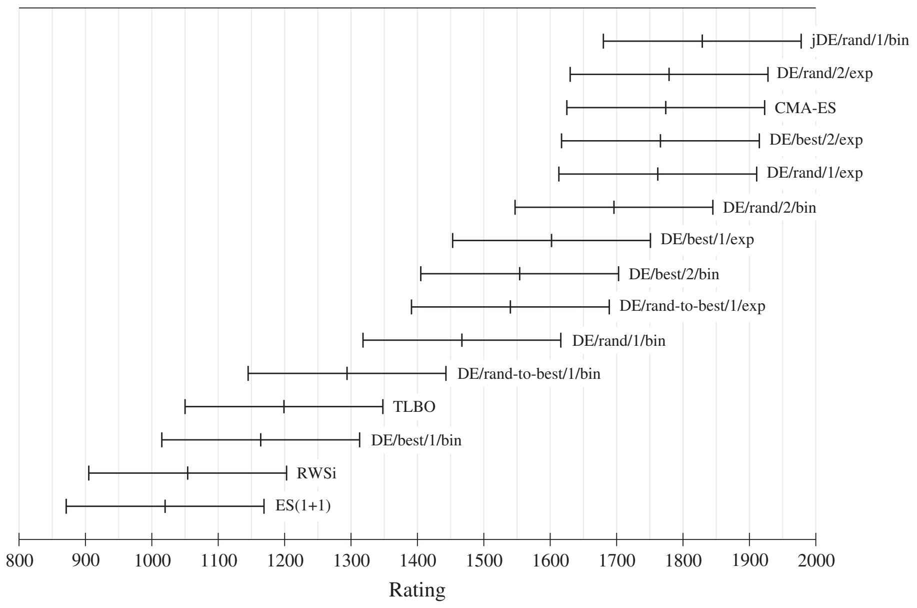


Fig. 2. 99.7% confidence intervals for the algorithms' ratings from Table 3. The points along the black horizontal axes represent ratings, the light grey auxilliary lines are for a differences of 50 rating points.

5. Experiment

The CRS4EAs in action is shown in this section. It is compared to NHST in order to gain some confidence about its suitability for comparison with evolutionary algorithms.

5.1. Experimental setup

An experiment was conducted to compare the performances of our Java implementations regarding 16 evolutionary algorithms from Table 1. The first is the basic random search suggested by Rastrigin [75] and named Random Walk Simple algorithm (RWSi). The second is the optimisation method motivated by the intelligent behaviour of honey bees, Artificial Bee Colony (ABC) [60]. The third algorithm is the Teaching Learning Based Optimization (TLBO) [15,74], which works on the effect the influence of a teacher has on learners, and consists of two parts, the 'teachers phase', and 'learners phase'. There are two variants of evolution strategies (ES). The first is ES (1 + 1) [76] and the second is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [48,88]. Another family of algorithms used in our experiment is the Differential Evolution (DE) [17,70,85,90]. There are several mutation and crossover variants that differentiate one DE scheme from another. The general convention for the DE scheme is DE/x/y/z, where x represents a string denoting the base vector to be mutated, y is the number of difference vectors considered for the mutation of x, and z stands for the type of crossover being used. In our experiment we used 10 DE schemes, combining three variants of the mutation (best, rand, rand-to-best) and two variants of the crossover (exp, bin). Another version of DE that is used is the Self-adaptive Differential Evolution (jDE) [11]. jDE is based on the adaptations of control parameters F (which affects the mutation) and Cr (which affects the crossover). Each individual has its own control parameters F_i and Cr_i that changes over generations.

The F_1–F_{20} benchmark functions from a Special Session and Competition on Large-Scale Global Optimization [87] were used in our experiment (Table 2). The benchmark functions are classified into five different groups, depending on their properties. A random orthogonal matrix with dimension $m \times m$ (used for rotation) is generated with library EMJL [24], all permutations are random, the shifts are predefined and are unique for each objective function. All problems have the global optimum within the given bounds, the optimum function values are 0 for all 20 problems.

The experiment was divided into two parts. In the first part, a tournament was conducted with 15 evolutionary algorithms and the relationships between these algorithms were studied. In the second part ABC joined and the focus was on the relationships between ABC and the other 15 algorithms. Data was gathered for all $N = 20$ problems with dimension $D = 30$ and group size parameter $m = 2$. The maximum number of function evaluation Max_FEs was 10^5 and the search for optimum terminated when Max_FEs were reached. The initial CRS4EAs settings for the new algorithm were rating $R = 1500$, rating deviation $RD = 350$ and rating volatility $\sigma = 0.06$. The thresholds for maximum and minimum rating deviations were $\max_{RD} = 350$ and $\min_{RD} = 50$. Increase in $\min_{RD}$ makes our approach more conservative, and any decrease more liberal. Minimum rating deviation of 50 rating points was a good adjustment for the test to be neither too conservative nor too liberal. The 99.7% confidence interval RI was used for analysis with the meaning that the difference between two ratings is significant if the difference is larger than $3 * RD$.

The first part of experiment was conducted with respect to two control parameters, the number of independent runs n and the draw limit ϵ. The results were compared for $n = \{25, 50, 100\}$ and $\epsilon = \{1.0e{-2}, 1.0e{-6}, 1.0e{-10}\}$. There was no significant difference between the ratings in all 9 experiments. Even when some algorithms performed better in some experiments, the changes were still within the confidence interval. As that was the case, we assume that the approach is insensitive to these parameters, and we present only the results for $n = 100$, and $\epsilon = 1.0e{-6}$. In the second part of the

Table 1
Evolutionary algorithms used in the experiment. Used abbreviations: NP – population size, λ – number of offspring, μ – number of parents, D – dimensionality of the problem, Cr – crossover factor, F – mutation factor, Cr_{init} and F_{init} – initial crossover and mutation factors, that change during evolution.

Algorithm	Name	Control parameters settings
RWSi	Random Walk Simple	–
ABC	Artificial Bee Colony	$NP = 20, limit = 100$
TLBO	Teaching Learning Based Optimization	$NP = 20, elitism = false, removeDuplicates = true$
ES (1 + 1)	Evolution Strategy	$\lambda = 1, \mu = 1, k = 40, c = 0.8$ (1/5th-rule)
CMA-ES	Covariance Matrix Adaptation Evolution Strategy	$\lambda = 4 + 3\log D, \mu = \lambda/2$
DE/best/1/bin	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/best/1/exp	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/best/2/bin	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/best/2/exp	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/rand/1/bin	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/rand/1/exp	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/rand/2/bin	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/rand/2/exp	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/rand-to-best/1/bin	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
DE/rand-to-best/1/exp	Differential Evolution	$NP = 20, Cr = 0.9, F = 0.5$
jDE/rand/1/bin	Self-adaptive Differential Evolution	$NP = 20, Cr_{init} = 0.9, F_{init} = 0.5$

Table 2
Benchmark functions for the CEC'2010 special session and competition on Large-Scale Global Optimization.

Function	Name	Range
Separable functions		
F_1	Shifted Elliptic Function	$[-100, 100]^D$
F_2	Shifted Rastrigin's Function	$[-5, 5]^D$
F_3	Shifted Ackley's Function	$[-32, 32]^D$
Single-group m-nonseparable functions		
F_4	Single-group Shifted and m-rotated Elliptic Function	$[-100, 100]^D$
F_5	Single-group Shifted and m-rotated Rastrigin's Function	$[-5, 5]^D$
F_6	Single-group Shifted and m-rotated Ackley's Function	$[-32, 32]^D$
F_7	Single-group Shifted m-dimensional Schwefel's Problem 1.2	$[-100, 100]^D$
F_8	Single-group Shifted m-dimensional Rosenbrock's Function	$[-100, 100]^D$
$D/2m$-group m-nonseparable functions		
F_9	$D/2m$-group Shifted and m-rotated Elliptic Function	$[-100, 100]^D$
F_{10}	$D/2m$-group Shifted and m-rotated Rastrigin's Function	$[-5, 5]^D$
F_{11}	$D/2m$-group Shifted and m-rotated Ackley's Function	$[-32, 32]^D$
F_{12}	$D/2m$-group Shifted m-dimensional Schwefel's Problem 1.2	$[-100, 100]^D$
F_{13}	$D/2m$-group Shifted m-dimensional Rosenbrock's Function	$[-100, 100]^D$
D/m-group m-nonseparable functions		
F_{14}	D/m-group Shifted and m-rotated Elliptic Function	$[-100, 100]^D$
F_{15}	D/m-group Shifted and m-rotated Rastrigin's Function	$[-5, 5]^D$
F_{16}	D/m-group Shifted and m-rotated Ackley's Function	$[-32, 32]^D$
F_{17}	D/m-group Shifted m-dimensional Schwefel's Problem 1.2	$[-100, 100]^D$
F_{18}	D/m-group Shifted m-dimensional Rosenbrock's Function	$[-100, 100]^D$
Nonseparable functions		
F_{19}	Shifted Schwefel's Problem 1.2	$[-100, 100]^D$
F_{20}	Shifted Rosenbrock's Function	$[-100, 100]^D$

experiment we continued with ϵ = 1.0e−6, and the control parameter was the number of independent runs n. The results were compared for $n = \{10, 25, 50, 100\}$ and again there was no significant difference between ratings.

The goal of this experiment was to show that the comparison of evolutionary algorithms with CRS4EAs provides comparable results for classical statistical inference – NHST. Hence, the comparisons between 16 evolutionary algorithms regarding 20 benchmark optimisation functions were conducted by the newly-proposed CRS4EAs method and by the classic statistic NHST method. For $k \times k$ comparison, we used our implementation of the MULTIPLE test package [40]. The input data for the statistics came from our program and not from an independent Excel file. The output was changed and added to our text output file.

