Searching for the Ground Truth: Assessing the Similarity of Benchmarking Runs

André Bauer
University of Chicago
Chicago, United States
andrebauer@uchicago.edu

Martin Straesser
University of Würzburg
Würzburg, Germany
martin.straesser@uni-wuerzburg.de

Mark Leznik
Ulm University
Ulm, Germany
mark.leznik@uni-ulm.de

Lukas Beierlieb
University of Würzburg
Würzburg, Germany
lukas.beierlieb@uni-wuerzburg.de

Marius Hadry
University of Würzburg
Würzburg, Germany
marius.hadry@uni-wuerzburg.de

Nathaniel Hudson
University of Chicago
Chicago, United States
hudsonn@uchicago.edu

Kyle Chard
University of Chicago
Chicago, United States
chard@uchicago.edu

Samuel Kounev
University of Würzburg
Würzburg, Germany
samuel.kounev@uni-wuerzburg.de

Ian Foster
University of Chicago
Chicago, United States
foster@cs.uchicago.edu

ABSTRACT

Stable and repeatable measurements are essential for comparing the performance of different systems or applications, and benchmarks are used to ensure accuracy and replication. However, if the corresponding measurements are not stable and repeatable, wrong conclusions can be drawn. To facilitate the task of determining whether the measurements are similar, we used a data set of 586 micro-benchmarks to (i) analyze the data set itself, (ii) examine our previous approach, and (iii) propose and evaluate a heuristic. To evaluate the different approaches, we perform a peer review to assess the dissimilarity of the benchmark runs. Our results show that this task is challenging even for humans and that our heuristic exhibits a sensitivity of 92%.

CCS CONCEPTS

• **General and reference → Measurement**; **Metrics**; • **Software and its engineering → Software performance**; • **Information systems → Similarity measures**.

KEYWORDS

ICPE data challenge, time series, similarity, distance measures

ACM Reference Format:
André Bauer, Martin Straesser, Mark Leznik, Lukas Beierlieb, Marius Hadry, Nathaniel Hudson, Kyle Chard, Samuel Kounev, and Ian Foster. 2023. Searching for the Ground Truth: Assessing the Similarity of Benchmarking Runs: This is the author version of this work. Original publication available at: https://doi.org/10.1145/3578245.3584693. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3578245.3584693

1 INTRODUCTION

To compare the performance of different systems or applications, it is essential to have consistent and repeatable measurements from one test to the next. Companies and researchers apply benchmarks for collecting and analyzing data to ensure that their findings are accurate and can be replicated by others. Hence, according to their definition [9], benchmarks promise accurate, reliable, and fair comparisons between different systems or applications.

To further increase the reliability of benchmarks, measurement runs are performed multiple times. Ideally, all of these measurement runs should be similar to each other. However, suppose the measurements are not stable and repeatable. Then, it is difficult to determine whether any observed differences between systems or applications are due to actual differences in their performance or simply due to variability in the measurement process itself or any external factors. This can lead to incorrect conclusions.

Consequently, assessing the similarity of experiment runs is a mandatory task. To understand the performance behavior of a particular system or process, expert knowledge is required. However, it is not always feasible to consult an expert, for instance, to compare many different applications at once, as performed in the underlying data set [8] used in this work. So, the interpretation of the results is the responsibility of non-experts and may be error-prone. To

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0072-9/23/04…$15.00
https://doi.org/10.1145/3578245.3584693

accommodate this, often mean and standard deviation are used for the interpretation [6].

By stating this, the aim of this paper is threefold: (i) We examine the underlying data set [8] by assessing the similarity of performance measurements of 586 micro-benchmarks from 30 popular Java open source projects; (ii) We use the data set to investigate our former approach, which we henceforth refer to as SAME, for detecting dissimilar measurement runs [6], which was only applied to three data sets; (iii) We introduce a heuristic for detecting dissimilar measurement runs based on five different dissimilarity measures[1]. To investigate SAME and the proposed heuristic, we performed a peer review on whether each benchmark run is similar. The reviewers had an average agreement of only 55% in the assessment, demonstrating that assessing the similarity of a benchmark run is challenging. We only considered the benchmark runs where the reviewers agreed to evaluate SAME and the proposed heuristic. On this subset, SAME exhibits a specificity of 83% while our heuristic yields a sensitivity of 92%. Simply put, SAME can detect 83% of the similar benchmark runs, whereas the proposed heuristic can find 92% of the dissimilar benchmark runs. In summary, SAME and our heuristics give good results for the severity of the task. Therefore, they can be used to pre-select whether a benchmark run is similar so that an expert can finally evaluate them.

