Towards Better Estimation of Statistical Significance When Comparing Evolutionary Algorithms

Maxim Buzdalov
ITMO University
Saint Petersburg, Russia
mbuzdalov@gmail.com

ABSTRACT

The use of well-established statistical testing procedures to compare the performance of evolutionary algorithms often yields pessimistic results. This requires increasing the number of independent samples, and thus the computation time, in order to get results with the necessary precision.

We aim at improving this situation by developing statistical tests that are good in answering typical questions coming from benchmarking of evolutionary algorithms. Our first step, presented in this paper, is a procedure that determines whether the performance distributions of two given algorithms are identical for each of the benchmarks. Our experimental study shows that this procedure is able to spot very small differences in the performance of algorithms while requiring computational budgets which are by an order of magnitude smaller (e.g. 15x) compared to the existing approaches.

CCS CONCEPTS

• **Mathematics of computing → Hypothesis testing and confidence interval computation**; **Nonparametric statistics**; *Statistical software.*

KEYWORDS

Multiple comparisons, statistical significance.

ACM Reference Format:
Maxim Buzdalov. 2019. Towards Better Estimation of Statistical Significance When Comparing Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3319619.3326899

1 INTRODUCTION

When a new randomized search heuristic is proposed, it shall be compared with other existing algorithms on benchmark problems, as well as on real-world problems when applicable. The performance is most often measured either as the fitness value after a fixed number of fitness function evaluations, or as the number of fitness function evaluations to reach a certain fitness threshold. The stochastic nature of randomized search heuristics implies that the performance (either of the measures from above), on each problem and even on each problem instance, is a random variable, which necessitates the use of statistical methods to derive conclusions about the mutual relation of these random variables. As a result, statistical testing is widely used in evolutionary computation.

Statistical testing is a well-established discipline with very important applications both in practical fields (medicine and finances) and in highly-developed theoretic fields (such as high-energy physics). However, it seems that most of available general-purpose tools are optimized for the usages that are noticeably different from typical usages within evolutionary computation (and computational intelligence in general). For instance, a typical use of a statistical test in a medical context is to find out which of K treatments results in a significant change of the observable state or behaviour, which implies that such a significant change is observed very rarely.

On the other hand, a typical scenario in evolutionary computation is to compare the performance of two (or more) algorithms on K possibly unrelated benchmark problems and to derive conclusions on the relationship between these algorithms. This setting seems to occur rarely in areas that heavily influenced statistics, so methods that address such questions are scarce. Attempts to use existing general-purpose tools, that are tailored to answer other questions, result in rather pessimistic results. In particular, this problem requires conducting much more experiments in order to get results of desired precision.

The **contribution** of this paper is the statistical testing procedure, developed specially for benchmarking of randomized search heuristics, that aims at determining whether two algorithms, on each of K possibly unrelated problem instances, have an identical distribution of the performance measure, given that on the i-th of these instances there were N_i^1 independent runs of the first algorithm and N_i^2 independent runs of the second algorithm. The procedure consists of two stages. First, for each problem instance, the two-sided Kolmogorov-Smirnov test [10] is conducted. Second, for each of the K outcomes of these tests (we use Kolmogorov-Smirnov statistic values), its rank among the possible outcomes for the particular input dimensions is evaluated, and then the sum of these ranks is used as a statistic, for which the procedure of deriving an exact p-value is proposed.

While testing the identity of performance distributions of two algorithms is probably not the most important problem in algorithm benchmarking, we indicate that the proposed principle of construction of statistical testing procedures is quite general and can be used for answering more important questions, including, but not limited to, testing for statistical dominance.

The rest of the paper is structured as follows. Section 2 explains the necessary terms and overviews the related work. In Section 3

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326899

we propose our rank-sum result joining procedure, which aggregates the outcomes of multiple individual statistical tests to produce a resulting p-value for the single statistical hypothesis that shall hold across multiple benchmark problems. Section 4 presents our experimental study of the proposed procedure, including one artificial setting and one setting related to experimental investigation of evolutionary algorithms, as well as comparison with the existing approaches. Section 5 concludes the paper and gives final remarks.

2 PRELIMINARIES AND RELATED WORK

First we introduce the notation used throughout the paper. We denote as $[1..n]$ the set of integer numbers $\{1, 2, \ldots, n\}$. By $\mathcal{N}(m, \sigma^2)$ we denote a Gaussian distribution with mean m and variance σ^2.

The rest of this section recalls various topics which this paper uses, such as the basics of hypothesis testing, a few statistical tests, as well as several evolutionary algorithms and benchmark problems.

2.1 Hypothesis Testing

Statistical hypothesis testing is a method of statistical inference. Usually, two hypotheses about the studied process are formulated: the *null hypothesis* $\mathcal{H}_0$ assumes a default position which is typically an undesired result (for instance, the newly proposed evolutionary algorithm is not better than the existing ones on the available benchmarks), and the *alternative hypothesis* $\mathcal{H}_1$ is its negation [14]. The observed behaviour of the process is then investigated for the probability to observe it assuming $\mathcal{H}_0$ is true. If this probability is small enough (e.g. below some application-dependent threshold), then the difference to $\mathcal{H}_0$ is inferenced to be *statistically significant*, and the null hypothesis is *rejected* in the favour of the alternative hypothesis.