5.2. Analysis and graphical presentation of results

5.2.1. Part I

During the first part of experiment, 15 evolutionary algorithms competed in a tournament and $(15 * 14/2) * 20 * 100 = 210,000$ games were played. The results of the tournament are presented in the form of the leaderboard in Table 3. All algorithms played through out the whole tournament, which was the reason for a decrease in the rating deviations. The RDs reached their minimum value (50) for all algorithms. There was a drastic change in the rating volatilities, as was expected, as this was the first tournament for all algorithms. Such a drastic change could also happen in a situation in which a lot of excellent algorithms were to join the system and one previously good algorithm was to lose all games against new algorithms. Ratings R and rating intervals RI indicate that the jDE/rand/1/bin is the strongest and most successful algorithm, with DE/rand/2/exp, CMA-ES, DE/best/2/exp, and DE/rand/1/exp following it. RWSi and ES (1 + 1) performed the worst and are close but at the bottom of leaderboard. Non-overlapping intervals indicate significant differences between the algorithms (Fig. 2). The more significant are the differences between algorithms jDE/rand/1/bin and ES (1 + 1), jDE/rand/1/bin and RWSi, DE/rand/2/exp and ES (1 + 1), CMA-ES and ES (1 + 1), DE/rand/2/exp and RWSi, and CMA-ES and RWSi. The algorithms that are the closest in performance are DE/best/2/exp and DE/rand/1/exp, and CMA-ES and DE/rand/2/exp.

The effect of the order of the games was additionally tested (Table 4). First, the results of each algorithm on each test function were ordered randomly. This was repeated three times to see whether the different combinations of the games would significantly affect the final results. The maximum difference occurred for the algorithm TLBO, between the first run and the second run, and it equals 3 rating points, which is not significant. Later, the results of each algorithm on each test function were ordered from the worst fitness value to the best fitness value. The biggest difference was detected for the algorithm

Table 3
CRS4EAs leaderboard of 15 evolutionary algorithms after the first tournament.

i	Algorithm	R	RD	σ	RI (99.7%)
1	jDE/rand/1/bin	1829	50	18	[1679, 1979]
2	DE/rand/2/exp	1779	50	18	[1629, 1929]
3	CMA-ES	1774	50	18	[1624, 1924]
4	DE/best/2/exp	1766	50	18	[1616, 1916]
5	DE/rand/1/exp	1762	50	18	[1612, 1912]
6	DE/rand/2/bin	1696	50	18	[1546, 1846]
7	DE/best/1/exp	1602	50	18	[1452, 1752]
8	DE/best/2/bin	1554	50	18	[1404, 1704]
9	DE/rand-to-best/1/exp	1540	50	18	[1390, 1690]
10	DE/rand/1/bin	1467	50	18	[1317, 1617]
11	DE/rand-to-best/1/bin	1294	50	18	[1144, 1444]
12	TLBO	1199	50	18	[1049, 1349]
13	DE/best/1/bin	1164	50	18	[1014, 1314]
14	RWSi	1054	50	18	[904, 1204]
15	ES (1 + 1)	1020	50	18	[870, 1170]

Table 4
Comparison of ratings obtained with different orders of games (for one algorithm within one optimization problem).

i	Algorithm	Obtained rating	Rating with results ordered randomly (1st)	Rating with results ordered randomly (2nd)	Rating with results ordered randomly (3rd)	Rating with ordered results
1	jDE/rand/1/bin	1829	1828	1829	1829	1837
2	DE/rand/2/exp	1779	1780	1780	1780	1768
3	CMA-ES	1774	1775	1775	1774	1771
4	DE/best/2/exp	1766	1766	1765	1765	1766
5	DE/rand/1/exp	1762	1763	1762	1762	1770
6	DE/rand/2/bin	1696	1696	1696	1696	1708
7	DE/best/1/exp	1602	1601	1602	1601	1608
8	DE/best/2/bin	1554	1553	1555	1555	1550
9	DE/rand-to-best/1/exp	1540	1539	1540	1540	1546
10	DE/rand/1/bin	1467	1467	1468	1468	1460
11	DE/rand-to-best/1/bin	1294	1294	1295	1294	1300
12	TLBO	1199	1200	1197	1198	1188
13	DE/best/1/bin	1164	1164	1164	1164	1161
14	RWSi	1054	1053	1053	1053	1052
15	ES (1 + 1)	1020	1020	1020	1020	1014

DE/rand/2/bin, and it equals 12 rating points, which is again not significant. Note that, ordering the results by the fitness value decreases the influences of the outliers even more, as the worst solutions are compared to each other. The order of games does not significantly affect the final ratings.

Classical statistical inference was conducted for the comparison between both approaches. The average rankings needed for the Friedman and Iman and Davenport tests, are shown in Table 5. The average rankings can also be used as indicators as to how successful the algorithm is, that is the lower the rank the more successful the algorithm. The first place belongs to DE/rand/2/exp, followed by jDE/rand/1/bin, CMA-ES, DE/rand/1/exp, DE/best/2/exp, and DE/rand/2/bin.

The Friedman statistic distributed according to the chi-square, with 14 degrees of freedom equals $\chi_F^2 = 211.173750$. The p value computed by the Friedman test equals 8.595746e$-$11. The Iman and Davenport statistic distributed according to F-distribution with 14 and 266 degrees of freedom equals $F_F^2 = 58.296090$. The p value computed by the Iman and Daveport test equals 1.492098e$-$72. Both times the p value is less than the significance level $\alpha = 0.05$ and the hypothesis that there is no difference in rankings for these 15 algorithms is rejected. Rejection of the null hypothesis was followed by post hoc analysis. The appropriate post hoc test is the Nemenyi test because we compare multiple algorithms on multiple data sets.

The adjusted p-values for the Nemenyi test are shown in Table 6. There are 105 hypotheses for $(15 * 14)/2$ comparisons. Hypotheses are ordered by p values, where the first algorithm has better (lower) average ranking than the second. If the corresponding p value is smaller than $\alpha = 0.05$, then the first algorithm is significantly better than the second. Critical difference CD is calculated, where $q_{0.05} = 4.8/\sqrt{2}$, which gives $CD = 3.39\sqrt{(15 \cdot 16)/(6 \cdot 20)} = 4.79$. The comparisons of all algorithms against each other using the critical difference from the Nemenyi test is shown in Fig. 3. Those groups of algorithms that are not significantly different are connected. NHST found the more significant differences between algorithms to be DE/rand/2/exp and ES (1 + 1), jDE/rand/1/bin and ES (1 + 1), DE/rand/2/exp and RWSi, jDE/rand/1/bin and RWSi, CMA-ES and ES (1 + 1), CMA-ES and RWSi, DE/rand/1/exp and ES (1 + 1) and DE/rand/1/exp and RWSi. Those algorithms that are the closest in performance are CMA-ES and DE/rand/1/exp with the same average rank, and DE/best/2/exp and DE/rand/2/bin.

Table 5
Ranking of 15 evolutionary algorithms based on average rankings.

i	Algorithm	Average rankings
1	DE/rand/2/exp	3.425
2	jDE/rand/1/bin	3.6
3	CMA-ES	4.325
4	DE/rand/1/exp	4.325
5	DE/best/2/exp	4.675
6	DE/rand/2/bin	4.75
7	DE/rand-to-best/1/exp	7.05
8	DE/best/1/exp	7.675
9	DE/best/2/bin	7.975
10	DE/rand/1/bin	8.5
11	DE/rand-to-best/1/bin	10.8
12	TLBO	12.05
13	DE/best/1/bin	12.55
14	RWSi	13.7
15	ES (1 + 1)	14.6

Due to the transformations from unadjusted to adjusted p values, there are some differences between the results in Table 6 and Fig. 3. The adjusted p value for the 42nd hypothesis jDE/rand/1/bin vs. DE/rand/1/bin is bigger than 0.05 and the hypothesis should therefore be retained (Table 6) with the meaning that jDE/rand/1/bin does not significantly outperform DE/rand/1/bin. However, that is not the case when we do not adjust the p values such as comparison with the critical difference (Fig. 3). Since the difference between the ranks of jDE/rand/1/bin and DE/rand/1/bin is 4.9, which is greater than $CD = 4.75$, this hypothesis should be rejected with the meaning that the jDE/rand/1/bin significantly outperforms the DE/rand/1/bin. The same holds for the 43rd hypothesis: DE/best/1/exp vs. DE/best/1/bin. Overall, the statistical tests are really sensitive and the results sometimes differed due to the number of decimal places, the precisions of the values for the distribution function, or the conservativity of the test.

As different statistical methods have different powers (possibility of Type II error is different), the results of the significance tests might be slightly different. However, how good and reliable are the rankings obtained by CRS4EAs? By comparing Table 3 with Table 5 it can be noticed that the rankings of the algorithms are different for CRS4EAs and NHST. However, Table 7 shows that the majority of algorithms took the same place in both approaches. The ones that does not, are neighbours in both approaches: jDE/rand/1/bin and DE/rand/2/exp, DE/best/2/exp and DE/rand/1/exp and DE/best/1/exp, DE/best/2/bin, and DE/rand-to-best/1/bin. More importantly, these small changes were amongst those algorithms that does not significantly outperform each other. Hence, we are convinced that the CRS4EAs rating is reliable and the method detects the significant differences comparable to NHST. An explanation for the difference in rankings among CRS4EAs and NHST is given next, followed by a discussion and explanation of the differences of significance amongst the algorithms.

Changes in rankings are due to the different interpretations of the results. CRS4EAs compares all the results pairwise, whilst NHST compares the means of the results. A look at the raw data [30] reveals that in CRS4EAs' comparison the jDE/rand/1/bin was better over 642 runs, DE/rand/2/exp over 423 runs, and they returned the same values in 935 runs (altogether $2000 = 20*100$ runs). The NHST comparison showed [30] that DE/rand/2/exp was better for 8 optimisation problems, jDE/rand/1/bin for 7 optimisation problems, and they performed the same for 5 optimisation problems (altogether 20 optimisation problems). The mean is a measure very sensitive to outliers. The results shows that only one outlier affected the mean value and ruined the average 0.000000 of jDE/rand/1/bin for problem F_4 and only one outlier for problem F_7. These two outliers were enough for the DE/rand/2/exp to outperform jDE/rand/1/bin for these two optimisation problems and affect the final average ranking. The reason for the differences also lies in the different performances regarding different optimisation problems. NHST compares the mean-values algorithms obtained for each optimisation problem. It could happen that one algorithm outperformed others in terms of the mean-value, whilst for run-by-run comparison its performance was average. CRS4EAs is less sensitive to such situations, since each result is compared independently. However, CRS4EAs ratings and NHST ranking cannot be directly compared, as NHST is always presented as a relative term. The average rankings calculated for the Friedman test depend on the algorithms to be compared, meaning that the values of average ranks and order of algorithms could change if some algorithms were to be added or removed. Average rank depends on the ranks the algorithm achieved on each problem, in regard to other algorithms. CRS4EAs ranking could also change but if one algorithm is removed, this would mean only $n * N$ games less for each algorithm, and games between two algorithms would not affect the games other $k - 2$ algorithms played. Whilst the average ranking shows relative power, CRS4EA shows more absolute power.

