# Adaptive Estimation of the Number of Algorithm Runs in Stochastic Optimization

TOME EFTIMOV, Computer Systems Department
Jožef Stefan Institute, Slovenia
PETER KOROŠEC, Computer Systems Department
Jožef Stefan Institute, Slovenia

Determining the number of algorithm runs is a critical aspect of experimental design, as it directly influences the experiment's duration and the reliability of its outcomes. This paper introduces an empirical approach to estimating the required number of runs per problem instance for accurate estimation of the performance of the continuous single-objective stochastic optimization algorithm. The method leverages probability theory, incorporating a robustness check to identify significant imbalances in the data distribution relative to the mean, and dynamically adjusts the number of runs during execution as an online approach.

The proposed methodology was extensively tested across two algorithm portfolios (104 Differential Evolution configurations and the Nevergrad portfolio) and the COCO benchmark suite, totaling 5,748,000 runs. The results demonstrate 82%–95% accuracy in estimations across different algorithms, allowing a reduction of approximately 50% in the number of runs without compromising optimization outcomes. This online calculation of required runs not only improves benchmarking efficiency, but also contributes to energy reduction, fostering a more environmentally sustainable computing ecosystem.

CCS Concepts: • **Computing methodologies** → **Continuous space search**; **Simulation evaluation**; • **Mathematics of computing** → **Probability and statistics**.

Additional Key Words and Phrases: experimental design, number of runs, stochastic optimization algorithms, green benchmarking

## 1 INTRODUCTION

Estimating the performance of the algorithm is a time-consuming process and requires the repetition of numerous algorithm runs on different problem instances. This is especially important for the algorithm's development phase, where we need to establish if the newly developed algorithm is significantly different, hopefully, better than the previous version or some other competing algorithms. This is most commonly achieved through benchmarking [2]. To derive a valid conclusion about algorithm performance, one needs to carefully select problem instances, establish experimental setups, and assess performance in accordance with benchmarking theory. The selection of the number of algorithm runs is one of the most crucial decisions in experiments setup, which influence the time required to perform the experiment and the validity of the conclusions drawn from it. If the number of experimental runs is insufficient, critical characteristics of the algorithm's performance may not be adequately captured. Conversely, an excessive number of runs could lead to inefficient use of resources, resulting in unnecessary time expenditure without a corresponding increase in insight. [28]. So the question that we would like to answer is, is there a way to determine if the number of runs so far performed (i.e., estimated after each run) provides enough information to draw valid conclusions, so no further runs would be needed?

The state-of-the-art performance evaluation is based on reporting descriptive statistics (e.g., mean, median, standard deviation) or comparing performance distributions [12]. Therefore, establishing enough information by performing enough algorithm runs is crucial. Given the stochastic nature of the algorithms, where each run may yield inherently different outcomes, it is theoretically optimal to maximize the number of runs to ensure comprehensive and reliable performance evaluation. Looking at journal papers and competitions held at different conferences, this number can be from 15 [14] to 51 [20], with common values set at 25 or 30. The critical questions to address are: Are 15 runs sufficient to capture the necessary performance characteristics, or is that already unnecessarily high? Alternatively, is 51 excessive, resulting in an inefficient allocation of time and resources?

Ideally, the number of runs should be minimized while still ensuring it is adequate to produce statistically robust and reliable performance evaluations.

In reality, the number of runs does not need to be the same for all problem instances. For problem instances where the algorithm demonstrates consistent behavior, a lower number of runs may suffice to obtain reliable performance estimates. Conversely, for problem instances where the algorithm exhibits highly variable or unpredictable behavior, a greater number of runs is necessary to ensure statistically robust evaluations. However, this issue is not solely dependent on the characteristics of the problem instance but is also influenced by the algorithm's performance on that specific problem instance. Thus, it is possible for one algorithm to exhibit consistent performance on a given problem instance, requiring fewer runs to achieve reliable evaluation, while another algorithm may display highly variable behavior on the same problem instance, necessitating a greater number of runs to obtain statistically robust results.

**Our contribution:** Given the critical importance of selecting an appropriate number of runs for valid and efficient algorithm performance evaluation, this paper proposes an empirical methodology for determining the required number of runs to ensure accurate analysis in continuous single-objective stochastic optimization. The approach uses probability theory, especially the symmetry or balance of the data distribution around its mean.It can be assumed as an online approach that estimates the number of runs during the execution phase of the experiment. This means there is no a priori number of runs set, but the execution of runs is stopped when the information gathered from so far runs is sufficient enough to represent the performance of the algorithm, which, in single-objective problems, is the quality of the best solution. The proposed methodology has been evaluated on a large set of experiments conducted by pairing two different algorithm portfolios (104 Differential Evolution (DE) configurations [25] and 11 Nevergrad algorithms [26]) by using the 24 problems from the Black-Box Optimization Benchmarking (COCO) [14] in different problem dimensions including 10, 20, and 40. The total amount of runs executed is 5,748,000. The experimental results have shown that the approach led to approximated 82% - 95% correct estimations depending on the combination of an algorithm portfolio and a benchmark suite. These results led to a rough estimation of reducing the number of runs conducted in this study and still providing accurate continuous single-objective stochastic optimization analysis. It comes out that approximately 50% of the overall runs conducted in this study can be omitted. Encouraging the online calculation of the number of runs required for accurate analysis for a specific algorithm and problem instance fosters experiments aimed at reducing energy usage and promoting an environmentally conscious computing ecosystem, also known as green benchmarking.

**Outline:** The remaining sections of the paper are structured as follows: Section 2 outlines the related work; Section 3 presents the empirical approach to determining the required number of runs for reliable single-objective continuous optimization; Section 4 provides a detailed explanation of the experimental design, focusing on the algorithm and problem portfolios used in the study; Section 5 presents the evaluation results of the proposed methodology; Section 6 addresses the green benchmarking aspect of the proposed methodology. Lastly, Section 7 concludes the study and suggests directions for future research.