The remainder of this paper is structured as follows: In Section 2, we introduce the data set, the statistical measurement we apply, SAME, and related work. In Section 3, we explain the proposed heuristic and investigate the data set. In Section 4, we discuss the evaluation of the peer review, SAME, and the heuristic. In Section 5, we conclude the paper.

2 BACKGROUND

2.1 Data Set

Traini et al. [8] provided the data set[2] used in this paper. It includes a large set of performance measurements for 586 microbenchmarks from 30 popular Java open-source projects. These microbenchmarks were executed using the Java Microbenchmark Harness (JMH) framework in a controlled environment, and the results include 3000 measurement batches per benchmark, repeated in 10 runs. This results in over 9 billion benchmark invocations. The data set also contains the Git revisions at which the benchmarks were executed. In this article, we consider only the benchmark runs.

2.2 Fleiss' Kappa Analysis

To decide whether all measurement runs of a benchmark are similar, we performed a peer-review analysis (see Section 4). As humans do this assessment, there may be conflicting reviews. To this end, we apply the statistical measurement Fleiss' Kappa analysis [1, 3] computing a value κ to quantify the level of agreement between reviewers and factor out agreement due to chance. Based on [5], $\kappa < 0$ is equivalent to a low agreement, $\kappa \in [0.01, 0.2]$ is equivalent to a slight agreement, $\kappa \in (0.2, 0.4]$ is equivalent to a fair agreement, $\kappa \in (0.4, 0.6]$ is equivalent to a moderate match, $\kappa \in (0.6, 0.8]$ is equivalent to a substantial match, and finally $\kappa \in (0.8, 1]$ is equivalent to a almost perfect match.

2.3 SAME

In a previous work [6], we proposed an automated approach to detect dissimilar measurement runs, which we refer to as SAME. The key idea is to calculate four (dis)similar measures for each pair of runs, resulting in four $n \times n$ matrices with n being the number of measurement runs, each entry describing the similarity between two measurements. The calculated measures are Cosine similarity, Wave-Hedgets distance, Kumar-Hassebrook distance, and scaled root mean squared error. Then, these four matrices are averaged into one $n \times n$ matrix. Afterward, this matrix is fed into an agglomerative clustering algorithm. If all measurement runs are similar, the clustering finds only one cluster. If there are at least two clusters, it is assumed that there are divergent runs.

2.4 Related Work

Different work, such as empirical studies, guidelines, or methods for detection, have been proposed to address dissimilar benchmark runs. For instance, Traini et al. [8] showed that a significant number of the 586 benchmarks they examined do not always reach a steady state. That is, these benchmarks suffer from performance fluctuations and thus produce dissimilar experiment runs. In another work, Costa et al. [2] investigated 123 Java projects and their micro-benchmarks regarding five bad practices. To mitigate these bad practices, the authors provide several recommendations for developing such benchmarks. To predict whether a benchmark is stable ahead of its execution, Laaber et al. [4] introduced a machine learning-based approach utilizing 58 statistical source code features. The model was trained on 4,461 Go benchmarks and exhibited an area under the curve of up to 90%.

3 APPROACH

3.1 Dissimilarity Measures

In contrast to SAME, which relies only on distance-based measures, our approach utilizes measures that capture different relationships between a pair of measurements. In the following, we describe the five measures $\in [0; 1]$ describing the similarity of two measurement runs (we refer to it as time series) Y and X with length n. The closer a measure is to 0, the more similar Y and X are to each other.

M1 The *Correlation Dissimilarity* considers the Pearson correlation coefficient $\rho_{X,Y}$ to capture whether two time series have the same timely behavior. We calculate $1 - max(\rho_{X,Y}, 0)$ to have a perfect correlation yielding zero.