Both approaches also detected different relationships between algorithms regarding significant differences amongst them. These relationships are presented in the graphs (Fig. 4). Each node represents one of 15 algorithms. Two nodes are connected/related when the discussed approach detects significant differences between the algorithms. All the differences detected using NHST (Fig. 4(b)) are also detected using CRS4EAs (Fig. 4(a)). However, there are 7 pairs of algorithms for which CRS4EAs detected the difference but NHST did not (Fig. 4(c)). Some differences amongst CRS4EAs and NHST were expected due to the aforementioned differences in rankings.

Table 6
Adjusted p values for Nemenyi test. Hypothesis is **rejected** when the p value is less than $\alpha = 0.05$.

i	Hypothesis	$p_{Nemenyi}$	i	Hypothesis	$p_{Nemenyi}$
1	**DE/rand/2/exp vs. ES (1 + 1)**	**0.000000**	54	DE/best/2/exp vs. DE/rand/1/bin	0.214748
2	**jDE/rand/1/bin vs. ES (1 + 1)**	**0.000000**	55	DE/rand-to-best/1/bin vs. ES (1 + 1)	0.214748
3	**DE/rand/1/exp vs. ES (1 + 1)**	**0.000000**	56	DE/rand-to-best/1/exp vs. DE/rand-to-best/1/bin	0.214748
4	**CMA-ES vs. ES (1 + 1)**	**0.000000**	57	DE/rand/2/bin vs. DE/rand/1/bin	0.214748
5	**DE/rand/2/exp vs. RWSi**	**0.000000**	58	DE/rand/1/exp vs. DE/best/2/bin	0.214748
6	**jDE/rand/1/bin vs. RWSi**	**0.000000**	59	CMA-ES vs. DE/best/2/bin	0.214748
7	**DE/best/2/exp vs. ES (1 + 1)**	**0.000000**	60	DE/rand/2/exp vs. DE/rand-to-best/1/exp	0.214748
8	**DE/rand/2/bin vs. ES (1 + 1)**	**0.000000**	61	DE/rand/1/bin vs. TLBO	0.214748
9	**DE/rand/1/exp vs. RWSi**	**0.000000**	62	jDE/rand/1/bin vs. DE/rand-to-best/1/exp	0.214748
10	**CMA-ES vs. RWSi**	**0.000000**	63	DE/rand/1/exp vs. DE/best/1/exp	0.214748
11	**DE/rand/2/exp vs. DE/best/1/bin**	**0.000000**	64	CMA-ES vs. DE/best/1/exp	0.214748
12	**DE/best/2/exp vs. RWSi**	**0.000000**	65	DE/best/2/exp vs. DE/best/2/bin	0.214748
13	**jDE/rand/1/bin vs. DE/best/1/bin**	**0.000000**	66	DE/rand/2/bin vs. DE/best/2/bin	0.214748
14	**DE/rand/2/bin vs. RWSi**	**0.000000**	67	DE/best/1/exp vs. DE/rand-to-best/1/bin	0.214748
15	**DE/rand/2/exp vs. TLBO**	**0.000000**	68	DE/best/2/exp vs. DE/best/1/exp	0.214748
16	**jDE/rand/1/bin vs. TLBO**	**0.000000**	69	DE/best/1/exp vs. DE/rand/2/bin	0.214748
17	**DE/rand/1/exp vs. DE/best/1/bin**	**0.000001**	70	DE/rand-to-best/1/bin vs. RWSi	0.214748
18	**CMA-ES vs. DE/best/1/bin**	**0.000001**	71	DE/rand-to-best/1/bin vs. DE/best/2/bin	0.214748
19	**DE/best/2/exp vs. DE/best/1/bin**	**0.000003**	72	DE/rand/1/exp vs. DE/rand-to-best/1/exp	0.214748
20	**DE/rand/2/bin vs. DE/best/1/bin**	**0.000004**	73	CMA-ES vs. DE/rand-to-best/1/exp	0.214748
21	**DE/rand/1/exp vs. TLBO**	**0.000005**	74	TLBO vs. ES (1 + 1)	0.214748
22	**CMA-ES vs. TLBO**	**0.000005**	75	DE/best/2/exp vs. DE/rand-to-best/1/exp	0.214748
23	**DE/rand-to-best/1/exp vs. ES (1 + 1)**	**0.000010**	76	DE/rand/1/bin vs. DE/rand-to-best/1/bin	0.214748
24	**DE/best/2/exp vs. TLBO**	**0.000019**	77	DE/rand/2/bin vs. DE/rand-to-best/1/exp	0.214748
25	**DE/rand/2/exp vs. DE/rand-to-best/1/bin**	**0.000019**	78	DE/best/1/bin vs. ES (1 + 1)	0.214748
26	**DE/rand/2/bin vs. TLBO**	**0.000026**	79	DE/rand-to-best/1/bin vs. DE/best/1/bin	0.214748
27	**jDE/rand/1/bin vs. DE/rand-to-best/1/bin**	**0.000037**	80	TLBO vs. RWSi	0.214748
28	**DE/best/1/exp vs. ES (1 + 1)**	**0.000102**	81	DE/rand-to-best/1/exp vs. DE/rand/1/bin	0.214748
29	**DE/rand-to-best/1/exp vs. RWSi**	**0.000270**	82	DE/rand/2/exp vs. DE/rand/2/bin	0.214748
30	**DE/best/2/bin vs. ES (1 + 1)**	**0.000295**	83	DE/rand-to-best/1/bin vs. TLBO	0.214748
31	**DE/rand/1/exp vs. DE/rand-to-best/1/bin**	**0.000492**	84	DE/rand/2/exp vs. DE/best/2/exp	0.214748
32	**CMA-ES vs. DE/rand-to-best/1/bin**	**0.000492**	85	jDE/rand/1/bin vs. DE/rand/2/bin	0.214748
33	**DE/best/2/exp vs. DE/rand-to-best/1/bin**	**0.001558**	86	DE/best/1/bin vs. RWSi	0.214748
34	**DE/rand/1/bin vs. ES (1 + 1)**	**0.001688**	87	jDE/rand/1/bin vs. DE/best/2/exp	0.214748
35	**DE/rand/2/bin vs. DE/rand-to-best/1/bin**	**0.001980**	88	DE/rand-to-best/1/exp vs. DE/best/2/bin	0.214748
36	**DE/best/1/exp vs. RWSi**	**0.002143**	89	RWSi vs. ES (1 + 1)	0.214748
37	**DE/best/2/bin vs. RWSi**	**0.005420**	90	DE/rand/2/exp vs. DE/rand/1/exp	0.214748
38	**DE/best/1/bin vs. DE/rand-to-best/1/exp**	**0.010565**	91	DE/rand/2/exp vs. CMA-ES	0.214748
39	**DE/rand/1/bin vs. RWSi**	**0.024784**	92	DE/best/1/exp vs. DE/rand/1/bin	0.214748
40	**DE/rand/2/exp vs. DE/rand/1/bin**	**0.034913**	93	jDE/rand/1/bin vs. DE/rand/1/exp	0.214748
41	**DE/rand-to-best/1/exp vs. TLBO**	**0.042730**	94	jDE/rand/1/bin vs. CMA-ES	0.214748
42	jDE/rand/1/bin vs. DE/rand/1/bin	0.055711	95	DE/rand-to-best/1/exp vs. DE/best/1/exp	0.214748
43	DE/best/1/exp vs. DE/best/1/bin	0.059487	96	DE/best/2/bin vs. DE/rand/1/bin	0.214748
44	DE/best/2/bin vs. DE/best/1/bin	0.127719	97	TLBO vs. DE/best/1/bin	0.214748
45	DE/rand/2/exp vs. DE/best/2/bin	0.135856	98	DE/rand/1/exp vs. DE/rand/2/bin	0.214748
46	DE/best/1/exp vs. TLBO	0.207637	99	CMA-ES vs. DE/rand/2/bin	0.214748
47	jDE/rand/1/bin vs. DE/best/2/bin	0.207637	100	DE/rand/1/exp vs. DE/best/2/exp	0.214748
48	DE/rand/2/exp vs. DE/best/1/exp	0.214748	101	CMA-ES vs. DE/best/2/exp	0.214748
49	DE/rand/1/exp vs. DE/rand/1/bin	0.214748	102	DE/best/1/exp vs. DE/best/2/bin	0.214748
50	CMA-ES vs. DE/rand/1/bin	0.214748	103	DE/rand/2/exp vs. jDE/rand/1/bin	0.214748
51	DE/best/2/bin vs. TLBO	0.214748	104	DE/best/2/exp vs. DE/rand/2/bin	0.214748
52	jDE/rand/1/bin vs. DE/best/1/exp	0.214748	105	CMA-ES vs. DE/rand/1/exp	0.214748
53	DE/rand/1/bin vs. DE/best/1/bin	0.214748			