**Data and code availability:** The data and the code used in this study are available at https://zenodo.org/records/15099850.

## 2 RELATED WORK

Estimating the number of runs for stochastic optimization algorithms is a challenging task, as it depends on several factors such as the problem size, the algorithm parameters, the stopping criteria, the desired accuracy, and the statistical significance. A common approach is to use a fixed number of runs for each problem and compare the average results of different algorithms. Recently, a study [30] shows that an insufficient number of runs can impact the effectiveness of automated algorithm configuration methods, leading to the possibility of selecting a suboptimal configuration by chance. The study demonstrates that relying solely on mean performance values, a

common practice among configurators, necessitates a substantial number of runs to obtain reliable comparisons among the configurations being considered.

In most cases, the conventional approach recommends using “standard" values for the number of runs, typically 30 or 50, or even more [8]. From a statistical perspective, increasing the number of runs enhances the likelihood of detecting significant performance differences between the algorithms being compared [3]. Larger sample sizes increase the sensitivity of statistical methods, allowing the detection of even minor differences. However, this heightened sensitivity can lead to the misinterpretation of negligible effects with no practical significance as being statistically significant [1, 11]. It is important to emphasize that statistical analyses conducted with moderately sized or even small samples can yield results that are equally valuable and robust as those derived from very large representative samples [23], provided the experimental design is rigorous, the assumptions underlying the statistical tests are met, and the sample is appropriately representative of the population of interest.

In [5], the authors introduce a methodology for determining the necessary sample sizes (i.e., number of problem instances and number of repeated runs) when designing experiments with specific statistical properties for comparing two methods within a given problem class. The proposed approach enables researchers to specify desired levels of accuracy for estimating mean performance differences on individual problem instances, as well as the desired statistical power for comparing mean performances across the problem class. This approach determines the number of runs ($n_i$, $i = 1, 2$) of two algorithms $A_1$ and $A_2$ on a specific problem instance as the problem of identifying the minimum total sample size, $n_1 + n_2$, required for the standard error of the means difference estimator to be below a predetermined accuracy threshold. This problem is a subject of constraint optimization. It is important to highlight that in this approach, the number of runs is always considered within the context of comparing two or multiple algorithms (the number of runs depends on the combination of which algorithms are paired). Furthermore, the authors present a generalization of the approach for comparing multiple algorithms [6]. However, in our study, our focus is not on comparing algorithms but rather on developing an empirical online approach that can determine when enough data has been collected from running an algorithm on a specific problem instance, neglecting the behavior of other algorithms on the same problem instance.

## 3 ESTIMATING THE NUMBER OF RUNS NEEDED

We propose an empirical method to determine the number of runs required for reliable single-objective continuous optimization. Our goal is to estimate the number of runs, $n$, necessary to gather sufficient representative data from executing an algorithm on a specific problem instance. This ensures that the $n$ runs, $(x_1, x_2, \ldots, x_n)$, provide a robust assessment of the algorithm’s performance.

To estimate this, we begin by executing the algorithm $p$ times on a given problem instance, where $p$ may correspond to a predefined threshold (the selection will be explained further). From the resulting $p$ values, we compute their mean and center each value by subtracting this mean. This operation is presented below:

$$Y_i = X_i - \overline{X}. \tag{1}$$

where $\overline{X}$ is the mean value from the $p$ runs. From probability theory, it follows that by subtracting the mean of $p$ values from the same distribution the resulting variables are not independent Bernoulli variables. The $Y_i$ are not independent because of their shared dependency of $\overline{X}$.

To determine the required number of runs, we conduct a robustness check, by evaluating the symmetry or balance of the data distribution around its mean. For this reason, we use the skewness of the $Y_i$ variables as a non-parametric diagnostic to assess whether the data distribution exhibits significant imbalance relative to the mean. Our hypothesis is that an algorithm gathering sufficient data from the runs should exhibit a symmetric distribution around the mean.

The pipeline for estimating the required number of runs is outlined as follows:

**(1) Initial Runs:** Execute the algorithm instance on a given problem instance five times, $p = 5$, collecting the solution values $(x_1, \ldots, x_5)$. The decision to use five initial runs is arbitrary, following a practice also adopted in other statistical methods unrelated to our case, such as the $\chi^2$ test of goodness of fit [7]. From these solution values, compute the transformations $Y_i$.

**(2) Symmetry Check:** Assess the symmetry of the data. Calculate the skewness, $\tilde{\mu}_3 = \mathbb{E}\left[\left(\frac{Y-\mu}{\sigma}\right)^3\right]$, of the $Y_i$ values. In an ideal symmetric scenario, the skewness should be close to zero. To accommodate minor deviations, a predefined threshold ($\tau$) around zero is set to evaluate whether the distribution can be considered symmetric ($-\tau \leq \tilde{\mu}_3 \leq \tau$).

**(3) Adjust Runs Based on Symmetry:** If the skewness falls outside the predefined threshold, execute an additional run ($p = p + 1$) and repeat all necessary calculations to reassess the distribution's symmetry. If the skewness lies within the predefined threshold, terminate the process. The needed number of runs is $n = p$.

Skewness can be used to assess the symmetry of a distribution; however, in practice, an algorithm instance's performance may be significantly impacted by a few outliers (a few extreme run values not consistent with the majority of runs because of the stochastic nature of the algorithms, that can be either on one or both sides of the distribution) that strongly influence the skewness. To address this, empirical adjustments are necessary, involving the application of techniques to manage outliers, ensuring the effectiveness of the proposed approach.