In this paper, we consciously stay away from the extensive (and, at times, furious) discussion about pros and cons of statistical hypothesis testing (the approach used in this paper) versus Bayesian statistics [1]. Our current opinion is that, although the question of whether p-values are a usable tool is intricate, Bayesian statistics are themselves, as of now, still too much of an art than of a tool.

Many commonly used statistical tests follow a particular framework. First, a certain numeric value is evaluated based on the observable data, which has the semantic of being the more extreme (e.g. greater), the less probable it is to observe such data assuming $\mathcal{H}_0$. This value is often called the *statistic* related to the corresponding statistical test, e.g. the Kolmogorov-Smirnov statistic. Second, this numeric value is either compared with the threshold values for the needed significance level α pre-computed for the data sizes, or (as more common now) is converted, using either an exact or an approximate algorithm, into another numeric value, called the *p-value*, which is the probability of observing data which is at least that extreme assuming $\mathcal{H}_0$. If the statistic value is too extreme, or, equivalently, the p-value is smaller than the significance level α, $\mathcal{H}_0$ is rejected. The p-value is often used to assess the strength of the difference between $\mathcal{H}_0$ and the actual situation: the less it is, the stronger is the observed difference.

The mathematics behind particular statistical tests may be valid only under certain assumptions. For instance, many commonly used tests assume normality of the data, such as the ANOVA family of tests [11]. However, in computational intelligence, including evolutionary computation, many common random variables that undergo statistical testing violate most of common assumptions (e.g. the number of fitness evaluations until the optimum is found, even of simple algorithms and on simple problems, is significantly different from the normal distribution). For this reason, *non-parametric* tests are more appropriate in these areas. These tests were originally developed to work with ordinal data and thus have much fewer assumptions about the data [9]. In applications where the assumptions of the commonly used tests hold, parametric tests typically have more resolution power than non-parametric ones (i.e. produce smaller p-values on same inputs). We will focus on non-parametric tests in the rest of the paper.

2.2 Hypothesis Testing in Evolutionary Computation

A survey on application of statistical testing for evolutionary computation [4] distinguishes the following groups of non-parametric statistical tests:

- *Pairwise tests* take the results of measurements of two algorithms on one problem and derive their statistics from a certain property of these measurements. Most often (and the only case considered in [4]) this is a median, as in the sign test [2, Chapter 3.4] and the Mann-Whitney U test [12], also known as the Wilcoxon rank-sum test [16]. Other tests, such as the Kolmogorov-Smirnov test [10, 15], test whether the cumulative distribution functions are the same.
- *Tests for multiple comparisons.* There are two different categories of tests that appeared under the same category in [4]:
  - *Paired difference tests.* Here it is assumed that the two compared algorithms were run *once* per each of K benchmarks, and we wish to derive conclusions based on comparisons within these pairs. The multiple sign test [2], which counts in how many benchmarks the first algorithm is better than the second one and derives the conclusions from this statistic, falls into this category. Another example of such a test is the Wilcoxon signed-rank test [16]. This test is more powerful for a price of a stronger assumption that the differences between the outcomes of the algorithms are ordered (i.e. can be compared).
  - *Multiple comparison tests.* These tests take the results of $k \geq 2$ algorithms on the same benchmark and check whether *all* of the algorithms perform the same. The most known example, the Friedman test [7], derives the statement about the medians. In a typical evolutionary computation scenario, this outcome is often of a little use, so more tests need to be conducted once a null hypothesis of a multiple comparison test is rejected.
- *Post-hoc procedures*, or *corrections for multiple comparisons.* As the p-value, which often serves as the main outcome of a statistical test, is itself a random variable, it has a chance to fall below any predefined significance level α with probability of roughly $1/\alpha$ even when $\mathcal{H}_0$ cannot be rejected. As a result, when several statistical tests need to be conducted, some of them might trigger a false positive just by chance. To compensate for this, the set of p-values need to be adjusted using one of the appropriate procedures, such as the

Algorithm 1 The pseudocode of Randomized Local Search

```
Choose x ∈ {0,1}^n uniformly at random
Evaluate x
while stopping criterion not fulfilled do
    Choose i ∈ [1..n] uniformly at random
    y ← x with the i-th bit flipped
    Evaluate y
    if f(y) ≥ f(x) then
        x ← y
    end if
end while
```

Holm-Bonferroni correction [6], before comparing with the significance level. As a result, one needs to aim at much lower individual p-values when designing the experiment with multiple intended statistical comparisons, which needs to be roughly α/K for K comparisons.

2.3 Statistical Tests Used in This Paper

This subsection contains short descriptions of the statistical tests used in this paper.

The *Kolmogorov-Smirnov test* [10, 15] uses the maximum difference between empirical cumulative distribution functions to reject the null hypothesis that the distributions behind the measured outcomes are identical. Both exact procedures for derivation of a p-value [10] and a large-number approximation [15] are known. We are going to use the former, so that the precision for small samples is not reduced.