Furthermore, three pairs of algorithms have rating difference close to our threshold for significant difference (the size of interval $[-3 * 50, 3 * 50]$ is 300 rating points). The difference in ratings between CMA-ES and DE/rand/1/bin is 307 points, between DE/best/1/exp and DE/rand-to-best/1/bin 308 points, between DE/rand/1/bin and DE/best/1/bin 303 points, and the difference between DE/best/2/exp and DE/rand/1/bin is 301. Two pairs of algorithms have differences in average ranks close to the critical difference ($CD = 4.79$). The difference between the average rankings of DE/best/2/bin and DE/best/1/bin is 4.575, and the difference between the average ranks of DE/best/1/exp and TLBO is 4.375. CRS4EAs detected these differences, whilst they were too small for NHST to detect them. The detected differences (Fig. 4(c)) show that CRS4EAs is less conservative than NHST. Note that even regarding statistical testing, some methods are more conservative/liberal than others. The less conservative methods are sometimes bad choice, as any decrease in conservativity increases the Type I error. However, the conclusions about significant differences in our approach were made with 99.7% confidence, and we did not allow any rating deviation to be too small ($RD_{min} = 50$), both being very strict factors. Our approach found almost the

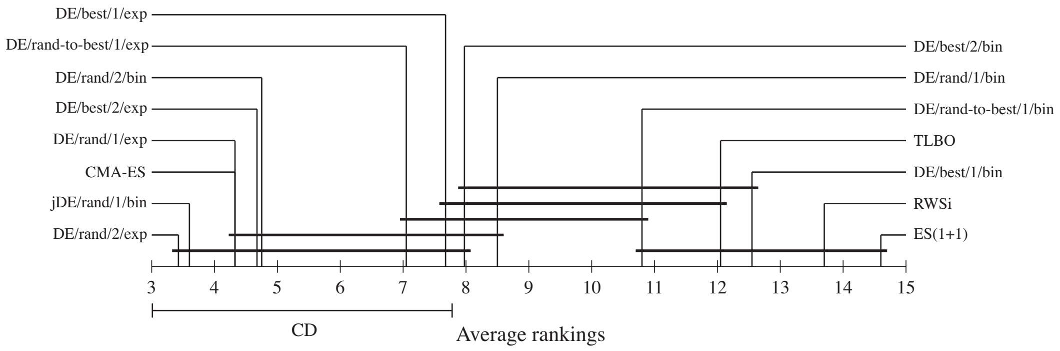


Fig. 3. Comparisons between 15 evolutionary algorithms against each other. The groups of algorithms that are not significantly different are connected. The critical difference CD is visible below the axis.

Table 7
Comparison of places obtained using the chess rating system for evolutionary algorithms (CRS4EAs) and places obtained using the null hypothesis significance testing (NHST).

Algorithm	CRS4EAs place	NHST place	Change in places
jDE/rand/1/bin	1	2	↓ 1
DE/rand/2/exp	2	1	↑ 1
CMA-ES	3	3	–
DE/best/2/exp	4	5	↓ 1
DE/rand/1/exp	5	4	↑ 1
DE/rand/2/bin	6	6	–
DE/best/1/exp	7	8	↓ 1
DE/best/2/bin	8	9	↓ 1
DE/rand-to-best/1/exp	9	7	↑ 2
DE/rand/1/bin	10	10	–
DE/rand-to-best/1/bin	11	11	–
TLBO	12	12	–
DE/best/1/bin	13	13	–
RWSi	14	14	–
ES (1 + 1)	15	15	–

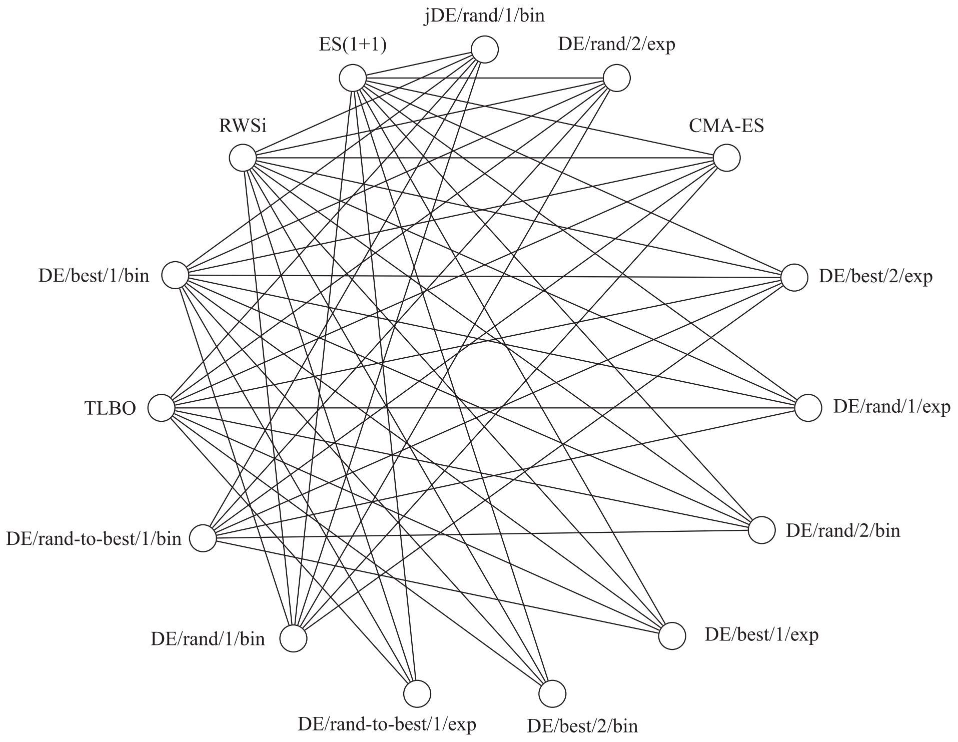
same number (but slightly more) significant differences than the post hoc Nemenyi test. However, it is very easy to make our approach more conservative or liberal. By increasing the minimum threshold for rating deviation (e.g., $RD_{min} = 60$) the CRS4EAs will become more conservative, whilst by decreasing the minimum threshold for rating deviation (e.g., $RD_{min} = 40$) the CRS4EAs will become a more liberal method. The experiment showed that its minimum rating deviation setting $RD_{min} = 52$ for collected data makes CRS4EAs more conservative than NHST. In regard to this setting, some differences that were detected with NHST could not be detected with CRS4EAs.

5.2.2. Part II

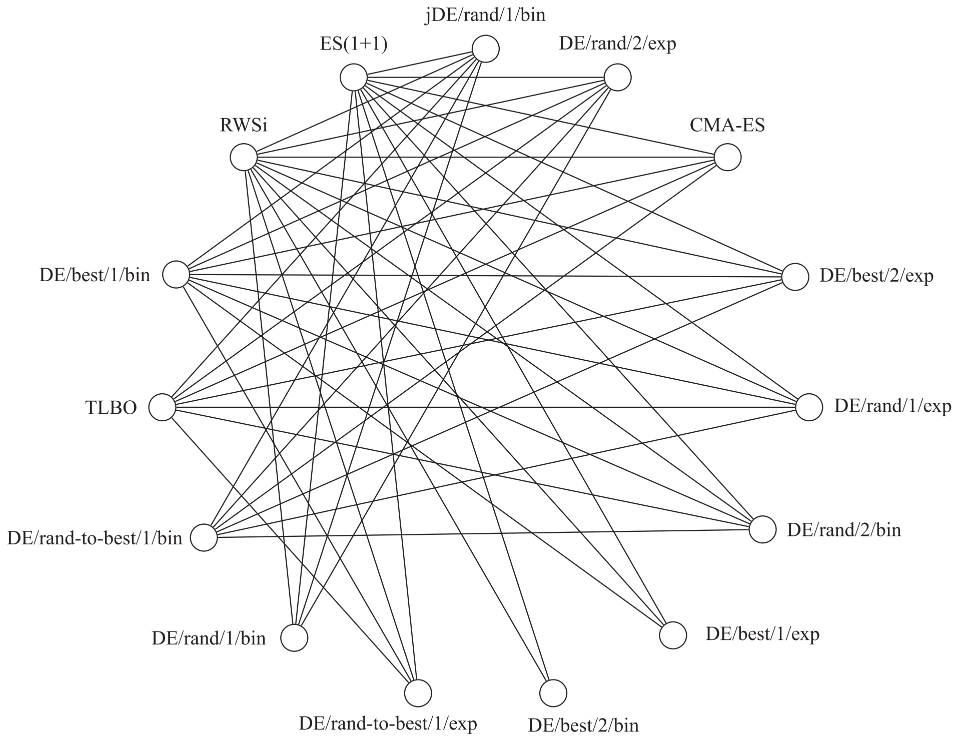
In the second part of the experiment, an ABC joined and $1 \times k$ comparison was conducted. First, ABC played $k * N * n$ games, then the rating, rating deviation, and rating volatility for ABC were updated. When ABC received its new settings, the settings for the other algorithms were also updated. The new outcomes of algorithm ABC were compared to the outcomes other algorithms already obtained when they had first participated (the outcomes obtained in the first part of the experiment). As each old algorithm had to play only with ABC, each old algorithm played only $N * n$ games. As mentioned before ABC's new rating was computed for four different numbers of runs. The new rating was 1689 for $n = 10$, 1689 for $n = 25$, 1687 for $n = 50$, and 1685 for $n = 100$. Note, the runs selected for the tournament were selected randomly from a set of 100 runs. As ABC's new ratings were very close in all four cases, the second part of the experiment with ABC rating $R = 1685$ and $n = 100$ runs was conducted to ensure the same conditions for both approaches.

The results from the second part of experiment are shown in Table 8. Again, all the rating deviations reached the minimum value of 50 rating points, and all rating volatilities changed to 17, except for ABC for which the rating volatility is larger (18). The order of ratings remained the same (see Table 3 and Table 8), except that ABC took 7th place on the leaderboard. The majority of algorithms lost rating points, only ABC and DE/rand/2/bin gained rating points. The rating points losses for the high-ranking algorithms were not as large (less than RD) as those for the low-ranking algorithms.

