Investigating Benchmark Correlations when Comparing Algorithms with Parameter Tuning

Lee A. Christie
University of Stirling, UK
lee.christie@stir.ac.uk

Alexander E.I. Brownlee
University of Stirling, UK
alexander.brownlee@stir.ac.uk

John R. Woodward
Queen Mary University of London, UK
j.woodward@qmul.ac.uk

ABSTRACT

Benchmarks are important for comparing performance of optimisation algorithms, but we can select instances that present our algorithm favourably, and dismiss those on which our algorithm under-performs. Also related are automated design of algorithms, which use problem instances (benchmarks) to train an algorithm: careful choice of instances is needed for the algorithm to generalise.

We sweep parameter settings of differential evolution to applied to the BBOB benchmarks. Several benchmark functions are highly correlated. This may lead to the false conclusion that an algorithm performs well in general, when it performs poorly on a few key instances. These correlations vary with the number of evaluations.

CCS CONCEPTS

• **Computing methodologies → Search methodologies**;

KEYWORDS

benchmarks, BBOB, ranking, differential evolution, continuous optimisation, parameter tuning, automated design of algorithms.

ACM Reference Format:

Lee A. Christie, Alexander E.I. Brownlee, and John R. Woodward. 2018. Investigating Benchmark Correlations when Comparing Algorithms with Parameter Tuning . In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3205747

1 INTRODUCTION

Continuous optimisation samples a continuous search space, to minimise (or maximise) an objective. The Black-Box Optimization Benchmarking (BBOB-2009) benchmarks comprise 24 noiseless and 30 noisy functions[2] commonly used to compare continuous optimisation metaheuristics [6]. It is well known that no one algorithm performs well over all functions, so we ask *how good are the benchmarks at teasing out the performances of different algorithms?* We also consider the implications for automatic design of algorithms (ADA), where the set of functions used for training is critical.

Two dangers of benchmarking are: not investing the same number of evaluations in tuning two algorithms (e.g. tuning your own more); and making comparisons of the form "Algorithm A outperformed B on 20/24 of the benchmarks, while B outperformed A on the remaining 4" (if the 20 are highly correlated, yet the 4 are not, then B could be said to be more general than A.)

We investigate the tuning of parameters on a set of benchmarks and examine correlations in performance between the algorithms on the functions. We consider algorithms with different parameter setting to be different algorithms, and therefore, the algorithms defined by the set of parameters defines the algorithm design space.

In this paper we make three major contributions:

(1) Algorithm performances on several functions are highly correlated (within the set of algorithms we consider).
(2) The number of evaluations has a dramatic impact when concluding either which algorithm performs best.
(3) A 2-stage (coarse- then fine-grained) systematic sweep of the parameters shows how performance varies with number of evaluations for different parameter settings.

Further experimental detail and analysis can be found in [1].

2 EXPERIMENTAL DESIGN

Our study considers settings for the differential evolution parameters *differential weight*, F, and *crossover probability*, CR, applied to the BBOB benchmarks. The experiment covered two stages to keep total run time practical: a coarse-grained sweep of the full range for each parameter, and a fine-grained sweep around the near-optimal setting found at stage 1. The BBOB benchmarks used were the 24 noiseless functions, on the 10-dimensional search space $[-5, 5]^{10}$.

The coarse-grained sweep runs were limited to 10 generations. For each algorithm configuration, each of the 24 benchmarks provide a ranking of configurations from best to worst. To reach a consensus of the best configuration over all benchmarks, the rankings were used a ballots in an instant run-off vote.

The fine-grained stage used a neighbourhood region of 10×10 samples centred around the best parameter settings from the course-grained sweep, with samples spread out in increments of 0.01. This gives us 100 DE configurations to compare.

3 RESULTS

Full result data is available from http://hdl.handle.net/11667/109. The coarse-grained sweep found $F = 0.3$, $CR = 0.9$ as overall consensus optima. The resulting 10×10 region for the fine-grained sweep was $F \in \{0.25, 0.26, \ldots 0.34\}$ $CR \in \{0.85, 0.86, \ldots 0.94\}$.

We consider the *meta-fitness* of each algorithm on each benchmark after a certain number of generations to be the average fitness reached by the 1000 repeat runs of the algorithm after that many generations. This is known as a *fixed-budget* measure of algorithm performance: this was chosen over the more conventional *fixed-target* measure so that experiments run for less time, as convergence is not required. This method is often used in parameter tuning (e.g.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3205747

[4]). For each benchmark, at each generation, we have a meta-fitness value from each of the 100 algorithm configurations. The Spearman's Rank correlation between these sets of values for pairs of algorithms starts near zero at generation 0, as the first generation of all algorithm runs is random. The correlation then varies dramatically in the range $(-1, 0)$ with generation limit.

Since we have 24 functions to compare pairwise, we can construct a correlation matrix between the functions. Each cell corresponds to the correlation in meta-fitness for all algorithms between the two functions in the row and column header. The correlation matrix for generation limit 10 is shown in Table 1.