To apply the proposed approach, an additional preprocessing step is conducted to address the presence of outliers prior to performing the estimation. Given a sample of size $p$, denoted as $(x_1, x_2, \ldots, x_p)$, an outlier detection technique is applied before executing the calculation steps and evaluating the symmetry of the distribution. If the technique identifies $m$ outliers, these observations are excluded, resulting in a refined sample of size $p - m$ used for the symmetry check. Notably, the final estimator for the required number of runs, provided the predefined conditions are met, is still based on the original sample size $p$.

We selected three well-established techniques for handling outliers, which are described in detail below:

**(1) Interquartile range (IQR) method** [31] – It is a measure of the spread of data. In general, the value *IQR* is the difference between the 25th ($q_{0.25}$) and 75th ($q_{0.75}$) percentiles of the data. When it is used for outlier detection, all data values below $q_{0.25} - 1.5 \cdot IQR$ and above $q_{0.75} + 1.5 \cdot IQR$ are considered outliers.

**(2) Percentiles-based method** [9] – In this method, all data values that are outside the 2.5th and the 97.5th percentiles will be detected as outliers.

**(3) Modified z-score** [16] – It is a more robust way to detect outliers than a z-score. The z-score, which provides information on how many standard deviations a value is from the mean value, can be affected by unusually large or small data values. The modified z-score is calculated as $modified - z - score = \frac{0.6745(x_i - \tilde{x})}{MAD}$, where $x_i$ is a single data value, $\tilde{x}$ is the median of the data, and *MAD* is the median absolute deviation of the data. If the $MAD = 0$, then we use the mean of the absolute deviation of the data instead of *MAD* for division. The values with modified z-scores that are less than -3.5 or greater than 3.5 are detected as outliers.

## 4 EXPERIMENTAL DESIGN

Here, we outline the experimental design used to collect data for evaluating the proposed methodology, detailing the benchmark suite and optimization algorithms included in the study.

**Problem portfolio:** We utilized the COmparing Continuous Optimizers (COCO) [13] problem suite, which comprises 24 single-objective optimization problem classes. For each class, an arbitrary number of instances can be generated, with instances representing shifted, scaled, and/or rotated variants of the same problem. From the suite's available problem dimensions, we selected $D \in \{10, 20, 40\}$.

**Algorithm portfolio:** For optimization algorithms, the primary evaluation was conducted using the standard Differential Evolution (DE) algorithm [25] on the COCO benchmark suite, using 15 problem instances. To

generate diverse data reflecting varying algorithm behaviors, 104 DE instances were created by randomly varying hyperparameters: mutation strategy, scaling factor, and crossover probability. Mutation strategies were selected from the following pool: Rand/1/Bin, Rand/1/Exp, Rand/2/Bin, Rand/2/Exp, Best/1/Bin, Best/1/Exp, Best/2/Bin, Best/2/Exp, Best/3/Bin, Rand
/Rand/Bin, RandToBest/1/Bin, and RandToBest/1/Exp. The scaling factor $F$ and crossover probability $Cr$ were chosen from the interval $(0, 1)$. The population size was set equal to the problem dimension, with stopping criteria defined as one of the following: $D \times 10,000$ function evaluations, 100 iterations without improvement, or achieving an optimum solution within $10^{-8}$ of the true optimal value.

To mitigate potential bias from focusing on a single family of optimization algorithms, additional experiments were conducted using 11 algorithms from Nevergrad [26]. The selected algorithms include Differential Evolution (DE) [27], Diagonal CMA [15], NaivelsoEMNA [19], NGOpt14, NGOpt38 (two versions of Nevergrad's built-in algorithm selection wizard [22]), OnePlusOne [4], modCMA [15], modDE [29], PSO [17], Random Search, and RCobyla [24]. All algorithms were run with default hyperparameter values, with stopping criteria set to $D \times 2,000$ function evaluations. This stopping criteria differs from our DE experiments because the data was reused from a publicly available study [18]. The data contains the performance of the Nevergrad algorithms using 10 instances per problem for a single dimension, $D = 20$.

**Performance data:** Each algorithm configuration was executed 50 times for each problem instance and corresponding dimension. For the DE configurations and the COCO benchmark suite, we utilized 1,080 unique triplets (24 problems × 15 instances × 3 problem dimensions). Each triplet was executed 50 times with 104 DE configurations, resulting in a total of 5,616,000 runs. For the Nevergrad experiments, 240 triplets (24 problems × 10 instances × 1 problem dimension) were tested 50 times with 11 algorithms, ending up in 132,000 runs. Altogether, this experimental setup produced 5,748,000 algorithm runs, generating optimization results (errors to the global optimum) used to evaluate the proposed methodology.

**Evaluation scenarios:** To evaluate the accuracy of estimating the required number of runs, we generated 50 independent solution values $(x_1, x_2, \ldots, x_{50})$ for each combination of a triplet (problem, instance, and dimension) and an algorithm instance, forming a ground truth sample. Using this sample, we sequentially analyzed subsets starting with the first five solutions to determine the required number of runs, $n$. The mean difference between the $n$-sized sample and the ground truth sample mean was computed to estimate the sample size effect. To assess precision and confidence in this estimation, we calculated the bootstrap confidence interval (CI) of the mean difference.

**Bootstrapping evaluation scenario**: Since outlier detection techniques are part of our estimation approach, the evaluation begins by applying an outlier detection method to the estimated $n$-sized sample $(x_1, x_2, \ldots, x_n)$. Let $m_e$ be the number of detected outliers, resulting in a sample of size $n - m_e$. Similarly, the same outlier detection technique is applied to the 50 ground truth samples, yielding a size of $50 - m_t$ after removing $m_t$ outliers. Next, $M$ resamples of size 50 are drawn from both the $(n - m_e)$ and $(50 - m_t)$ samples, and their mean differences are computed. Using these $M$ mean differences, the 2.5th and 97.5th percentiles are calculated to construct a 95% percentile confidence interval (CI). While the bootstrapped mean difference typically follows a normal distribution by the Central Limit Theorem [10], skewed distributions may occur. In such cases, the bias-corrected and accelerated bootstrap [9] is used to account for skewness. By examining whether the CI contains zero, we determine the percentage of cases where the number of runs estimation leads to accurate optimization analysis. For cases where the analysis is inaccurate, the bias-corrected and accelerated bootstrap is further checked to handle skewness and verify if the CI contains zero.