In order to compare CRS4EAs and NHST regarding the second part of the experiment the Friedman and the Iman and Davenport statistical tests were again used. The average rankings needed for both these statistical tests are shown in Table 9. The

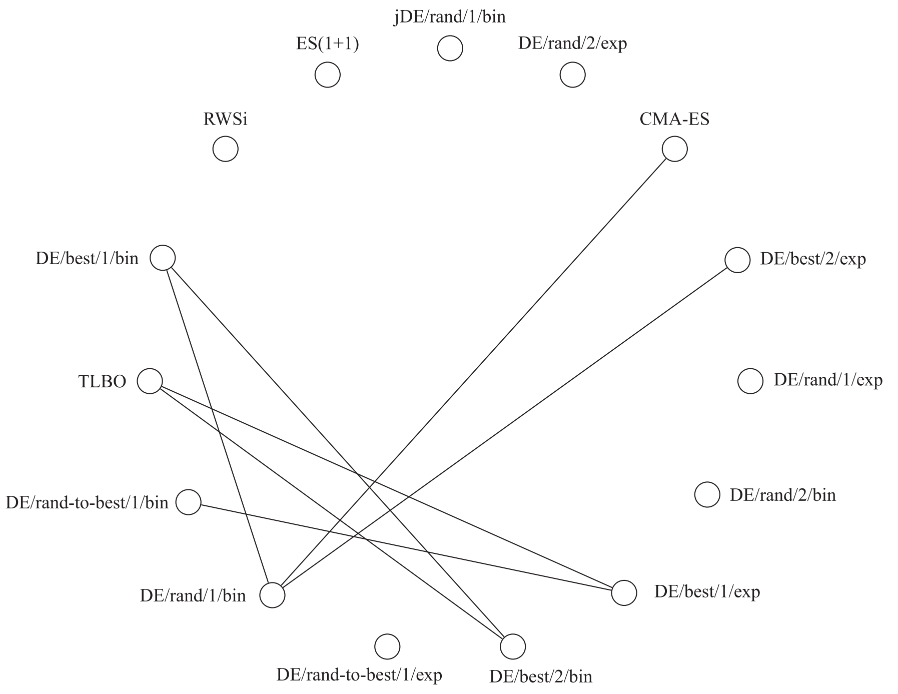


(a) Significant differences detected with CRS4EAs. Two algorithms are significantly different when the 99.7% confidence intervals do not overlap.



(b) Significant differences detected with NHST. Two algorithms are significantly different when the difference between the average ranks is less than the previously calculated critical difference CD.

Fig. 4. 15 Algorithms are presented with nodes, for which two nodes are connected/related when the suggested approach detected significant differences between the two algorithms.



(c) Significant differences that were found by CRS4EAs, but not with NHST.

Fig. 4 (*continued*)

Table 8
CRS4EAs' leaderboard of 16 evolutionary algorithms after the second tournament.

i	Algorithm	old R	new R	RD	σ	*RI* (99.7%)
1	jDE/rand/1/bin	1829	1790	50	17	[1640, 1940]
2	CMA-ES	1779	1776	50	17	[1626, 1926]
3	DE/rand/2/exp	1774	1741	50	17	[1591, 1891]
4	DE/best/2/exp	1766	1740	50	17	[1590, 1890]
5	DE/rand/1/exp	1762	1720	50	17	[1570, 1870]
6	DE/rand/2/bin	1696	1718	50	17	[1568, 1868]
7	ABC	1500	1685	50	18	[1535, 1835]
8	DE/best/1/exp	1602	1585	50	17	[1435, 1735]
9	DE/best/2/bin	1554	1520	50	17	[1370, 1670]
10	DE/rand-to-best/1/exp	1540	1394	50	17	[1244, 1544]
11	DE/rand/1/bin	1467	1296	50	17	[1146, 1446]
12	DE/rand-to-best/1/bin	1294	1014	50	17	[864, 1164]
13	TLBO	1199	892	50	17	[742, 1042]
14	DE/best/1/bin	1164	834	50	17	[684, 984]
15	RWSi	1054	713	50	17	[563, 863]
16	ES (1 + 1)	1020	678	50	17	[528, 828]

same as in CRS4EAs, the order of the average rankings remained the same (see Table 5 and Table 9), except that ABC took 4th place in the rankings. The Friedman statistic distributed according to chi-square with 15 degrees of freedom equals $\chi^2_F = 217.384191$. The p value computed by the Friedman test equals 1.157375e−10. The Iman and Davenport statistic distributed according to F-distribution with 15 and 285 degrees of freedom equals $F^2_F = 49.994059$. The p value computed by the Iman and Daveport test equals 1.232249e−70. The calculated p values are again both less than $\alpha = 0.05$, and the null hypothesis that there is no difference between the rankings of all 16 algorithms can be rejected. The rejection of the null hypothesis was followed by post hoc analysis. The appropriate post hoc tests for comparison with the control algorithm are the Holm test and the Wilcoxon signed-rank test.

The standard error SE for the post hoc analysis with Holm test using the control algorithm is $SE = \sqrt{k(k+1)/6N} = \sqrt{(16*17)/(6*20)} = 1.506$. The corresponding statistics and p values for the other $k-1 = 15$ algorithms had to be computed and ordered (Table 10). Hypotheses from $i = 1$ to 6 are rejected, as $i = 6$ is the biggest i for which the p value is still smaller than $\alpha/(16-i)$. Other hypotheses (from $i = 7$ to 15) are retained.

Table 9
Rankings of 16 evolutionary algorithms.

i	Algorithm	Average rank
1	DE/rand/2/exp	3.875
2	jDE/rand/1/bin	4.1
3	CMA-ES 2	4.8
4	ABC	4.95
5	DE/rand/1/exp	5.0
6	DE/best/2/exp	5.2
7	DE/rand/2/bin	5.275
8	DE/rand-to-best/1/exp	7.8
9	DE/best/1/exp	8.45
10	DE/best/2/bin	8.65
11	DE/rand/1/bin	9.35
12	DE/rand-to-best/1/bin	11.75
13	TLBO	13.0
14	DE/best/1/bin	13.55
15	RWSi	14.65
16	ES (1 + 1)	15.6

Table 10
Post-hoc analysis for $1 \times k$ comparison using the Holm test. Where the p value is smaller than $\alpha/(k-i)$, for $\alpha = 0.05$ the hypothesis is **rejected**. The average rankings R_i are from Table 9.

i	Algorithm	$z = (R_{ABC} - R_{algorithm})/SE$	p value	$\alpha/(k-i)$
1	**ES (1 + 1)**	**−7.071713**	**0.000000**	**0.003333**
2	**RWSi**	**−6.440903**	**0.000000**	**0.003571**
3	**DE/best/1/bin**	**−5.710491**	**0.000000**	**0.003846**
4	**TLBO**	**−5.345286**	**0.000000**	**0.004167**
5	**DE/rand-to-best/1/bin**	**−4.515272**	**0.000003**	**0.004545**
6	**DE/rand/1/bin**	**−2.921647**	**0.001741**	**0.005000**
7	DE/best/2/bin	−2.456839	0.007008	0.005556
8	DE/best/1/exp	−2.324037	0.010062	0.006250
9	DE/rand-to-best/1/exp	−1.892430	0.029217	0.007143
10	DE/rand/2/bin	−0.215803	0.414570	0.008333
11	DE/best/2/exp	−0.166003	0.434077	0.010000
12	DE/rand/1/exp	−0.033201	0.486757	0.012500
13	CMA-ES	0.099602	0.539670	0.016667
14	jDE/rand/1/bin	0.564409	0.713762	0.025000
15	DE/rand/2/exp	0.713811	0.762328	0.050000

Table 11
Post-hoc analysis for $1 \times k$ comparison using the Wilcoxon signed-rank test. Where the z value is smaller than -1.96, for $\alpha = 0.05$ the hypothesis is **rejected**.

i	Hypothesis	R^{+}	R^{-}	z Value
1	**ABC vs. ES (1 + 1)**	**0**	**210**	**−3.91993**
2	**ABC vs. DE/best/1/bin**	**0**	**210**	**−3.91993**
3	**ABC vs. RWSi**	**10**	**200**	**−3.54660**
4	**ABC vs. DE/rand-to-best/1/bin**	**13**	**197**	**−3.43461**
5	**ABC vs. TLBO**	**14**	**196**	**−3.39727**
6	**ABC vs. DE/rand/1/bin**	**45**	**165**	**−2.23996**
7	**ABC vs. DE/best/1/exp**	**52**	**158**	**−1.97863**
8	jDE/rand/1/bin vs. ABC	75	135	−1.11998
9	DE/rand/2/exp vs. ABC	77.5	132.5	−1.02665
10	DE/rand-to-best/1/exp vs. ABC	132	78	−1.00798
11	DE/best/2/bin vs. ABC	129	81	−0.89598
12	DE/rand/2/bin vs. ABC	83.5	126.5	−0.80265
13	DE/best/2/exp vs. ABC	84	126	−0.78399
14	CMA-ES vs. ABC	84.5	125.5	−0.76532
15	DE/rand/1/exp vs. ABC	121	89	−0.59732

The Wilcoxon signed rank test (Table 11) found significant differences for the same hypotheses as the Holm test, with the additional hypothesis ABC vs. DE/best/1/exp. However, this hypothesis' z value is -1.97863, which is close to the threshold for significant difference (-1.96). The difference is also due to the fact that the Wilcoxon test ranks the differences on one optimisation problem, whereas the Holm's procedure works with the average ranks from the Friedman test (Table 9). Both

Table 12
Comparison of places obtained using the chess rating system for evolutionary algorithms (CRS4EAs) and places obtained using null hypothesis significance testing (NHST).

Algorithm	CRS4EAs place	NHST place	Change in places
jDE/rand/1/bin	1	2	↓ 1
DE/rand/2/exp	2	1	↑ 1
CMA-ES	3	3	–
DE/best/2/exp	4	6	↓ 2
DE/rand/1/exp	5	5	–
DE/rand/2/bin	6	7	↓ 1
ABC	7	4	↑ 3
DE/best/1/exp	8	9	↓ 1
DE/best/2/bin	9	10	↓ 1
DE/rand-to-best/1/exp	10	8	↑ 2
DE/rand/1/bin	11	11	–
DE/rand-to-best/1/bin	12	12	–
TLBO	13	13	–
DE/best/1/bin	14	14	–
RWSi	15	15	–
ES (1 + 1)	16	16	–

the Holm procedure and the Wilcoxon signed rank test are known as being powerful but the detected significant differences were still slightly different.