The null of *Wilcoxon rank-sum test* [12, 16] is that the medians of the distributions are identical, and the statistic is the sum of ranks of the samples from one of the sides (either a smaller or a larger of the two values is usually chosen) when all the samples are merged, sorted and ranked.

The *Wilcoxon signed rank test* [16] tests whether the differences between the outcomes of two algorithms on a series of benchmarks are symmetrically distributed around the zero (the null hypothesis is that the algorithms perform the same). To do that, it sorts the differences by the absolute values and evaluates the sum of the ranks, taken with the sign matching the sign of the corresponding difference, as a statistic.

All these tests assume by default that *response is continuous*, that is, the probability of hitting two identical outcomes is exactly zero. Numerous extensions exist for these tests; however, for the Kolmogorov-Smirnov test it is known that the null hypothesis *cannot* be mistakenly rejected when some parts of the tested distributions are discrete.

2.4 Algorithms and Problems Used in Experiments

The class of functions OneMax is defined on bit strings of length n as follows:

$$\textsc{OneMax}_z : \{0,1\}^n \to \mathbb{R}; x \mapsto |\{i \in [1..n] \mid x_i = z_i\}|.$$

The simple local optimizer called Randomized Local Search (RLS, see Algorithm 1) optimizes functions defined on bit strings of length n iteratively by flipping one randomly chosen bit on each iteration and replacing the parent with the offspring if the latter is not worse. Its flavour, which takes the best of k randomly sampled individuals as the initial individiual, is called RLS_k (see Algorithm 2). It was theoretically studied in [3] and found to outperform RLS by a roughly $\sqrt{n \log n}$ margin when k is chosen optimally.

Algorithm 2 The pseudocode of RLS_k

```
Choose x ∈ {0,1}^n uniformly at random
Evaluate x
for i ← {2,...,k} do                ▷ Extended initialization phase
    Choose z ∈ {0,1}^n uniformly at random
    Evaluate z
    if f(z) ≥ f(x) then
        x ← z
    end if
end for
while stopping criterion not fulfilled do        ▷ General phase
    Choose i ∈ [1..n] uniformly at random
    y ← x with the i-th bit flipped
    Evaluate y
    if f(y) ≥ f(x) then
        x ← y
    end if
end while
```

3 THE METHOD: RANK-SUM RESULT JOINER

In this section, we are going to explain our method for statistical comparison of two algorithms when there are K benchmarks, and there are *multiple* runs of every algorithm on every benchmark. Our method assumes independence of all measurmenets and works will hypotheses that cover the relations of algorithms as a whole (e.g. a typical $\mathcal{H}_0$ would be that the two algorithms behave identically on *all* benchmarks), and the result of the application is a *single* p-value. In the following, we first give our motivation about the particular design of the method, and then describe the method formally.

3.1 Motivation for the Design

We consider the situation with multiple benchmarks and multiple measures. The usual assumption within evolutionary computation is that every measure takes some resources, so we would like to get as much precision as possible from every measure to derive the necessary "global" conclusions.

This effectively prevents us from using paired difference tests, where the improvement following from using multiple measures for every benchmark over a single measurement is not large: it is typically advised that the measurements for a single benchmark are averaged, or a median is taken, and the paired difference test is then run on these averages [4]. As a result, the precision of the final result grows with the number of benchmarks, but only marginally with the number of samples.

Another possible scenario is to conduct multiple pairwise tests, one for each benchmark, and then to perform a post-hoc correction on the results of these tests. The "global" $\mathcal{H}_0$ will be rejected,

depending on the setting, if either at least one of the multiple comparison survives (e.g. when checking whether algorithms behave identically) or *all* of them survive (e.g. when checking stochastic dominance, as a single failure might be an indication of the reversal of domination for a particular benchmark). In this case, increasing the number of samples (for each benchmark) improves the results, but when the number of benchmarks increases, the results can get worse, as the corrections need to become more aggressive.

More precisely, many corrections are designed in such a way that the (possibly unknown) dependencies between the individual samples can not result in a false rejection of $\mathcal{H}_0$. For instance, the Holm-Bonferroni correction takes the p-values $p_1, \ldots, p_k$, sorted in a non-decreasing order, and rejects only hypotheses indexed by $[1..t]$ such that, for all $1 \le i \le t$, $p_i \le \alpha/(k+1-i)$ for a significance level α. With this correction, the measurement series with p-values $[0.01, 0.3, 0.4, 0.5, 0.6, 0.7]$ will not result in rejection of $\mathcal{H}_0$ at the level $\alpha = 0.05$ even for the first measurement, as $0.01 > 0.05/6$. However, it will conservatively reject as well the entire series $[0.01, 0.01, 0.01, 0.01, 0.01, 0.01]$, as it assumes that all these measurements can, in the worst case, be the same identical measurement. In the context of truly independent samples, which prevails in evolutionary computation, such a series shall definitely be an indication of statistical significance.

We are going to construct a method which would profit *both* from increasing the number of samples *as well as* the number of benchmarks. At the same time it is still able to ignore spurious false positives when the rest of the measurements is not up to that.