**Post-hoc empirical evaluation:** If neither the original confidence interval nor the bias-corrected and accelerated confidence interval includes a zero mean difference, we calculate an empirical measure to assess how often the bootstrapped mean value from the estimated sample falls within a specified percentage of the bootstrapped mean

value from the ground truth sample. The following percentage thresholds are used: 0.50%, 1.00%, 5.00%, 10.00%, 15.00%, and 20.00%.

## 5 RESULTS

**Comprehensive Benchmarking of DE Configurations Across Problem Dimensions:** Table 1 shows the percentage of triplets (problem, instance, and dimension) where the proposed method accurately estimates the number of runs. The "True" column represents percentages based on bootstrapping CIs (original and bias-corrected), while the other six columns reflect post-hoc evaluation within specified error ranges. The results are based on 104 DE configurations tested on the COCO benchmark suite (24 problem classes, 15 instances per class, and three dimensions: $D \in \{10, 20, 40\}$). The evaluation, based on bootstrapping CIs requiring stochastic sampling, was repeated 10 times for each triplet and configuration using different sampling seeds. The percentages for each configuration were averaged across these repetitions, and the final values were averaged over all configurations, as shown in Table 1.

Table 1. The percentage of triplets where the proposed method accurately estimates the number of runs. The "True" column shows results from bootstrapping CIs, while the next six columns indicate cases within specified error margins (as labeled).

| Skewness threshold | Outlier technique | true | ≤0.5% | ≤1% | ≤5% | ≤10% | ≤15% | ≤20% |
|---|---|---|---|---|---|---|---|---|
| 0.05 | IQR | 84.13 | 84.17 | 84.22 | 86.12 | 88.73 | 90.85 | 92.66 |
| | Percentile | 79.91 | 79.93 | 79.97 | 82.47 | 85.86 | 88.44 | 90.48 |
| | MAD | 86.16 | 86.20 | 86.24 | 87.66 | 89.77 | 91.48 | 93.00 |
| 0.10 | IQR | 77.21 | 77.26 | 77.33 | 79.94 | 83.62 | 86.62 | 89.15 |
| | Percentile | 72.36 | 72.38 | 72.44 | 75.79 | 80.38 | 83.96 | 86.80 |
| | MAD | 79.72 | 79.78 | 79.83 | 81.86 | 84.89 | 87.34 | 89.46 |
| 0.15 | IQR | 72.91 | 72.97 | 73.04 | 76.09 | 80.41 | 83.90 | 86.88 |
| | Percentile | 67.86 | 67.89 | 67.97 | 71.81 | 77.11 | 81.30 | 84.57 |
| | MAD | 75.68 | 75.75 | 75.81 | 78.23 | 81.78 | 84.69 | 87.21 |
| 0.20 | IQR | 69.50 | 69.57 | 69.64 | 73.01 | 77.74 | 81.64 | 84.95 |
| | Percentile | 64.63 | 64.66 | 64.74 | 68.94 | 74.66 | 79.26 | 82.77 |
| | MAD | 72.37 | 72.44 | 72.52 | 75.21 | 79.15 | 82.45 | 85.26 |

The approach was evaluated for four skewness thresholds (0.05, 0.10, 0.15, and 0.20) using three outlier detection techniques. Among the 104 DE configurations, the *MAD* technique achieved the highest accuracy (86.16%), correctly estimating 933 out of 1,080 triplets based on bootstrap CIs at a skewness threshold of 0.05. The *IQR* and *Percentile* techniques yielded similar results with slight reductions in accuracy. For triplets where the correct number of runs was not estimated via bootstrap CIs, post-hoc evaluation was applied. With the *MAD* technique at a 0.05 skewness threshold, cumulative accuracy increased to 87.66% within a 5% error of the ground truth and 89.77% within a 10% error. The results show that increasing the skewness threshold, which allows greater flexibility and deviations from distribution symmetry, leads to a decrease in estimation accuracy as anticipated. However, accuracy remains high ($\geq$ 74.00%) when considering solutions within a 10% error margin of the ground truth sample mean.

Figure 1 provides additional insights into the percentages of correctly estimated runs across all 104 DE configurations for skewness thresholds of 0.05 (top row) and 0.15 (bottom row). Most configurations show percentages of correctly estimated triplets near the average reported in Table 1. For triplets with incorrect run estimations, post-hoc evaluation reveals that as the error range increases (e.g., $\leq$ 5%, $\leq$ 10%, $\leq$ 20%), the percentage of triplets shifts to higher values across all configurations. The patterns are similar also for the other skewness thresholds and are not presented due to the page limit. Similar patterns are observed for other skewness thresholds but are omitted here due to space constraints.

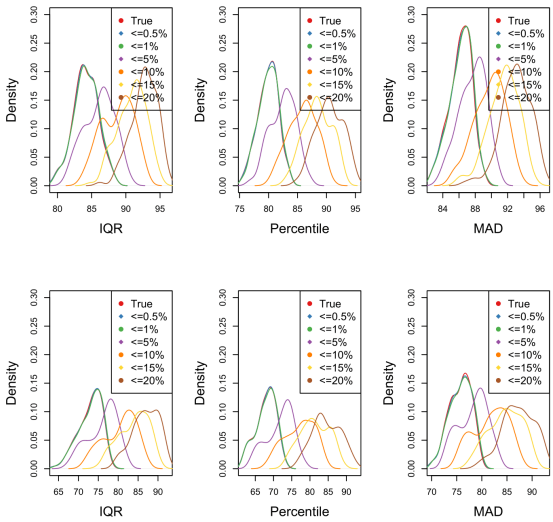


Fig. 1. The distribution of correctly estimated run percentages across all 104 configurations is presented for a skewness threshold of 0.05 (top row) and 0.15 (bottom row). The columns correspond to different outlier detection techniques (*IQR*, *Percentile*, and *MAD*).