We also calculate the median value of correlation between a given benchmark with every other benchmark. These also vary considerably with generation. In generation 2, *all* functions are strongly positively correlated with one-another, except F23 which has a median correlation of −0.30. The lowest median correlation observed is F7 at generation 5, where median correlation is −0.90.

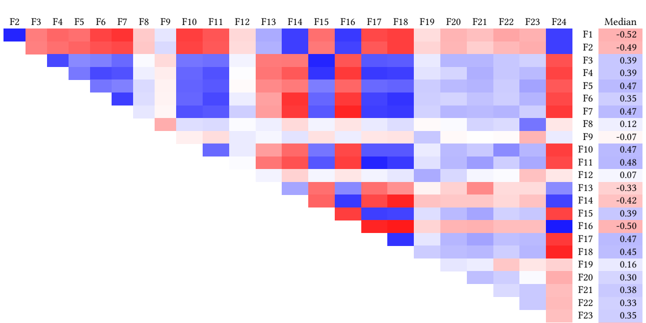
At generation 8, there is a crossing point: many functions with high median correlations drop to negative correlations, and many with previously negative median correlation move to high positive median correlations. F7, F17, and F18 maintain a strong positive correlation with one another and as generation limit increases, the three approach a correlation near 1.0 with one another. Thus, these three functions (within the scope of which we have studied them) give us the same information about the algorithm's performance, and so it is redundant to run all three in this case.

For most generation limits L during these runs, most functions are on average positively correlated with a few exceptions. This is visible in Table 1 as most cells being blue..

4 CONCLUSION AND FUTURE WORK

We applied DE to the well-known BBOB benchmarks. Sweeping across 2 DE parameters, we found that early in the run there was very little correlation between the performances of different parameter settings, but after only a few generations, most functions are highly correlated, while some are anti-correlated. These correlations vary as the number of generations increases. The correlations

Table 1: Matrix of correlations between all benchmark functions at generation 10. Blue cells denote a strongly-positive correlation, red cells denote a strongly-negative correlation, white cells denote zero correlation. The right-most column is the median correlation of the related function with every other function.



we observed do not just occur within the five groups of functions based on their structure (defined by [2]), corroborating earlier results [6] that clustered the BBOB functions in to different groups according to algorithm performance. To draw broader conclusions, we intend to investigate how benchmark functions compare for other algorithms, such as CMA-ES and conventional genetic algorithms: preliminary experiments with a GA show the same pattern, but still to be determined is whether correlated functions for DE are also correlated when using, for example, CMA-ES.

The problem with correlated benchmarks is that, given the performance of DE on one of the functions, the other two add no information to the benchmarking process. If we can identify highly-correlated benchmark functions in terms of algorithm performance, we may be able to select and eliminate redundant functions from a benchmark set. Furthermore, if we can identify highly-correlated benchmarks, we may also be able to identify possible combinations of performance features absent in the benchmarks. These gaps could be filled by generating new benchmarks, e.g. [3, 5, 7].

In ADA, benchmark instances can be used to *train* our algorithm; and demonstrate the utility of our algorithm on unseen instances. As for machine learning, these two sets of instances (training and test) need to be somehow similar for the trained model to perform well on the test set. That these correlations vary with the number of function evaluations must be considered when using exploratory landscape analysis to predict performance and select appropriate algorithms as advocated by [6]. Choice of the evaluation budget must match the available budget in the target "unseen" instances, or the performance model will be flawed, biased by correlations present only in part of the search space. Furthermore, it is false to assume that parameter tuning on a smaller evaluation budget will lead to fair comparisons with a larger budget [8].

5 ACKNOWLEDGEMENT

Funded by UK EPSRC [grants EP/N002849/1, EP/J017515/1]. Experiments used EPSRC funded ARCHIE-WeSt HPC [grant EP/K000586/1].

6 DATA ACCESS STATEMENT

The data sets, including all computed features, the evolved policies, and their performances, and the visualisations for all feature sets, are available from http://hdl.handle.net/11667/109.

REFERENCES

[1] L. Christie, A. Brownlee, and J. Woodward. Investigating benchmark correlations when comparing algorithms with parameter tuning (detailed experiments & results). Technical report, Univ. Stirling, 2018. http://hdl.handle.net/1893/26956.
[2] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829.
[3] B. Lacroix, L. Christie, and J. McCall. Interpolated continuous optimisation problems with tunable landscape features. In *GECCO Comp.*, pages 169–170. 2017.
[4] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. StÃijtzle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, sep 2016.
[5] J. McCall, L. Christie, and A. Brownlee. Generating easy and hard problems using the proximate optimality principle. *GECCO Companion*, pages 767–768, 2015.
[6] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, and C. Weihs. Analyzing the bbob results by means of benchmarking concepts. *Evolutionary Computation*, 23(1):161–185, 2015. PMID: 24967695.
[7] K. Smith-Miles and S. Bowly. Generating new test instances by evolving in instance space. *Computers & OR*, 63:102–113, Nov 2015.
[8] R. Tanabe and A. Fukunaga. Tuning differential evolution for cheap, medium, and expensive computational budgets. In *IEEE CEC*, pages 2018–2025, 2015.