M2 The key idea of *Compression-based Dissimilarity* is to compress two time series and find similar patterns within them. First, each time series is transformed into a string of characters with Symbolic Aggregate approXimation (SAX) [7]. Then, each transformed time series is compressed by finding recurring patterns in the time series and replacing them with a symbol. If two time series show the same patterns but are shifted in time, the two compressions yield the same length. Let c_{XY} be the length of the compression of the concatenated time series, c_X and c_Y the length of the compression of the individual time series, we calculate this measure as $\frac{2 \cdot c_{XY}}{c_X + c_Y} - 1$.

[1] The implementation of the heuristic, SAME, and the complete evaluation is available at https://codeocean.com/capsule/2914545/tree/v1

[2] GitHub: https://github.com/SEALABQualityGroup/icpe-data-challenge-jmh

M3 The *Fourier Coefficient Distance* quantifies the distance between the time series in the spectral domain. To this end, the fast Fourier transformation is applied to each time series. Then, the normalized L$_2$ norm $\frac{\|X-Y\|_2}{\|X\|_2+\|Y\|_2}$ is applied to measure the similarity between the transformed time series.

M4 The *Cosine Dissimilarity* quantifies the similarity of two time series with the cosine similarity $S_c = \frac{\langle X,Y \rangle}{\|X\|_2^2+\|Y\|_2^2}$. To have an absolute similarity yielding zero, we calculate $1 - max(S_c(X,Y), 0)$.

M5 The *Kolmogorov–Smirnov Statistic* quantifies the maximum distance between the two empirical distribution functions F of the two time series and is defined as $\sup_t |F_X(t) - F_Y(t)|$.

3.2 Proposed Heuristic

To assess the similarity between two measurement runs, we apply the measures M1 to M5. In case the majority of measures (i.e., at least three of five) exhibit a value above a pre-set threshold θ, we consider two runs dissimilar. We set the threshold[3] $\theta = 0.25$ because we tend to be risk-averse and would rather repeat a benchmark run than publish a dissimilar run. Figure 1 shows four examples with the associated measures (red indicates a value > 0.25). The offset examples (left column) and the double frequency example (top right) are labeled dissimilar as at least three measures exhibit a value greater than 0.25. In contrast, the noise example (bottom right) is labeled as similar.

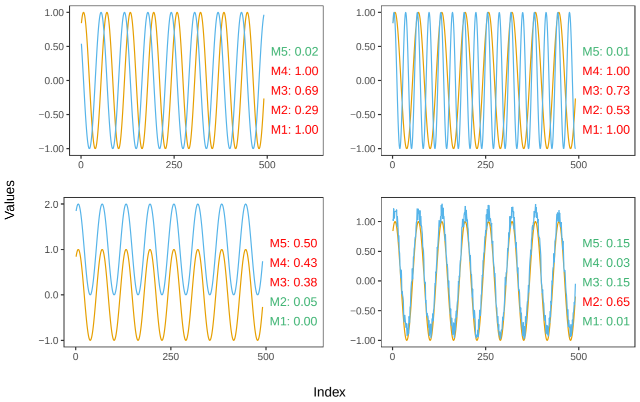

Figure 1: Schematic idea of the proposed heuristic.

To decide if a benchmark run (in this case of the data set, the ten measurement repetitions) is dissimilar, Algorithm 1 is applied. First, the heuristic calculates for each pair of measurement runs the five measures (Line 3–6), leading to $5 \cdot \sum_i^{n-1} i = \frac{5}{2}(n-1)n$ measures stored in a $0.5(n-1)n \times 5$ matrix. In the case of the data set, the resulting matrix has the dimensions 45×5. Then in Line 7, the mean is applied to the different measures so that each of the five measures represents the averaged value over the whole benchmark run. Finally, if three of the five averaged measures yield a value higher than 0.25, the benchmark run is considered dissimilar (Line 11–13).