3.2 The Structure of the Method

Our method uses an existing statistical test $\mathcal{S}$. Assuming its null hypothesis is $\mathcal{H}_0$, the null hypothesis of our method that uses $\mathcal{S}$ is that $\mathcal{H}_0$ holds for *every* benchmark out of K benchmarks on which the algorithms are compared, so a proper rejection of just one such hypothesis is enough to reject the "global" null hypothesis. We assume that the test $\mathcal{S}$ is based on computing a statistic $s \in S$, such that S is the ordered set of all possible values for this statistic, and then on computing a p-value $p = p_{\mathcal{S}}(s)$, such that, without loss of generality, whenever $s_1 < s_2$, it holds that $p_{\mathcal{S}}(s_1) > p_{\mathcal{S}}(s_2)$.

The set S, as well as the function $p_{\mathcal{S}} : S \to \mathbb{R}$, are allowed to depend on the number of samples in a benchmark, i.e. N_i^1 is the number of samples of the first compared algorithm for the i-th benchmark, and N_i^2 is for the second compared algorithm, so the actual notation will be S_i and $p_{\mathcal{S},i}$. However, we leave out the index i when it is safe.

Our method consists of the following phases. First, it runs the statistical test $\mathcal{S}$ on the outcomes of the algorithms for every benchmark, and records the received statistics s_i along with the sample sizes N_i^1 and N_i^2, as well as the sets of possible statistic values S_i. Second, it computes, for each i, the rank R_i of s_i, such that $0 \le R_i < |S_i|$, among all possible elements of S_i ordered by the associated p-value, along with the probabilities to observe each element of S_i. To do that, for each i it performs the following:

- For every j, $1 \le j \le |S|$, assuming $s_j \in S$ are ordered according to their indices, the value $q_j = p_{\mathcal{S}}(s_j)$ is computed, taking into account the sample sizes N^1 and N^2. Note that q_1 is the greatest value and $q_{|S|}$ is the smallest one.
- For every j, the probability of observing exactly s_j is computed as $r_j = q_j - q_{j+1}$, assuming $q_{|S|+1} = 0$.

Third, the rank-sum statistic $R = \sum_i R_i$ is computed. Note that, as greater values of R_i correspond to more extreme values, greater values of R also correspond to more extreme observations. Fourth, for every possible sum of ranks σ, $0 \le \sigma \le \sum_i (|S_i| - 1)$, it computes, using the values $r_{i,j}$, the probability π_σ of observing exactly the rank σ assuming all $\mathcal{H}_0$ hypotheses hold for every benchmark. It is possible to do in time polynomial in the size of the statistic sets, more precisely, in $O((\sum_i |S_i|)^2)$, using straightforward dynamic programming. Finally, the resulting global p-value is computed by definition as:

$$p = \sum_{\sigma=R}^{\sum_i (|S_i|-1)} \pi_\sigma .$$

Note that the algorithm above computes the p-value, corresponding to the intersection of the null hypotheses about the individual benchmarks, essentially by definition, using the supplied statistical test $\mathcal{S}$ in an almost black-box protocol. More precisely, we require such a statistical test to give away the raw statistic value (which is often done in practical implementations, such as in the one from the R system [13]), as well as to provide the set of possible statistic values for the given sample sizes, which is also often possible for both exact and approximated tests, but may require some knowledge of the internals of the test. In this paper we limit ourselves to exact computations, leaving large-size approximations for the future work.

4 EXPERIMENTS

In this section, we present and discuss our first experimental results regarding the proposed method. We limit ourselves to testing whether the performance distribution of the algorithms is identical, and thus to the two-sided Kolmogorov-Smirnov test serving as the test $\mathcal{S}$ inside our method. Our main experiments, each of which assumes running the underlying algorithms for multiple times, will be run in multiple trials themselves to assess the variability of the produced p-values. These p-values do not carry a paper-wide meaning, which means that we do not have to apply corrections for multiple comparisons on them, although it does not allow the potential users of our method to do the same in their research.

4.1 A Smoke Test

The first set of experiments is designed to demonstrate the behaviour of the proposed method on two possible use cases, namely, when the difference between the designed random variables does exist and when it does not. It shall clearly produce low p-values in the first case and high p-values in the second case. In order to perform the experiments, the following functions were designed:

$$\mathcal{T}_1(n) = n \ln(n+1) \cdot (1.00 + 0.1 \cdot \mathcal{N}(0,1));$$
$$\mathcal{T}_2(n) = n \ln(n+1) \cdot (1.05 + 0.1 \cdot \mathcal{N}(0,1)).$$

These functions are chosen in such a way that, for every n, the corresponding random variables substantially overlap, but the overlap ratio is fixed and does not depend on n. In the first experiment we compare $\mathcal{T}_1(n)$ and $\mathcal{T}_2(n)$, and in the second experiment we compare $\mathcal{T}_1(n)$ with itself. To perform the comparison, we sample, for

every $n \in [1..200]$, five realizations of each side of the comparison, with every value of n serving as a separate benchmark, and then use the Kolmogorov-Smirnov test to estimate the differences between the distribution functions. The results of experimental comparison of $\mathcal{T}_1$ and $\mathcal{T}_2$ are presented in Table 1, while the results of comparing $\mathcal{T}_1$ with itself are presented in Table 2. These tables illustrate five independent trials of comparisons, one trial per row. They also display the statistics of the individual p-values reported for various n using the Kolmogorov-Smirnov test.