To go into more detail, Figure 2 provides a detailed view with error bars representing the CIs for the percentage of triplets where the number of runs was not appropriately estimated, shown for each problem dimension. The percentages were first averaged across 10 sampling seeds for each DE configuration, and the error bars were then computed across 104 DE configurations. The top row shows results for a skewness threshold of 0.05, while the bottom row shows results for 0.15. Columns represent different outlier detection techniques: *IQR"*, *Percentile"*, and *"MAD"*.

From the figure, we observe that, based on bootstrapping CIs or error ranges of $\leq 0.5\%$ and $\leq 1\%$ (i.e., post-hoc analysis), the percentage of triplets where the number of runs is not correctly estimated remains consistent across dimensions, ranging from 5% to 7% depending on the outlier detection method (in case of the skewness threshold set to 0.05). Larger error ranges ($\leq 5\%$, $\leq 10\%$, $\leq 15\%$, $\leq 20\%$) favor higher dimensions, as the percentage of incorrectly estimated triplets decreases with increased flexibility in error range. This occurs because, in higher dimensions, the collected results typically exhibit greater variance, leading to more widely dispersed solutions. When bootstrapping from such distributions, the mean value tends to fall within a specified range around the true mean. In contrast, in lower dimensions, solutions are more tightly clustered, and additional averaging can shift the bootstrap mean. Increasing the skewness threshold from 0.05 to 0.15 raises the percentages but maintains similar patterns, which also hold true for the omitted skewness thresholds.

Next, for each problem and dimension, the percentage of triplets where the estimated number of runs fails to provide accurate results is calculated across 104 DE configurations. This percentage is determined by averaging the cases where the estimation is incorrect over 10 sampling seeds, separately for each DE configuration. For the COCO benchmark suite, the results are first averaged at the problem level (treating all instances of a problem as a single entity) before averaging across sampling seeds. The final results, averaged across all DE configurations for each problem and dimension, are shown in Figure 3.

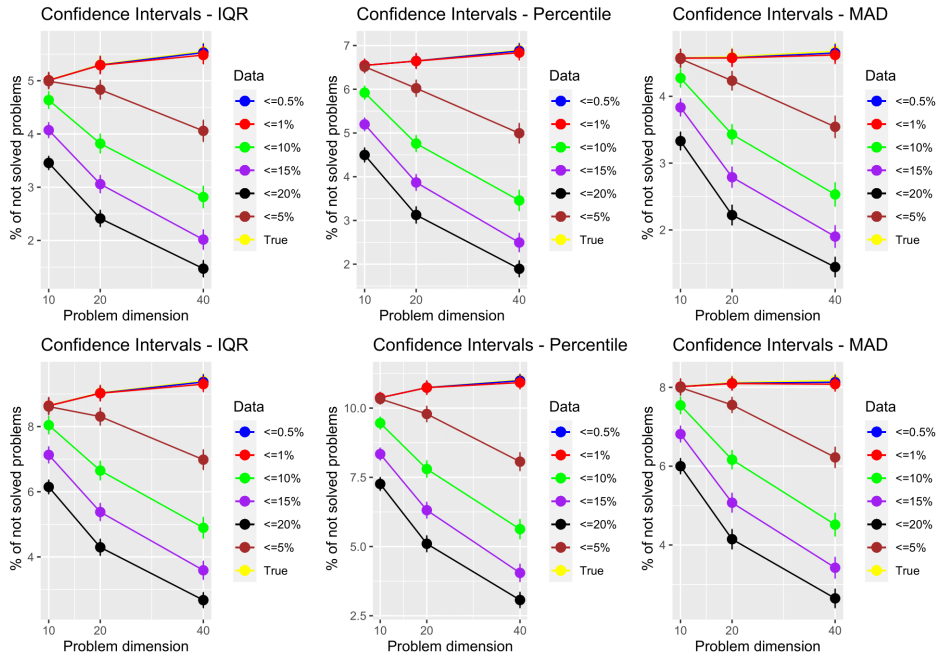


Fig. 2. The error bars represent confidence intervals for the percentage of problems per dimension where the estimated runs do not produce accurate results, calculated across 104 DE configurations. The top row shows results for a skewness threshold of 0.05, the bottom row for 0.15, with columns for outlier detection techniques (*IQR*, *Percentile*, and *MAD*).

The top row corresponds to a skewness threshold of 0.05, and the bottom row to 0.15, with columns representing outlier detection techniques (*IQR"*, *Percentile"*, and *"MAD"*). Each heatmap cell indicates the probability of failing to appropriately estimate the number of runs for a given problem and dimension. Lower percentages suggest a higher likelihood of accurate estimation.

The figure reveals that all problems across dimensions are represented in cases where the estimation fails, though functions like the sphere (problem 1) and linear slope (problem 5) in the COCO suite are easier to estimate accurately. By increasing the skewness threshold, the percentages increase which is an expected pattern. It is important to note that these results aggregate the varying behaviors of different DE configurations.