By comparing Table 8 with Table 9 it can be noticed that the ranking of algorithms is different regarding CRS4EAs and NHST. Table 12 shows that half of the algorithms took the same place in both approaches. Those that does not, are: jDE/rand/1/bin, DE/rand/2/exp, DE/best/2/exp, DE/rand/2/bin, ABC, DE/best/1/exp, DE/best/2/bin, and DE/rand-to-best/1/bin. These changes are due same reasons as in the $k \times k$ comparison.

Whilst in $k \times k$ comparison the used statistical approaches detected different relationships regarding significance, in the $1 \times k$ comparison the detected relationships are almost the same. Fig. 5 show that both CRS4EAs and the Holm test detected significant differences between the algorithm ABC and the algorithms DE/rand/1/bin, DE/rand-to-best/1/bin, TLBO, DE/best/1/bin, RWSi, and ES (1 + 1); the Wilcoxon test detected an additional significant difference between the algorithms ABC and DE/best/1/exp.

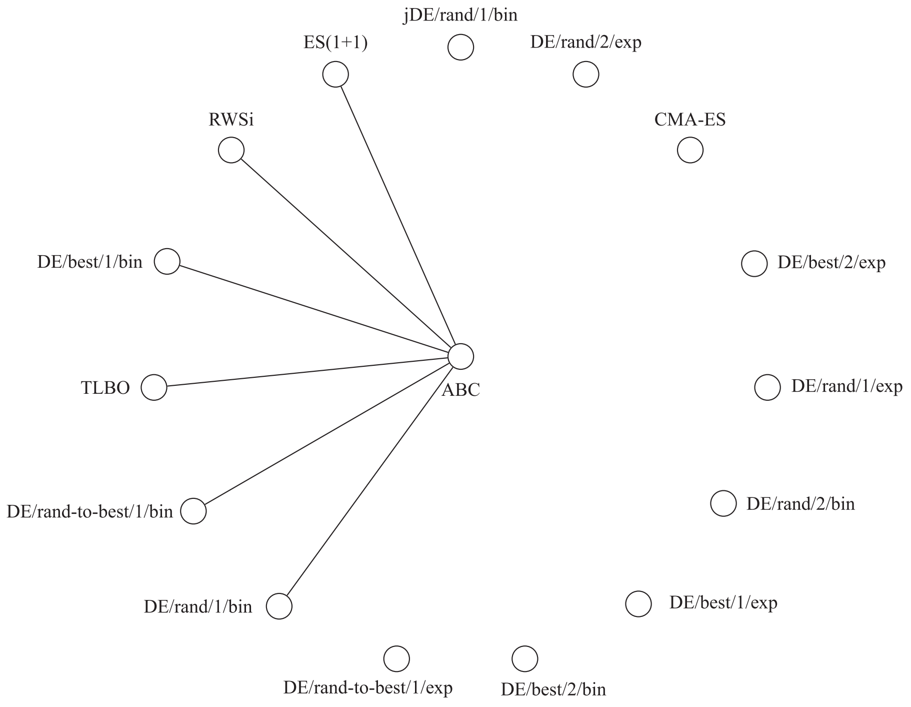
The computational complexities of both methods are quadratic. However, CRS4EAs uses formulae (Eq. (4)–(14)) that do not operate using complicated numerical computations that are sensitive to small differences and the number of digits, e.g. calculating critical difference, approximate values of continuous statistical curves, the value of the area under the curve, and other numerical issues within statistical computing [3]. Hence, CRS4EAs could be more efficiently computed than NHST. Furthermore, the number of runs required to obtain a reliable rating and results were small in the sense of detecting significant differences in CRS4EAs. Only 25 runs were required in the first part of the experiment, and only 10 runs in the second part. In both parts of the experiment, the average ranks and detected significant differences in NHST differed for the different numbers of runs. CRS4EAs is not only applicable when the number of algorithms is large. The Table 13 shows that for $k = 3$ CRS4EAs detects the same significant differences as the Nemenyi test: jDE/rand/1/bin vs. TLBO and CMA-ES vs. TLBO.

6. Discussion

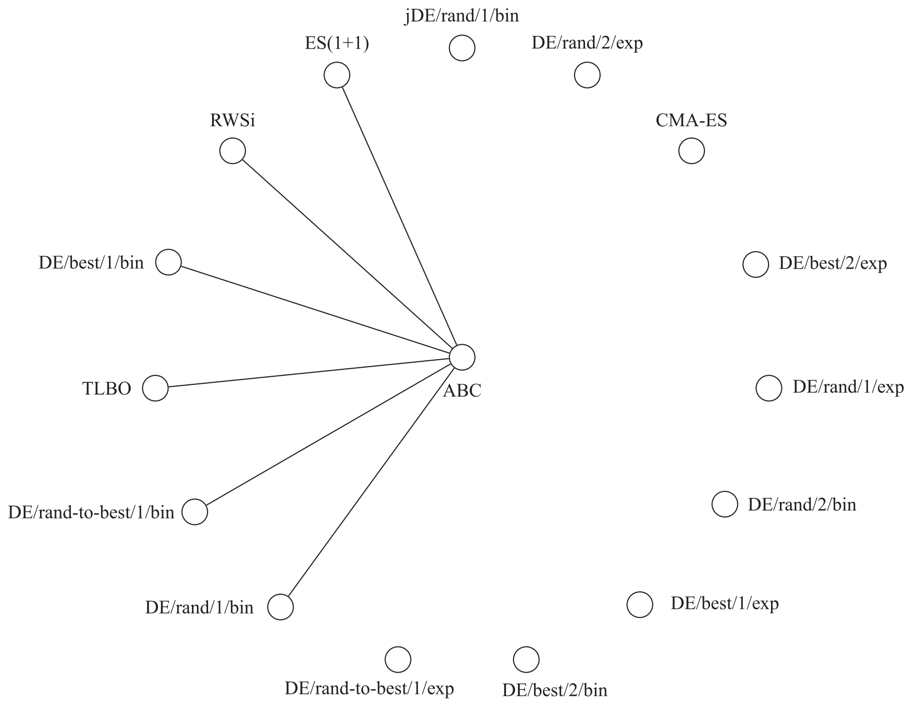
Applying CRS4EAs for the evaluation of performances and comparisons between different evolutionary algorithms has its benefits and shortcomings, as presented in this section as follows.

6.1. Experiment design and results interpretation

The aforementioned carelessness during experimental design does not ensure the same conditions for all the algorithms to be compared. Often researchers compare the results of new algorithms to those of old algorithms. It is difficult to provide the same or at least similar conditions, due to lack of details regarding experiments descriptions, different hardware, or the inexperiences of researchers in the field of statistics. Researchers should consider that older experiments were probably conducted on different computers, using different numbers of independent runs, different termination conditions, or different rounding precisions, and thus designed their experiments so they fit these conditions. The experimental design (Section 4.1) in CRS4EAs allows researchers to execute experiments under the same conditions – external factors that could affect the outcome of an experiment are excluded. This condition is what provides results with more trust. The weaknesses of statistical inference and carelessness during experimental designs directly affect the interpretations of the results. In our approach we do not have to be careful about the pitfalls of NHST [14,41,49,64,79], as it is not used. The leaderboard itself provides reliable information about the performances of the algorithms, and the confidence intervals are used instead of null hypothesis testing, for more straightforward interpretations.

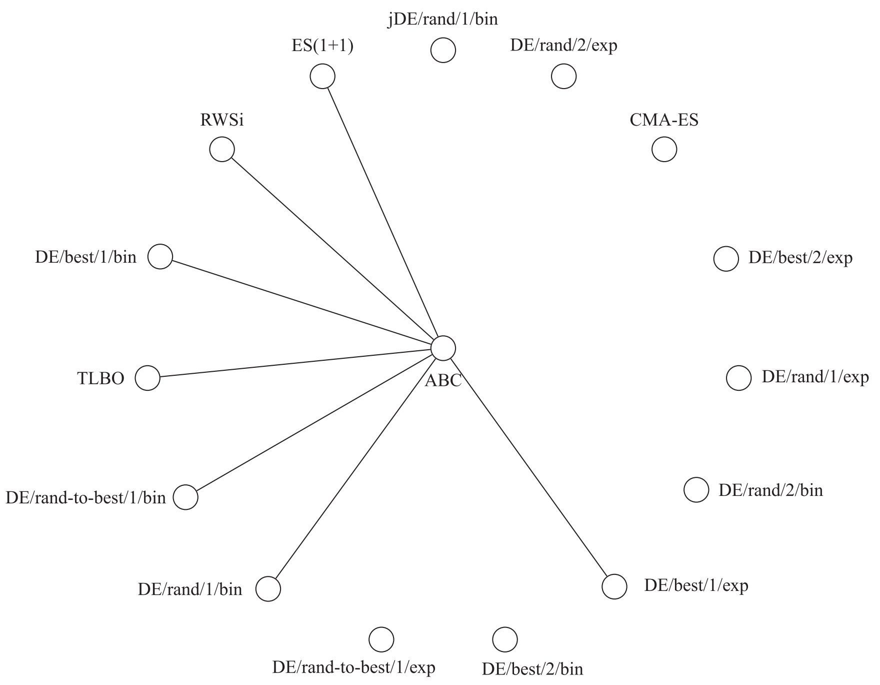


(a) Significant differences detected with CRS4EAs between ABC and the other 15 algorithms. Two algorithms are significantly different when the 99.7% confidence intervals do not overlap.



(b) Significant differences detected with Holm test between ABC and the other 15 algorithms. Two algorithms are significantly different if the Holm procedure in Table 10 detected it.

Fig. 5. 16 Algorithms are presented with nodes, for which two nodes are connected/related when the suggested approach detected significant differences between the two algorithms. The emphasis is only on the relationships between ABC and the other 15 algorithms.



(c) Significant differences detected with Wilcoxon test between ABC and the other 15 algorithms. Two algorithms are significantly different if the Wilcoxon test in Table 11 detected it.