Note that, based on just five samples for every problem size n, we achieve a minimum p-value of approximately 0.007937 for every separate run of the Kolmogorov-Smirnov test. We can see that in Table 1 only at most 7 out of 200 comparisons ended up with $p = 0.007937$, while all others result in $p > 0.079$. As a result, the Holm-Bonferroni correction, in either the Holm step-down or the Hochberg step-up [8] flavour, will not reject any of the null hypotheses when comparing $\mathcal{T}_1$ and $\mathcal{T}_2$ with the sample size of five, 200 comparisons and the significance level $\alpha = 0.05$. The proposed method, on the contrary, consistently produced global p-values of $10^{-4} \ldots 10^{-6}$, so it distinguished the slightly differing functions.

On the other hand, just as expected, the null hypothesis was not rejected in any of the trials regarding comparison of $\mathcal{T}_1$ with itself (Table 2). The observed p-values happened to be all greater than $1/3$, which is way above any common significance thresholds, despite the p-values resulting from individual comparisons occasionally fall below 0.08 and, once or twice in 200 runs, even below 0.008. This indicates that our method is capable of getting rid of spurious false positives resulting from occasionally producing non-overlapping measurements for some of the benchmarks, as long as there is no significant difference in the remaining benchmarks.

Note that our method derives its conclusions not solely from the "good-looking" outcomes of the statistical tests: for example, the fourth trial in Table 1 has only one benchmark with the smallest p-value, similar to three trials in Table 2, however, the global p-value is still well below 10^{-4} thanks mostly to a completely different distribution of benchmarks among other p-values, which are themselves well above the commonly used significance thresholds.

To assess the benefits of our method from the computational budget perspective, we conducted a similar study, involving comparison of $\mathcal{T}_1$ with $\mathcal{T}_2$, using the Kolmogorov-Smirnov test to derive a p-value for each of the 200 problem sizes, but this time with $k \in \{20, 40, 60, 80, 100\}$ samples for each problem size. The resulting p-values are then processed using the Holm-Bonferroni correction logic, and the minimum among the corrected p-values is reported (Table 3). From these results we can observe that it takes $k \geq 80$ to reach the same order of p-values which are possible with our method and $k = 5$, resulting in a budget reduction of roughly 15x.

4.2 Spotting the Impact of Better Initialization

In the paper [3], an impact of having a better initialization on the expected running time has been studied. More precisely, the OneMax problem was considered, randomized local search (RLS) was chosen for the algorithm, and the *best-of-k* strategy was analyzed for choosing the initial individual. With the runtime of RLS on OneMax is known to be $\Theta(n \log n)$ for the problem size n (more precisely, it is $nH_{n/2} - 1/2 \pm o(1)$, where H_t is the t-th harmonic number [5]), the best-of-k strategy results in a roughly $\sqrt{n \log n} + O(\sqrt{n})$ additive speed-up when $k = \sqrt{n/\log n}$ is chosen — even despite the fact that sampling these k individuals also counts toward the total number of fitness evaluations.

Table 1: Results of smoke testing, $\mathcal{T}_1$ versus $\mathcal{T}_2$. Each row represents a single independent trial of 200×5 measurements for both sides.

Number of n with this p-value (sums up to 200)					Global
0.00794	0.07937	0.35714	0.87302	1.00000	p-value
6	24	70	84	16	$1.043 \cdot 10^{-6}$
4	28	59	86	23	$3.169 \cdot 10^{-4}$
3	27	71	81	18	$5.060 \cdot 10^{-6}$
1	28	66	89	16	$6.309 \cdot 10^{-5}$
7	22	74	76	21	$2.326 \cdot 10^{-6}$

Table 2: Results of smoke testing, $\mathcal{T}_1$ versus $\mathcal{T}_1$. Each row represents a single independent trial of 200×5 measurements for both sides.

Number of n with this p-value (sums up to 200)					Global
0.00794	0.07937	0.35714	0.87302	1.00000	p-value
2	13	50	116	19	0.532
2	14	55	107	22	0.395
1	22	57	89	31	0.216
1	15	50	106	28	0.782
1	16	55	105	23	0.395

Note that, contrary to the previous setting, the expected additive speed-up is of a lower order of magnitude compared to the actual running time. For instance, the theoretically proven difference between the expected running times is roughly 42 for $n = 1000$ when $k = 16$ is chosen [3, Fig. 1]. Compared to the expected runtime RLS on OneMax of roughly 6792.32, especially paying attention to the noticeable standard deviation of this random variable, this difference is quite small.