**Benchmarking the Nevergrad Algorithm Portfolio Across 20*d* Problems:** Figure 4 shows the percentage of triplets for which the proposed method accurately estimates the required number of runs for the Nevergrad portfolio, using the *"MAD"* outlier detection method (similar results are achived by the other methods). The results are grouped by 11 Nevergard algorithms and presented for skewness thresholds of 0.05, 0.10, 0.15, and 0.20. Focusing on the skewness threshold set to 0.05, the results indicate that the proposed method achieves accurate estimations, with percentages ranging from 82.58% (RCobyla, 198/240 triplets correctly estimated) to around 95.16% for modCMA, 91.12% for diagonal CMA, 92.10% for NGOpt14 and 92.08% for NGOpt38. Post-hoc evaluation for RCobyla further improved accuracy to 85.54% within a 10% error to the ground truth mean. In all other algorithms, the post-hoc evaluation improves the results to approximately 90.00% or even higher percentages in finding a solution within an error of 10% to the mean of the ground truth sample. By increasing the skewness threshold, allowing more flexibility and deviations of the distribution symmetry, the estimation accuracy decreases, as expected. Nonetheless, accuracy remains high $\geq$ 80.00%) when focusing on solutions

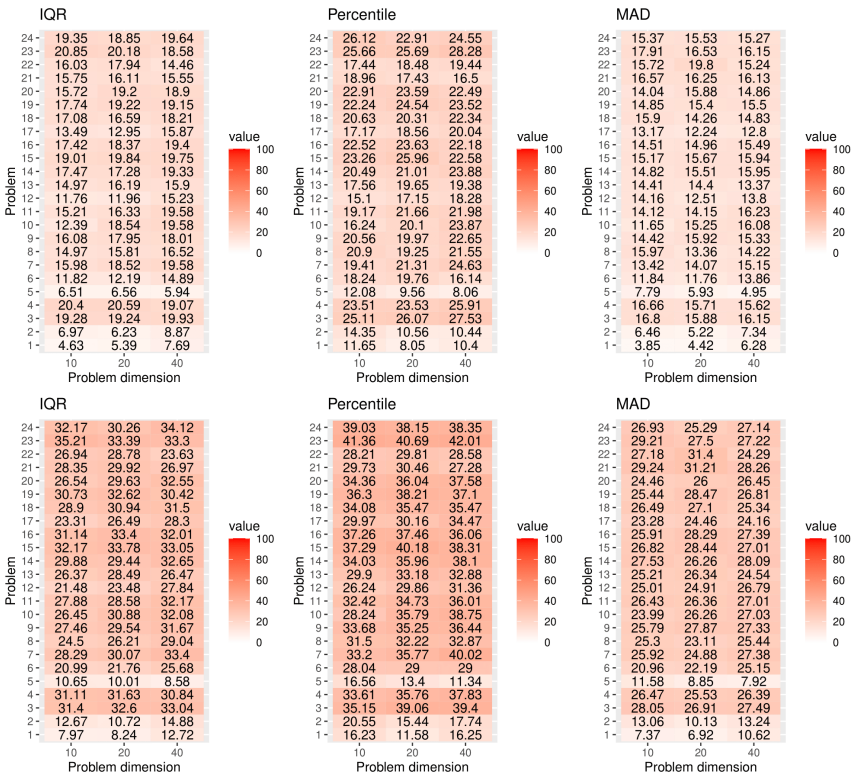


Fig. 3. The percentage of triplets with inaccurate estimations across 104 DE configurations is presented by problem and dimension. The top row shows results for a skewness threshold of 0.05, the bottom row for 0.15, with columns representing outlier detection techniques (*IQR*, *Percentile*, and *MAD*).

within a 10% error margin from the ground truth sample mean. These results are shown for the *"MAD"* outlier detection method, with similar findings for other methods not presented due to the page limits.

## 6 GREEN BENCHMARKING

Online estimation of the required number of runs for a given algorithm and problem instance supports energy-efficient experimentation, aligning with the principles of green benchmarking. This approach minimizes unnecessary computations, contributing to a more sustainable and environmentally friendly computing ecosystem. By enabling algorithm designers to focus on energy-efficient solutions, it fosters advancements toward a greener future in computing. In our study, we estimated the potential reduction in runs for each outlier detection technique across combinations of benchmark suites and algorithm portfolios. The results, summarized in Table 2, include metrics such as total runs (executed 50 times per triplet of problem, instance, and dimension), estimated runs, and saved runs, representing the difference between total and estimated runs. Saved runs highlight unnecessary computations that could be eliminated without affecting results, expressed as a percentage of total runs. To account for inaccuracies in estimation, we also report expected saved runs, calculated as the percentage of accurate estimations applied to saved runs. The percentage of accurate estimations is calculated as an aggregate across all algorithms in the analyzed portfolio. The top row of the table corresponds to the results obtained for the

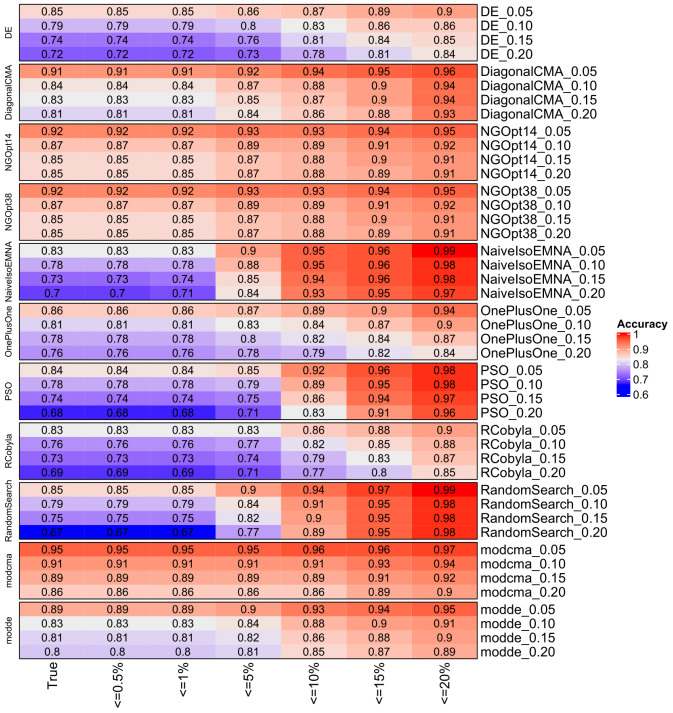


Fig. 4. The heatmap shows the percentage of triplets where the proposed method correctly estimates the number of runs, using bootstrapped CIs and post-hoc evaluations. It is divided into 11 groups, each representing a different Nevergard algorithm, with results presented for skewness thresholds of 0.05, 0.10, 0.15, and 0.20.