Fig. 5 (*continued*)

Table 13
Average ranks, ratings, rating deviations and rating intervals for tournament with $k = 3$ players, $N = 20$ optimization problems, and $n = 100$ runs. Critical difference for Nemenyi test is $CD = 0.741$.

i	Algorithm	Average rank	R	RD	*RI* (99.7%)
1	JDE/rand/1/bin	1.5	1773	50	[1623, 1923]
2	CMA-ES	1.6	1722	50	[1572, 1872]
3	TLBO	2.9	1005	50	[855, 1155]

6.2. Outliers

Mean, which is a common measurement in statistics, is not a robust value, meaning that the outliers that are common within the field of evolutionary algorithms, have a big impact on the final results. As we cannot remove some data, trimmed mean is inapplicable. In our approach, the outliers only insignificantly affect the final results because one outlier means only one loss for the $(k - 1) * N * n$ games played by algorithm. One loss does not affect an algorithm's rating as much as one outlier affects the mean value and average ranking, especially if the algorithm's rating is reliable. On the other hand, with EARS [28] the outliers can be easily tracked when monitoring wins, losses, and draws among algorithms. In the cases where the performance of an algorithm needs to be stable, i.e. without big deviations in fitness, the losses with algorithms having worst ratings can be easily checked. The occurrence of outliers could be detected through observing the values of mean and standard deviation. In those methods where the results are ranked from best to worst – such as CRS4EAs and the Friedman test – this is impossible, which makes such methods less suitable for situations that demand high success rates. For example, when observing two algorithms A and B, where A's outcomes are 0.100, 0.200, 0.300, and 100.000, and B's outcomes are 0.101, 0.201, 0.301, and 0.310, it would be assumed (with ϵ = 1.0e−3) that with CRS4EAs A is better than B (since it has won 3 out of 4 games), and with NHST they are not significantly different. But application of CRS4EAs and statistical tests for two samples (*t*-test [86]) both of them give result that the differences in performances of these two algorithms are not significantly different. The p value for the *t*-test is 0.3564, while the new rating for A is 1527, with a new rating deviation of 99, and the new rating for B is 1473, with a new rating deviation of 99. The difference in ratings is 54 and even if the minimal rating deviation was smaller and the smaller rating interval was used for analysis, CRS4EAs would not detect significant difference between A and B. Because the number of runs is really small ($n = 4$) ratings and p values are unreliable, but the example shows a special case when further observation of the data would be needed. The difference for the first three outcomes was only 1% (in regard to ϵ), while A's fourth outcome was very poor. The difference in 1% might also be for ϵ = 1.0e−40 or 1.0e−100 and the phenomenon would be undetected.

6.3. Algorithm evaluation

When using the statistical inference $k * (k - 1)/2$ null hypotheses, which state that there are no differences in the performances of algorithms a_i and $a_j, \forall i, j = 1, 2, \ldots, k, i \neq j$, are formed. If the test statistic p is less than the required significance level α, the corresponding null hypothesis is rejected, implying that there is a difference between algorithms a_i and a_j. The means show which of the algorithms is significantly better, and the calculated test statistic p shows the strength of evidence against the null hypothesis. However, some of $k * (k - 1)/2$ relationships between algorithms are usually unknown, as certain hypotheses cannot be rejected. NHST therefore shows only the relative powers of the algorithms, whilst CRS4EAs returns absolute powers in the form of ratings.

6.4. $1 \times k$ Comparison

A special case regarding the comparisons of multiple algorithms on multiple data sets is comparison with a control algorithm. Such a comparison is also called $1 \times k$ comparison and is only applicable when all the k experiments are conducted under the same or stricter conditions, and all the means for all the k algorithms for each problem separately are known. As mentioned before, $1 \times k$ comparison with NHST is conducted using post hoc analysis (e.g., the Holm procedure). The $1 \times k$ comparison in CRS4EAs is conducted using $k * n * N$ games (instead of $(k * (k - 1)/2) * n * N$) in those cases when a new algorithm joins the system. In a chess world a tournament of one player against multiple opponents is called a simultaneous exhibition or simul. Only the data for a new algorithm has to be gathered ($n * N$ results instead of $k * N * n$), but the data for other algorithms stay the same (Fig. 1). The advantage of this particular case shows in the small number of runs n (25 or less) required for a new algorithm to reach its rating. This was shown in the second part of the experiment, where ratings for ABC differed by only 4 rating points regarding the different number of runs. Other factors that affect the experiment stayed the same. When the rating, rating deviation, and rating volatility for a new algorithm are set, the old algorithms' settings are also updated by taking into account only those new games with the new algorithm.

6.5. New method

CRS4EAs is a novelty within the field of empirically comparing evolutionary algorithms, and therefore has a few shortcomings that new methods usually face. CRS4EAs was first introduced in this paper by focusing on the empirical approach, whereas NHST is a proven method supported by the theory of statistics. The theoretical analysis of CRS4EAs, which is our plan for the future, would further support the appropriateness of the suggested method. Currently the CRS4EAs method is supported only by EARS (which requires a certain style of coding) but similar systems are expected to appear in the future. Hence, another problem could be the unwillingness of the researchers to use the EARS system. Researchers could have problems with their inabilities to fully control benchmark problem suites, inabilities to select algorithms for comparison, or inabilities to select their preferred programming languages. Nevertheless, the final results using EARS would be more reliable and the interpretations more objective. Since CRS4EAs is currently under development, it supports only unconstrained optimisation problems. However, the extension to constrained optimisations problems would be straightforward by applying the following simple rules:

- if both solutions are feasible, the outcome (win, loose, draw) is determined as for unconstrained problems (see Fig. 1),
- if one solution is feasible and the other infeasible, the feasible solution wins unconditionally,
- if both solutions are infeasible, the outcome is a draw.

7. Conclusions and future work

The pitfalls when conducting computational experiments within the field of Evolutionary Computing may fall into experimentation and/or statistics categories. This paper has dealt with the latter problem. Various problems under the first category are: evolutionary algorithms are not publicly available, experimental designs and executions are improperly documented regarding replications of the experiments and verifications of their results, leading to unfair comparisons between the results obtained by different experiments under different conditions. Whilst, the problems regarding the second category are: insufficient details of the statistical method used, inappropriate uses of statistics, and the derived conclusions are unsupported by computational experiments. Even worse, occasionally the statistical methods are omitted completely and any statistical significance of one algorithm over another is not shown. In such cases researchers just speculate that their algorithm outperforms others. We have come across too many papers claiming: *"It can be clearly seen from the tables that the proposed algorithm obtained much better results than other algorithms on most functions"*. Clearly, proper design, execution, and reporting of computational experiments is a difficult task. This paper proposes an alternative approach for comparing evolutionary algorithms, which might be easier to use than various statistical methods.

The idea of those ratings used amongst chess players has been applied for comparisons amongst evolutionary algorithms. A chess player with a higher rating is a stronger player and will, in most cases, outperform a lower-rated player. If the difference in ratings amongst two such players is bigger than the confidence interval, the first player is significantly better than the second player. Various chess ratings exist and in this work the Glicko-2 rating system was employed. It has been demonstrated that the Chess Rating System for Evolutionary Algorithms (CRS4EAs) can be effectively used for the comparisons and rankings of evolutionary algorithms. The results of CRS4EAs rating are comparable to the classical Null Hypothesis Significance Testing (NHST), although not the same. In an extensive experiment using 16 different evolutionary algorithms on a benchmark test suite comprised of 20 numerical optimisation problems, it was shown that CRS4EAs and NHST produced comparable results. Even whilst using different statistical methods with various powers, the same significance amongst the algorithms was not always obtained. The same also held for the CRS4EAs. However, these differences can be attributed to the difference in the conservativity/liberality of the particular approach. In CRS4EAs the conservativity is easily controlled by increasing the threshold for rating deviation. On the other hand, CRS4EAs is considerable easier to use than NHST and many of the aforementioned problems that fall into the statistical category whilst performing computational experiments can be alleviated. The following benefits of CRS4EAs have been identified:

- CRS4EAs is comparable to NHST but seems easier to use (it is unnecessary to: (1) determine the number of independent variables, (2) decide about data being parametric and non-parametric, (3) select and execute the post hoc analysis, or (4) be careful about sample size [31, p. 274]),
- CRS4EAs is less sensitive to outliers than NHST,
- CRS4EAs' conservativity/liberality is easy to control,
- the ranking of newly-developed algorithms can be more efficiently computed than using NHST,
- CRS4EAs is implemented within an EARS environment, that is publicly available. EARS promotes a publicly-available code for evolutionary algorithms, a standard environment for executing computational experiments, and the customisation of benchmark test suites.

As for the future work, we would like to compare CRS4EAs and NHST, not only by computational experiments but also by sound theoretical analysis, which would strengthen the confidence of our approach. We also have a desire that EARS would become a standard platform for comparing evolutionary algorithms, where researchers will publicly publish their newly-developed evolutionary algorithms, having evaluated them objectively on various benchmark test suites. Evaluation by third parties who do not have their own interests in promoting a particular algorithm might be much more objective. By participating in a few tournaments a newly-developed algorithm can quickly obtain its rating, thus indicating the power and performance of this algorithm. EARS can also be used by journals on evolutionary computations for easier evaluations of newly-proposed algorithms. The vision is to better assist the reviewers, who often need to only evaluate newly-developed algorithms using very abstract pseudo codes where many important details are not described.