Algorithm 1: Proposed heuristic.

```
Data: measurement runs $r$
Result: Decision $d$ whether the runs are similar.
1  $m$ = [];
2  $count$ = 0;
3  for $i = 0; i < size(r)-1; i$++ do
4      for $j = i+1; i < size(r); j$++ do
5          $[m_1, m_2, m_3, m_4, m_5]$ = calcDissimilarity($r[i], r[j]$);
6          $m$.appedRow($[m_1, m_2, m_3, m_4, m_5]$);
7  m = columnMean(m);
8  for $i = 0; i < 5; i$++ do
9      if $m[i] > \theta$ then
10         count++;
11 if $count > 2$ then
12     return dissimilar
13 return similar
```

3.3 Analysis of the Data Set

To illustrate the used measures and analyze the data set, Figure 2 shows averaged values for each application in the data set and for each dissimilarity measure. More precisely, each cell is the averaged measure over all benchmarks in the application. Again, a value of 0 means perfect similarity according to the measure. Measure M2 reports high values for dissimilarity throughout all applications, while other measures like M4 or M5 report low values for all applications. The application exhibiting the highest dissimilarity is *HdrHistogram*. To this end, Figure 3 depicts the dissimilarity measure for each benchmark used for *HdrHistogram*. Here, Measure M1–M4 yield high values for almost all benchmarks. The benchmark run with the highest dissimilarity is B17, which can be explained by taking Figure 4 into account. For instance, the last 500 measurements of all runs are different. In addition, some runs, such as Run 1 or 7, show a wave pattern, while others, such as Run 8, show no or very weak wave patterns.

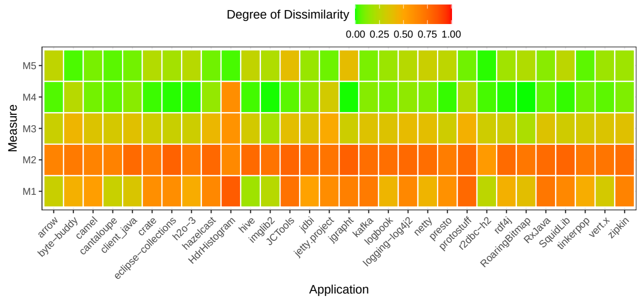

Figure 2: Averaged measures for each application.

4 EVALUATION

4.1 Peer Review Assessment

We applied a peer review-based approach with four reviewers, leading to six review pairs. To this end, we divided the data set randomly

[3] We have tried different numbers, and setting the threshold to 0.25 had shown the best results according to our risk aversion

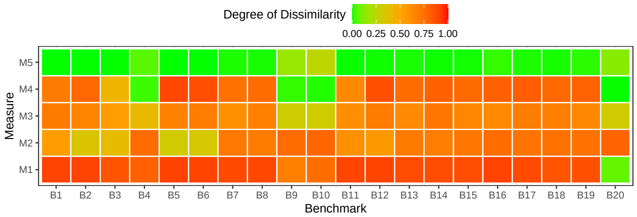

Figure 3: Averaged measures for each benchmark for the HdrHistogram application.

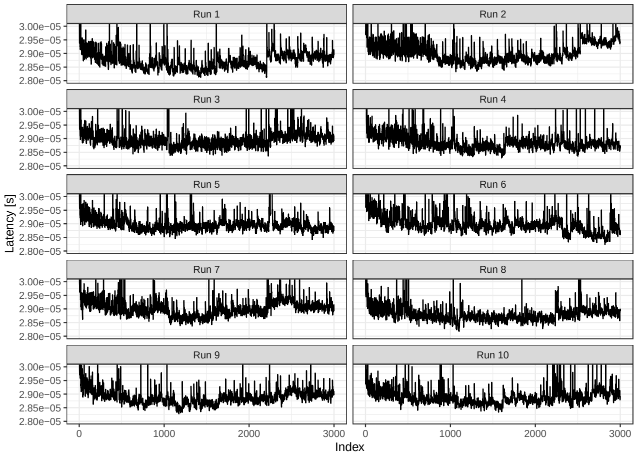

Figure 4: Measurement runs for B17 from HdrHistogramm.

into six parts. Each part was then analyzed by one of the possible reviewer pairs. The assignment of the pairs was unknown to the reviewers to guarantee unbiased reviews. To analyze a benchmark run, the reviewer saw all ten measurements at the same time and had to decide if the measurements are similar or dissimilar.

To assess the agreement of the four reviewers, we first investigate the pairwise agreement. To this end, Table 1 shows the agreement between the reviewers in percentage. The first and second reviewer had the lowest agreement, while the second and fourth reviewer had the highest agreement. In total, the reviewers agreed on 55% of all runs. The corresponding Fleiss' Kappa analysis yielded a value of 0.1, that is, a slight agreement between the four reviewers. The agreement rate and the low kappa value indicate that deciding whether a benchmark run is similar in the data set is challenging and might even be subjective.