104 DE configurations, while the bottom row of the table corresponds to the results obtained for the Nevergrad portfolio.

From the table, we observe that increasing the skewness threshold reduces the number of required runs (i.e., saving a lot of runs) but at the cost of lower estimation accuracy. For instance, with a skewness threshold of 0.05 and the Nevergrad portfolio, the estimated number of required runs is highest across all outlier techniques. This indicates that achieving a closer approximation to a symmetric distribution demands more runs, resulting in greater accuracy. In this case, 43.10% of the runs are expected to be unnecessary. Among the outlier techniques, all produce similar results; however, *"MAD"* stands out as the most rigorous, estimating slightly more runs than the other methods while delivering the highest estimation accuracy. The same patterns are also visible for the DE configurations experiment.

This estimation is based exclusively on the evaluation of bootstrapping CIs. If post-hoc evaluation or identifying solutions within a predefined error range is considered, the percentage of estimated saved runs is likely to increase.

It is also important to highlight that this analysis provides a preliminary approximation. A more detailed investigation of individual problems and dimensions is needed for future research and deeper insights.

## 7 CONCLUSION

Determining the optimal number of algorithm runs is critical in experimental setups, influencing both efficiency and result reliability. This paper presents a novel methodology for estimating the required runs in continuous single-objective stochastic optimization. Using probability theory and a robustness check to detect significant

Table 2. Computational analysis.

| Algorithm portfolio | Skewness threshold | Outlier technique | Total runs | Estimated runs | % of estimated runs | Saved runs | % of saved runs | % of accurate estimation | Expected saved runs | % of expected saved runs |
|---|---|---|---|---|---|---|---|---|---|---|
| DE configurations | 0.05 | IQR | 5616000 | 2939131 | 52.33 | 2676869 | 47.67 | 84.13 | 2252087.64 | 40.10 |
| | | Percentile | 5616000 | 2936222 | 52.28 | 2679778 | 47.72 | 79.91 | 2141341.12 | 38.13 |
| | | MAD | 5616000 | 3184780 | 56.71 | 2431220 | 43.29 | 86.16 | 2094654.47 | 37.30 |
| | 0.10 | IQR | 5616000 | 2325231 | 41.40 | 3290769 | 58.60 | 77.21 | 2540911.97 | 45.24 |
| | | Percentile | 5616000 | 2335983 | 41.60 | 3280017 | 58.40 | 72.36 | 2373337.02 | 42.26 |
| | | MAD | 5616000 | 2567520 | 45.72 | 3048480 | 54.28 | 79.72 | 2430286.14 | 43.27 |
| | 0.15 | IQR | 5616000 | 1988204 | 35.40 | 3627796 | 64.60 | 72.91 | 2645093.50 | 47.10 |
| | | Percentile | 5616000 | 2010232 | 35.79 | 3605768 | 64.21 | 67.86 | 2447049.06 | 43.57 |
| | | MAD | 5616000 | 2204761 | 39.26 | 3411239 | 60.74 | 75.68 | 2581738.78 | 45.97 |
| | 0.20 | IQR | 5616000 | 1731759 | 30.84 | 3884241 | 69.16 | 69.50 | 2699723.86 | 48.07 |
| | | Percentile | 5616000 | 1777237 | 31.65 | 3838763 | 68.35 | 64.63 | 2481128.69 | 44.18 |
| | | MAD | 5616000 | 1918048 | 34.15 | 3697952 | 65.85 | 72.37 | 2676172.17 | 47.65 |
| Nevergrad portfolio | 0.05 | IQR | 132000 | 59683 | 45.21 | 72317 | 54.79 | 85.14 | 61570.69 | 46.64 |
| | | Percentile | 132000 | 59372 | 44.98 | 72628 | 55.02 | 81.05 | 58864.99 | 44.59 |
| | | MAD | 132000 | 67217 | 50.92 | 64783 | 49.08 | 87.81 | 56885.95 | 43.10 |
| | 0.10 | IQR | 132000 | 46105 | 34.93 | 85895 | 65.07 | 78.59 | 67504.88 | 51.14 |
| | | Percentile | 132000 | 46985 | 35.59 | 85015 | 64.41 | 74.26 | 63132.13 | 47.83 |
| | | MAD | 132000 | 53421 | 40.47 | 78579 | 59.53 | 82.19 | 64584.08 | 48.93 |
| | 0.15 | IQR | 132000 | 39320 | 29.79 | 92680 | 70.21 | 75.70 | 70158.76 | 53.15 |
| | | Percentile | 132000 | 39764 | 30.12 | 92236 | 69.88 | 70.85 | 65349.20 | 49.51 |
| | | MAD | 132000 | 46149 | 34.96 | 85851 | 65.04 | 79.15 | 67951.06 | 51.48 |
| | 0.20 | IQR | 132000 | 34790 | 26.36 | 97210 | 73.64 | 73.11 | 71070.23 | 53.84 |
| | | Percentile | 132000 | 35084 | 26.58 | 96916 | 73.42 | 67.57 | 65486.14 | 49.61 |
| | | MAD | 132000 | 40915 | 31.00 | 91085 | 69.00 | 76.28 | 69479.63 | 52.64 |

imbalances in the data distribution relative to the mean, the method dynamically adjusts the number of runs during execution. Evaluated across two algorithm portfolios and one benchmark suite with 5,748,000 runs, the methodology achieved an estimation accuracy of 82%–95%, reducing the number of runs by approximately 50% without compromising optimization quality. This approach enhances efficiency and has the potential to support sustainable computing by minimizing energy consumption through online optimization.