References

[1] M. Ali, M. Pant, A. Abraham, Improving differential algorithm by synergizing different improvement mechanisms, ACM Trans. Auton. Adapt. Syst. 7 (2) (2012) 20:1–20:32.
[2] H.M. Alizadeh, A.A. Khamseh, S.F. Ghomi, Fuzzy hypothesis testing with vague data using likelihood ratio test, Soft Comput. 17 (2013) 1629–1641.
[3] M. Altman, J. Gill, M.P. McDonald, Numerical Issues in Statistical Computing for the Social Scientist, vol. 508, John Wiley & Sons, 2004.
[4] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computations, Oxford University Press, 1996.
[5] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, W.R. Stewart Jr., Designing and reporting on computational experiments with heuristic methods, J. Metaheur. 1 (1995) 9–32.
[6] T. Bartz-Beielstein, Experimental Research in Evolutionary Computation, Springer, 2006.
[7] V.W. Berger, A Priori v Post Hoc Testing Encyclopedia of Statistics in Behavioral Science, John Wiley & Sons, 2005.
[8] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[9] C. Blum, J. Puchinger, G.A. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.
[10] R.A. Bradley, M.E. Terry, Rank analysis of incomplete block designs: I. The method of paired comparisons, Biometrika 39 (3/4) (1952) 324–345.
[11] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evolution. Comput. 10 (6) (2006) 646–657.
[12] J. Brest, P. Korošec, J. Šilc, A. Zamuda, B. Bošković, M.S. Maučec, Differential evolution and differential ant-stigmergy on dynamic optimisation problems, Int. J. Syst. Sci. 44 (4) (2013) 663–679.
[13] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Inform. Sci. 227 (2013) 60–82.
[14] J. Cohen, The earth is round ($p < .05$), Am. Psychol. 49 (12) (1994) 997–1003.
[15] M. Črepinšek, S.H. Liu, L. Mernik, A note on teaching–learning-based optimization algorithm, Inform. Sci. 212 (2012) 79–93.
[16] M. Črepinšek, S.H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. 45(3) (2013) 35.
[17] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evolut. Comput. 15 (1) (2011) 4–31.
[18] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[19] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[20] M. Dowell, P. Jarratt, A modified regula falsi method for computing the root of an equation, BIT Numer. Math. 11 (2) (1971) 168–174.
[21] O.J. Dunn, Multiple comparisons among means, J. Am. Statist. Assoc. 56 (1961) 52–64.
[22] C.W. Dunnett, A multiple comparison procedure for comparing several treatments with a control, J. Am. Statist. Assoc. 50 (1980) 1096–1121.
[23] T. Dybå, V.B. Kampenes, D.I. Sjøberg, A systematic review of statistical power in software engineering experiments, Inform. Softw. Technol. 48 (8) (2006) 745–755.
[24] Efficient Java Matrix Library <http://code.google.com/p/efficient-java-matrix-library>, 2013.

[25] Á.E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, in: Proceedings of the 2002 Congress (CEC'2002), vol. 1, 2002, pp. 582–587.
[26] Á.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, 2008.
[27] A.E. Elo, The Rating of Chessplayers, Past and Present, vol. 3, Batsford, 1978.
[28] Evolutionary Algorithms Rating System <http://earatingsystem.appspot.com/> (version 1), 2013.
[29] Evolutionary Algorithms Rating System (Github) <https://github.com/matejxxx/EARS> (version 1), 2013.
[30] Experimental results for CRS4EAs (raw data) <http://lpm.uni-mb.si/crepinsek/dataINS.txt>, 2013.
[31] A.P. Field, G. Hole, How to Design and Report Experiments, Sage publications, 2003.
[32] H. Finner, On a monotonicity problem in step-down multiple test procedures, J. Am. Statist. Assoc. 88 (1993) 920–923.
[33] R.A. Fisher, Statistical Methods for Research Workers, Oliver & Boyd, 1925.
[34] R.A. Fisher, The Design of Experiments, Oliver & Boyd, 1935.
[35] R.A. Fisher, Statistical Methods and Scientific Inference, second ed., Hafner Publishing Co., 1959.
[36] R.A. Fisher, Statistical Methods, Experimental Design, and Scientific Inference: A Re-Issue of Statistical Methods for Research Workers, The Design of Experiments, and Statistical Methods and Scientific Inference, Oxford University Press, 1995.
[37] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Statist. Assoc. 326 (1937) 75–701.
[38] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Statist. 11 (1940) 86–92.
[39] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inform. Sci. 180 (10) (2010) 2044–2064.
[40] S. García, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[41] J. Gill, The insignificance of null hypothesis significance testing, Polit. Res. Quart. 52 (3) (1999) 647–674.
[42] M.E. Glickman, A comprehensive guide to chess ratings, Am. Chess J. 3 (1995) 59–102.
[43] M.E. Glickman, The Glicko System, Boston University, 1995.
[44] M.E. Glickman, Parameter estimation in large dynamic paired comparison experiments, J. Roy. Statist. Soc.: Ser. C (Appl. Statist.) 48 (3) (1999) 377–394.
[45] M.E. Glickman, Dynamic paired comparison models with stochastic variances, J. Appl. Statist. 28 (6) (2001) 673–689.
[46] M.E. Glickman, Example of the Glicko-2 System, Boston University, 2012.
[47] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans. Cybernet. 1–16 (2013).
[48] N. Hansen, The CMA Evolution Strategy: A Comparing Review, Towards A New Evolutionary Computation, Springer, 2006. pp. 75–102.
[49] L.L. Harlow, S.A. Mulaik, What If There Were No Significance Tests?, Lawrence Erlbaum Associates, 1997
[50] K. Harkness, Official Chess Handbook, D. McKay Company, 1967.
[51] Y. Hochberg, A sharper Bonferroni procedure for multiple tests of significance, Biometrika 75 (1988) 800–803.
[52] J.L. Hodges, E.L. Lehmann, Rank methods for combination of independent experiments in analysis of variance, The Ann. Math. Statist. 33 (2) (1962) 482–497.
[53] B.S. Holland, M.D. Copenhaver, An improved sequentially rejective Bonferroni test procedure, Biometrics 43 (1987) 417–423.
[54] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Statist. 6 (1979) 65–70.
[55] G. Hommel, A stagewise rejective multiple test procedure based on a modified Bonferroni test, Biometrika 75 (1988) 383–386.
[56] T. Hoshino, Bayesian significance testing and multiple comparisons from MCMC outputs, Comput. Statist. Data Anal. 52 (7) (2008) 3543–3559.
[57] G. Iacca, F. Neri, E. Mininno, Y.S. Ong, M.H. Lim, Ockham's razor in memetic computing: three stage optimal memetic exploration, Inform. Sci. 188 (2012) 17–43.
[58] R.L. Iman, J.M. Davenport, Approximations of the critical region of the Friedman statistic, Commun. Statist. – Theory Meth. 9 (6) (1980) 571–595.
[59] G. Jia, Y. Wang, Z. Cai, Y. Jin, An improved (<mu>+<lambda>)-constrained differential evolution for constrained optimization, Inform. Sci. 222 (2013) 302–322.
[60] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[61] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Trans. Softw. Eng. 28 (8) (2002) 721–734.
[62] A. Knezevic, StatNews #73: Overlapping Confidence Intervals and Statistical Significance, Cornell Statistical Consulting Unit, 2008.
[63] P. Korošec, J. Šilc, B. Filipič, The differential ant-stigmergy algorithm, Inform. Sci. 192 (1) (2012) 82–96.
[64] T.R. Levine, R. Weber, C. Hullett, H.S. Park, L.L.M. Lindsey, A critical assessment of null hypothesis significance testing in quantitative communication research, Hum. Commun. Res. 34 (2) (2008) 171–187.
[65] J. Li, A two-step rejection procedure for testing multiple hypotheses, J. Statist. Plann. Infer. 138 (2008) 1521–1527.
[66] D.V. Lindley, New Cambridge Statistical Tables, Cambridge University Press, 1995.
[67] S.-H. Liu, M. Mernik, D. Hrnčič, M. Črepinšek, A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model, Appl. Soft Comput. 13 (9) (2013) 3792–3805.
[68] P.B. Nemenyi, Distribution-Free Multiple Comparisons, Ph.D. thesis, Princeton University, 1963.
[69] F. Neri, E. Mininno, G. Iacca, Compact particle swarm optimization, Inform. Sci. 239 (2013) 96–121.
[70] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, R Artif. Intell. Rev. 33 (1-2) (2010) 61–106.
[71] J. Neyman, E. Pearson, On the problem of the most efficient test of statistical hypothesis, Philos. Trans. R. Soc. Lond. – Ser. A 231 (1933) 289–337.
[72] B. Pandolfini, Weapons of Chess: An Omnibus of Chess Strategies, Touchstone, 1989.
[73] F. Pukelsheim, The three sigma rule, Am. Statist. 48 (2) (1994) 88–91.
[74] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems, Inform. Sci. 183 (1) (2012) 1–15.
[75] L.A. Rastrigin, The convergence of the random search method in the extremal control of a many-parameter system, Autom. Rem. Control 24 (10) (1963) 1337–1342.
[76] I. Rechenberg, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog (1973).
[77] D.M. Rom, A sequentially rejective test procedure based on a modified Bonferroni inequality, Biometrika 77 (1990) 663–665.
[78] D. Quade, Using weighted rankings in the analysis of complete blocks with additive block effects, J. Am. Statist. Assoc. 74 (367) (1979) 680–683.
[79] N. Schenker, J.F. Gentleman, On judging the significance of differences by examining the overlap between confidence intervals, Am. Statist. 55 (3) (2001) 182–186.
[80] F.L. Schmid, Statistical significance testing and cumulative knowledge in psychology, Psychol. Meth. 1 (1996) 115–129.
[81] J.P. Shaffer, Modified sequentially rejective multiple test procedures, J. Am. Statist. Assoc. 81 (395) (1986) 826–831.
[82] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & HallCRC, 2006.
[83] D. Shilane, J. Martikainen, S. Dudoit, S. Ovaska, A general framework for statistical performance comparisons of evolutionary computation algorithms, Inform. Sci. 178 (14) (2008) 2870–2879.
[84] R.G. Steel, A multiple comparison sign test: treatments versus control, J. Am. Statist. Assoc. 54 (1959) 767–775.
[85] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[86] Student, The probable error of a mean, Biometrika (1908) 1–25.

[87] K. Tang, X. Li, P.N. Suganthan, Z. Yang, T. Weise. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization, Nature Inspired Computation and Applications Laboratory, 2009.
[88] The CMA-ES Source Code <https://www.lri.fr/~hansen/cmaes_inmatlab.html>, 2013.
[89] J.W. Tukey, Comparing individual means in the analysis of variance, Biometrics 5 (1949) 99–114.
[90] J. Tvrdík, Adaptive differential evolution: application to nonlinear regression, in: Proceedings of the International Multiconference on Computer Science and Information Technology, 2007, pp. 193–202.
[91] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (6) (1945) 80–83.