Table 1: Reviewer agreement.

R1 - R2	R1 - R3	R1 - R4	R2 - R3	R2 - R4	R3 - R4
47%	57%	35%	53%	77%	62%

In the following, we only consider the cases where the reviewers agreed, that is, 324 out of the original 568 benchmark runs. On this subset, the reviewers identified 189 benchmark runs as dissimilar and 135 as similar.

4.2 Investigation of SAME

As mentioned above, we evaluated SAME on the remaining 324 benchmark runs. Table 2 shows the comparison of SAME's classification with the reviewed results. SAME correctly identified 112 of 135 benchmark runs as similar, resulting in a specificity of 82% (i.e., the quality of the approach detecting similar benchmark runs). However, SAME also classified 137 runs as similar, even though they were labeled dissimilar.

Table 2: Confusion matrix between review and classification.

SAME Review	Similar	Dissimilar
Similar	112	23
Dissimilar	137	52

4.3 Investigation of the Heuristic

We evaluated our heuristic on the remaining 324 benchmark runs. Table 3 shows the confusion matrix between the classification of the heuristic and the reviewed results. Our heuristic found 173 of 189 benchmark runs that were labeled as dissimilar. This leads to a sensitivity of 92%, that is, the quality of our heuristic's ability to detect dissimilar benchmark runs. However, the heuristic wrongly classifies 82 benchmark runs as dissimilar. Nonetheless, in this case, it is better to classify one benchmark run too many as dissimilar than to overlook one dissimilar benchmark run. This is because the dissimilar classified benchmark runs can still be analyzed afterward.

Table 3: Confusion matrix between review and classification.

Heuristic Review	Similar	Dissimilar
Similar	53	82
Dissimilar	16	173

4.4 Discussion and Threats to Validity

As our heuristic was evaluated on a peer-reviewed analysis, the results must be viewed critically, as the analysis may be prone to errors. Thus, we cannot verify the correctness of the heuristic. However, we see the heuristic as a helpful tool that can be used to pre-filter dissimilar runs, and then an expert can make a final decision. Nonetheless, applying the heuristic on labeled data sets is easy because it is generic. Also, the heuristic can be easily adapted for other purposes since the only parameter is the similarity threshold θ, which can be set individually. In case measurements should be wrongly classified as dissimilar rather than vice versa, we recommend low values (e.g., 0.25, as we did in the experiments). However, the parameter could also be tuned to any data set.

5 CONCLUSION

To support the task of determining whether performance measurements are similar, we used a data set of 586 micro-benchmarks to analyze the data, evaluate the approach of Leznik et al. [6], and propose and test a heuristic. We used a peer review to assess the dissimilarity of 586 micro-benchmark runs and found that even for humans, this task is difficult. Our heuristic exhibits a sensitivity of 92% on this data set, enabling a pre-filtering of measurements that an expert can verify.

REFERENCES

[1] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960).

[2] Diego Costa, Cor-Paul Bezemer, Philipp Leitner, and Artur Andrzejak. 2019. What's wrong with my benchmark results? studying bad practices in JMH benchmarks. *IEEE Transactions on Software Engineering* 47, 7 (2019), 1452–1467.

[3] Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76, 5 (1971).

[4] Christoph Laaber, Mikael Basmaci, and Pasquale Salza. 2021. Predicting unstable software benchmarks using static source code features. *Empirical Software Engineering* 26, 6 (2021), 1–53.

[5] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174.

[6] Mark Leznik, Johannes Grohmann, Nina Kliche, André Bauer, Daniel Seybold, Simon Eismann, Samuel Kounev, and Jörg Domaschka. [n. d.]. Same, Same, but Dissimilar: Exploring Measurements for Workload Time-series Similarity.. In *ICPE*. 89–96.

[7] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15, 2 (2007), 107–144.

[8] Luca Traini, Vittorio Cortellessa, Daniele Di Pompeo, and Michele Tucci. 2023. Towards effective assessment of steady state performance in Java software: are we there yet? *Empirical Software Engineering* 28, 1 (2023), 1–57.

[9] Jóakim v. Kistowski, Jeremy A Arnold, Karl Huppler, Klaus-Dieter Lange, John L Henning, and Paul Cao. 2015. How to build a benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. 333–336.