We aim at spotting the difference between the original RLS and RLS_k that uses the best-of-k strategy in experimental results. Given the considerations above, this is quite a hard problem. Our experimental setup for this problem is as follows. We take different values of $n \in \{100, 200, 300, 400, 500\}$ and different number of samples $s \in \{100, 200, 300, 400, 500\}$. For each combination of n and s, we make up a problem with n benchmarks, each having a problem size

Table 3: The minimum p-values after the Holm-Bonferroni-style correction: $p_i \leftarrow p_i \cdot (k - i + 1)$, where k is the number of samples and p_i are initially sorted. Each column consists of five independent trials, columns are sorted.

$k = 20$	$k = 40$	$k = 60$	$k = 80$	$k = 100$
0.054	$2.469 \cdot 10^{-5}$	$2.317 \cdot 10^{-5}$	$3.754 \cdot 10^{-7}$	$8.828 \cdot 10^{-7}$
0.054	$2.469 \cdot 10^{-5}$	$7.028 \cdot 10^{-5}$	$8.543 \cdot 10^{-6}$	$1.323 \cdot 10^{-5}$
0.054	$3.776 \cdot 10^{-4}$	$7.028 \cdot 10^{-5}$	$2.272 \cdot 10^{-5}$	$3.079 \cdot 10^{-5}$
0.054	$1.322 \cdot 10^{-3}$	$5.646 \cdot 10^{-4}$	$2.272 \cdot 10^{-5}$	$3.095 \cdot 10^{-5}$
0.222	$1.322 \cdot 10^{-3}$	$1.506 \cdot 10^{-3}$	$1.457 \cdot 10^{-4}$	$3.110 \cdot 10^{-5}$

Table 4: The derived p-values for experiments comparing RLS and RLS_k (the *Diff* columns), as well as RLS with itself (the *Same* columns). For every n and every number of samples, five independent experiment runs were conducted, and the resulting p-values are written columnwise and sorted, smaller values at the top.

n	100 samples each		200 samples each		300 samples each		400 samples each		500 samples each	
	Same	Diff	Same	Diff	Same	Diff	Same	Diff	Same	Diff
100	0.690	0.025	0.832	$2.722 \cdot 10^{-6}$	0.744	$9.108 \cdot 10^{-7}$	0.349	$4.066 \cdot 10^{-14}$	0.182	$5.051 \cdot 10^{-18}$
	0.833	0.032	0.977	$3.528 \cdot 10^{-4}$	0.883	$6.253 \cdot 10^{-6}$	0.768	$2.869 \cdot 10^{-10}$	0.969	$1.723 \cdot 10^{-16}$
	0.937	0.047	0.991	$6.534 \cdot 10^{-3}$	0.958	$2.048 \cdot 10^{-5}$	0.952	$3.577 \cdot 10^{-10}$	0.988	$2.252 \cdot 10^{-14}$
	0.967	0.053	0.996	0.012	0.965	$2.129 \cdot 10^{-3}$	0.989	$4.142 \cdot 10^{-10}$	0.994	$2.971 \cdot 10^{-11}$
	0.996	0.334	0.998	0.111	0.985	0.077	0.998	$2.084 \cdot 10^{-8}$	0.999	$8.903 \cdot 10^{-10}$
200	0.113	0.043	0.739	$6.034 \cdot 10^{-6}$	0.731	$1.144 \cdot 10^{-9}$	0.622	$4.986 \cdot 10^{-10}$	0.541	$3.226 \cdot 10^{-17}$
	0.733	0.235	0.889	$1.754 \cdot 10^{-3}$	0.888	$1.748 \cdot 10^{-9}$	0.779	$2.305 \cdot 10^{-9}$	0.736	$2.174 \cdot 10^{-15}$
	0.883	0.448	0.939	$5.254 \cdot 10^{-3}$	0.935	$6.084 \cdot 10^{-9}$	0.963	$4.498 \cdot 10^{-9}$	0.903	$2.597 \cdot 10^{-15}$
	0.929	0.525	0.989	0.050	0.988	$1.830 \cdot 10^{-7}$	0.972	$1.009 \cdot 10^{-8}$	0.971	$6.797 \cdot 10^{-10}$
	0.976	0.570	0.993	0.141	0.998	$1.091 \cdot 10^{-3}$	0.989	$5.089 \cdot 10^{-7}$	0.999	$9.940 \cdot 10^{-10}$
300	0.571	$3.944 \cdot 10^{-4}$	0.466	$1.681 \cdot 10^{-9}$	0.659	$7.109 \cdot 10^{-13}$	0.931	$3.879 \cdot 10^{-13}$	0.892	$9.353 \cdot 10^{-24}$
	0.689	0.010	0.515	$2.938 \cdot 10^{-7}$	0.902	$4.558 \cdot 10^{-10}$	0.991	$1.187 \cdot 10^{-12}$	0.971	$3.230 \cdot 10^{-14}$
	0.763	0.048	0.841	$1.079 \cdot 10^{-4}$	0.942	$1.275 \cdot 10^{-8}$	0.992	$6.814 \cdot 10^{-10}$	0.980	$5.531 \cdot 10^{-11}$
	0.782	0.069	0.969	$5.000 \cdot 10^{-4}$	0.982	$5.429 \cdot 10^{-8}$	0.993	$8.887 \cdot 10^{-10}$	0.992	$1.170 \cdot 10^{-10}$
	0.988	0.943	1.000	0.095	0.997	$1.909 \cdot 10^{-7}$	0.999	$8.317 \cdot 10^{-8}$	0.993	$3.519 \cdot 10^{-10}$
400	0.422	0.050	0.016	$6.194 \cdot 10^{-7}$	0.617	$1.095 \cdot 10^{-9}$	0.806	$4.633 \cdot 10^{-18}$	0.581	$4.060 \cdot 10^{-24}$
	0.448	0.133	0.668	$5.822 \cdot 10^{-5}$	0.906	$1.251 \cdot 10^{-9}$	0.969	$3.538 \cdot 10^{-17}$	0.822	$2.770 \cdot 10^{-21}$
	0.620	0.167	0.833	$6.362 \cdot 10^{-4}$	0.935	$1.491 \cdot 10^{-7}$	0.984	$8.080 \cdot 10^{-13}$	0.868	$1.247 \cdot 10^{-19}$
	0.902	0.199	0.919	$7.702 \cdot 10^{-4}$	0.935	$1.364 \cdot 10^{-4}$	0.984	$5.306 \cdot 10^{-12}$	0.978	$2.357 \cdot 10^{-19}$
	0.963	0.572	0.976	$2.161 \cdot 10^{-3}$	0.947	$3.646 \cdot 10^{-4}$	1.000	$1.251 \cdot 10^{-8}$	0.981	$5.321 \cdot 10^{-18}$
500	0.672	$2.199 \cdot 10^{-3}$	0.522	$1.233 \cdot 10^{-4}$	0.312	$2.707 \cdot 10^{-13}$	0.546	$2.451 \cdot 10^{-16}$	0.568	$3.273 \cdot 10^{-26}$
	0.759	0.093	0.702	$1.515 \cdot 10^{-3}$	0.785	$3.588 \cdot 10^{-11}$	0.775	$1.181 \cdot 10^{-15}$	0.760	$1.426 \cdot 10^{-22}$
	0.855	0.137	0.958	$1.687 \cdot 10^{-3}$	0.964	$2.414 \cdot 10^{-9}$	0.925	$3.337 \cdot 10^{-14}$	0.929	$6.458 \cdot 10^{-22}$
	0.963	0.236	0.960	$5.525 \cdot 10^{-3}$	0.998	$1.067 \cdot 10^{-5}$	0.962	$1.775 \cdot 10^{-13}$	0.933	$1.208 \cdot 10^{-18}$
	0.965	0.727	0.981	0.017	0.999	$3.531 \cdot 10^{-4}$	0.986	$2.083 \cdot 10^{-12}$	0.998	$4.781 \cdot 10^{-18}$