A limitation of the proposed methodology is its occasional inaccuracy in estimating the required number of runs, with errors ranging from 5% to 25%, depending on the combination of an algorithm portfolio and a benchmark suite. Currently, these inaccuracies can only be identified after the evaluation is complete. However, our study has allowed us to annotate such cases. In future work, we plan to use these annotated cases to extract statistical features from the estimated samples, enriched with problem landscape characteristics [21], and algorithm hyperparameters. This enriched data will be used to train a supervised machine learning (ML) classifier capable of determining whether an estimation is accurate, along with a confidence probability. This advancement will address the limitation, enabling real-time implementation of the methodology in an online setting.

## ACKNOWLEDGMENTS

We acknowledge the support of the Slovenian Research and Innovation Agency through program grant P2-0098, and project grants No.J2-4460 and No. GC-0001. This work is also funded by the European Union under Grant Agreement 101187010 (HE ERA Chair AutoLearn-SI). We would like to thank Diederick Vermetten from the Leiden Institute of Advanced Computer Science (LIACS) in the Netherlands for providing us with the Nevergrad performance data.

## REFERENCES


[1] Thomas Bartz-Beielstein. 2006. The new experimentalism. *Experimental Research in Evolutionary Computation: The New Experimentalism* (2006), 13–39.

[2] Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel Lopez-Ibanez, et al. 2020. Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488* (2020).

[3] Thomas Bartz-Beielstein and Mike Preuss. 2007. Experimental research in evolutionary computation. In *Proceedings of the 9th annual conference companion on genetic and evolutionary computation*. 3001–3020.

[4] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies – A comprehensive introduction. *Natural computing* 1 (2002), 3–52. https://doi.org/10.1023/A:1015059928466

[5] Felipe Campelo and Fernanda Takahashi. 2019. Sample size estimation for power and accuracy in the experimental comparison of algorithms. *Journal of Heuristics* 25 (2019), 305–338.

[6] Felipe Campelo and Elizabeth F Wanner. 2020. Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances. *Journal of Heuristics* 26, 6 (2020), 851–883.

[7] William G Cochran. 1952. The $\chi 2$ test of goodness of fit. *The Annals of mathematical statistics* (1952), 315–345.

[8] Joaquín Derrac, Salvador García, 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.

[9] Thomas J Diciccio and Joseph P Romano. 1988. A review of bootstrap confidence intervals. *Journal of the Royal Statistical Society: Series B (Methodological)* 50, 3 (1988), 338–354.

[10] Richard M Dudley. 1978. Central limit theorems for empirical measures. *The Annals of Probability* (1978), 899–929.

[11] Tome Eftimov and Peter Korošec. 2019. Identifying practical significance through statistical comparison of meta-heuristic stochastic optimization algorithms. *Applied Soft Computing* 85 (2019), 105862.

[12] Tome Eftimov, Peter Korošec, and Barbara Koroušić Seljak. 2017. A Novel Approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics. *Information Sciences* 417 (2017), 186–215.

[13] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. 2021. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* 36 (2021), 114–144. Issue 1. https://doi.org/10.1080/10556788.2020.1808977

[14] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions*. Research Report RR-6829. INRIA. https://hal.inria.fr/inria-00362633

[15] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195. https://doi.org/10.1162/106365601750190398

[16] K Senthamarai Kannan, K Manoj, and S Arumugam. 2015. Labeling methods for identifying outliers. *International Journal of Statistics and Systems* 10, 2 (2015), 231–238.

[17] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of International Conference on Neural Networks (ICNN'95), Perth, WA, Australia, November 27 - December 1, 1995*. IEEE, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

[18] Ana Kostovska, Anja Jankovci, Diederick Vermetten, Sašo Džeroski, Tome Eftimov, and Carola Doerr. 2023. Comparing Algorithm Selection Approaches on Black-Box Optimization Problems. In *Proceedings of the 2023 Genetic and Evolutionary Computation Conference Companion*. In Press.

[19] Pedro Larrañaga and José Antonio Lozano (Eds.). 2002. *Estimation of Distribution Algorithms*. Springer. https://doi.org/10.1007/978-1-4615-1539-5

[20] Jing J Liang, Bo Y Qu, and Ponnuthurai N Suganthan. 2013. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Technical report Zhengzhou, China* 635 (2013), 490.

[21] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 829–836.

[22] Laurent Meunier, Herilalaina Rakotoarison, Pak Kan Wong, Baptiste Roziere, Jeremy Rapin, Olivier Teytaud, Antoine Moreau, and Carola Doerr. 2022. Black-box optimization revisited: Improving algorithm selection wizards through massive benchmarking. *IEEE Transactions on Evolutionary Computation* 26, 3 (2022), 490–500. https://doi.org/10.1109/TEVC.2021.3108185

[23] Andrew Ng. 2022. Unbiggen ai. *IEEE Spectrum* 9 (2022).

[24] Michael JD Powell. 1994. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer.

[25] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. 2006. *Differential Evolution: A Practical Approach to Global Optimization*. Springer Science & Business Media.

[26] J. Rapin and O. Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

[27] Rainer Storn and Kenneth Price. 1997. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341.
[28] Niki Veček, Matej Črepinšek, and Marjan Mernik. 2017. On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms. *Applied Soft Computing* 54 (2017), 23–45.
[29] Diederick Vermetten, Fabio Caraffini, Anna V Kononova, and Thomas Bäck. 2023. Modular differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 864–872.
[30] Diederick Vermetten, Hao Wang, Manuel López-Ibañez, Carola Doerr, and Thomas Bäck. 2022. Analyzing the impact of undersampling on the benchmarking and configuration of evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 867–875.
[31] Dewey Lonzo Whaley III. 2005. *The interquartile range: Theory and estimation.* Ph. D. Dissertation. East Tennessee State University.