$n' \in [1..n]$, where for every benchmark s independent algorithm runs are conducted, and the number of fitness function evaluations needed to reach the optimum is recorded.

Table 4 indicates that this problem is indeed harder than the previous one, but our method is able to find the difference for every value of n and large enough number of samples s. The columns labeled "Diff" present the p-values resulting from comparison of RLS and RLS_k for k set as above, which get stabilized at small values when the number of samples s is at least 400. Small p-values start to appear already at $s = 200$, however, occasionally large p-values are also produced for $s \leq 300$. The fact that p-values may differ in order by as large as 10^{14} for different trials of the same process deserves more attention, which we leave for the future work.

On the contrast, the control columns labeled "Same" indicate that our method is not misled by deriving conclusions from multiple comparisons, some of which occasionally produce extreme statistics, even for up to 500 samples in up to 500 benchmarks. The only observed occasion for such a p-value to be noticeably small (0.016, occurring at $n = 400$, $s = 200$) does not contradict this statement, as it happened only once in 125 trials, which can be filtered out by any post-hoc correction in practice.

Note that Table 4 does not show a noticeable increase of precision with the growth of n. This is explained by a fact that was mentioned earlier, namely, the relative difference between RLS and RLS_k decreases as n increases, so the effective "weights" of those $n' \leq n$ where this difference is noticeable reduce when n grows.

Table 5: The p-values for comparing RLS and RLS_2. In this table, $n = 100$ and $s = 500$.

Same	Diff
0.826	$7.085 \cdot 10^{-7}$
0.835	$1.653 \cdot 10^{-6}$
0.970	$3.761 \cdot 10^{-6}$
0.992	$1.060 \cdot 10^{-4}$
0.996	$6.479 \cdot 10^{-4}$

In fact, we can even spot the difference between RLS and RLS_2, where the initial individual is chosen to be the best out of just two randomly sampled individuals. Table 5 shows the similar report for this case, limited by $n = 100$ and $s = 500$. Note that the p-values from the "Diff" column are much larger than the ones in the matching cell of Table 4, which reflects a smaller difference between RLS and RLS_2 compared to the one between RLS and RLS_k.

5 CONCLUSION

We presented our first step towards statistical tests designed specially for the settings common in evolutionary computation, and

probably for computational intelligence in general. Our method, the *rank-sum result joiner*, runs the existing statistical tests on the results for every separate benchmark, and then joins their results together, assuming all runs of the algorithms on all benchmarks are independent. This is done using dynamic programming without any extra assumptions and approximations. The result is a single p-value for the null hypothesis constructed as an intersection of the null hypotheses of the subordinate tests.

Our first results are promising, suggesting that good replacements for the default statistical assessment procedures, which will be able to produce significantly more precise results from the same input data, are only a few steps ahead at least for some of the possible applications. Our method is rather generic and requires only a moderate change of the commonly used interfaces to the statistical tests, so its adaptation to any other statistical tests than the two-sided Kolmogorov-Smirnov test shall be very straightforward.

In particular, using the outcomes of *different* statistical tests, for different benchmarks, is also supported. Although the users would need to be much more accurate, as running multiple tests on the same data no longer satisfies the independence assumptions, this is still a good feature which potentially increases the applicability of the proposed method.

A few drawbacks of the proposed method are already recognized during the presented experiments:

- A noticeable computation cost. As our current implementation does not perform any approximation, even where theoretically feasible, the running time, apart from running the exact versions of the statistical tests, is an additional $O(\Sigma^2)$ term, where Σ is the sum of numbers of possible statistic values among all subordinate statistical tests. For the number of benchmarks $n = 500$ and the number of samples $s = 500$ for each benchmark, this is order 500^4 with the Kolmogorov-Smirnov test, or several minutes on an ordinary computer. More work towards understanding where the central limit theorems are applicable will of course reduce this to the tolerable values.
- In the presence of different sample sizes at different benchmarks, the current implementation will effectively pay more attention to the results corresponding to larger sample sizes. This seems to require only a little more work, as it is possible to rescale the ranks of statistics in order to have identical effective weights. However, a straightforward implementation of this idea increases the computational complexity.
- The consistency of the produced p-values across multiple trials shall also be further investigated.

The proposed method is currently implemented as a part of a small data analysis software package by the author of this paper[1], and it is used to decide which of the benchmarks to restart during continuous integration in one of his projects[2]. The author thinks that a similar approach might be helpful to monitor the changes in performance of evolutionary algorithms while maintaining benchmarking software, such as COCO[3] or IOHprofiler[4].

Finally, we must note that the evolutionary computation community needs to learn how to set the statistical questions right, and only then to search for tools that can answer these questions — or to build their own tools. Correct formulation of null and alternative hypotheses is a difficult question, and only occasionally these formulations can be straightforward (e.g. runtime distribution equality, or stochastic dominance). The inherent limitation for the statistical outcomes to be sample-based further complicates these questions, as, for instance, it is not easy to formulate a hypothesis about the asymptotically smaller runtime of an algorithm A compared to the algorithm B using only sample-based language.

ACKNOWLEDGMENTS

This research was supported by the Russian Scientific Foundation, agreement No. 17-71-20178.

REFERENCES

[1] 2015. Bayesian statistics. *Nature Methods* 12 (2015), 377–378.
[2] William Jay Conover. 1999. *Practical Nonparametric Statistics* (3rd ed.). Wiley.
[3] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. 2015. Money for Nothing: Speeding Up Evolutionary Algorithms Through Better Initialization. In *Proceedings of Genetic and Evolutionary Computation Conference*. 815–822.
[4] Joaquin Derrac, Salvador Garcia, Daniel Molina, and Francisco Herrera. 2011. A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. *Swarm and Evolutionary Computation* 1, 1 (2011), 3–18.
[5] Benjamin Doerr and Carola Doerr. 2016. The Impact of Random Initialization on the Runtime of Randomized Search Heuristics. *Algorithmica* 75, 3 (2016), 529–553.
[6] Olive Jean Dunn. 1961. Multiple Comparisons Among Means. *J. Amer. Statist. Assoc.* 56, 293 (1961), 52–64.
[7] Milton Friedman. 1940. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics* 11, 1 (1940), 86–92.
[8] Yosef Hochberg. 1988. A Sharper Bonferroni Procedure for Multiple Tests of Significance. *Biometrika* 75, 4 (1988), 800–802.
[9] Myles Hollander, Douglas A. Wolfe, and Eric Chicken. 2007. *Nonparametric Statistical Methods* (3rd ed.). Wiley.
[10] Andrey Kolmogorov. 1933. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari* 4 (1933), 83–91.
[11] William H. Kruskal and W. Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *J. Amer. Statist. Assoc.* 47 (1952), 583–621.
[12] Henry B. Mann and Donald R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics* 18, 1 (1947), 50–60.
[13] R Core Team. 2013. R: A Language and Environment for Statistical Computing. http://www.R-project.org/. http://www.R-project.org/
[14] John A. Rice. 2007. *Mathematical Statistics and Data Analysis* (3rd ed.). Cengage Learning.
[15] Nikolai Smirnov. 1948. Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics* 19, 2 (1948), 279–281.
[16] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin* 1, 6 (1945), 80–83.

[1] Available on GitHub: https://github.com/mbuzdalov/data-slicer.
[2] https://github.com/mbuzdalov/non-dominated-sorting
[3] https://github.com/numbbo/coco
[4] https://github.com/IOHprofiler