Anytime Benchmarking of Budget-Dependent Algorithms with the COCO Platform

Tea Tušar, Nikolaus Hansen, Dimo Brockhoff

Anytime Benchmarking of Budget-Dependent Algorithms with the COCO Platform

Tea Tušar
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia
tea.tusar@ijs.si

Nikolaus Hansen
Inria
École Polytechnique CMAP
(UMR 7641)
Palaiseau, France
nikolaus.hansen@inria.fr

Dimo Brockhoff
Inria
École Polytechnique CMAP
(UMR 7641)
Palaiseau, France
dimo.brockhoff@inria.fr

ABSTRACT

Anytime performance assessment of black-box optimization algorithms assumes that the performance of an algorithm at a specific time does not depend on the total budget of function evaluations at its disposal. It therefore should not be used for benchmarking budget-depending algorithms, i.e., algorithms whose performance depends on the total budget of function evaluations, such as some surrogate-assisted or hybrid algorithms. This paper presents an anytime benchmarking approach suited for budget-depending algorithms. The approach is illustrated on a budget-dependent variant of the Differential Evolution algorithm.

1. INTRODUCTION

In black-box optimization, the problem to be optimized cannot be explicitly written as a function of its input parameters (if an underlying function exists, it is unknown). This is often the case with real-world problems where solutions are evaluated using simulations. Without the possibility of exploiting the structure of the function, optimization algorithms resort to repeatedly sample its decision space and use previously evaluated solutions to steer the search towards promising regions. Since the evaluations of real-world problem functions are often more time-consuming than the internal computations of optimization algorithms, the running time, or *runtime*, of an algorithm is generally measured by counting the number of performed function evaluations. The goal of an algorithm in black-box optimization is thus to find satisfactory solutions to the given problem in as few function evaluations as possible.

When measuring the performance of an algorithm in the black-box setting, we are interested in the required runtime to reach a target value. Or rather, we wish to obtain all runtimes corresponding to increasingly difficult targets [3]. In problems with a single objective, the targets are usually defined as differences from the optimal function value, while in problems with multiple objectives, the targets are determined as differences from the optimal value of a multiobjective performance indicator.

The proportion of reached targets plotted against the runtimes in the sense of an empirical cumulative distribution function yields a *data profile* [7]—a graph showing the *anytime* performance of an algorithm, essentially mimicking the convergence graph (the plot of best found function or indicator values over time). In addition to being easy to interpret, the data profile has another important advantage—it can be used to represent algorithm performance aggregated over multiple runs on different problems of the same dimensionality (see Section 2 for more details). This considerably alleviates presentation and understanding of algorithm results on a large number of problems.

The underlying assumption in anytime performance assessment is that the performance of an algorithm at a specific runtime does not depend on the total budget of function evaluations. That is, performance of an algorithm at 1000 function evaluations is expected to be the same if the algorithm was run with a budget of 1000 or 100 000 function evaluations (everything else being equal). If this is not the case, data profiles should not be employed to infer performance of the same algorithm with a total budget different from the one used in the experiments.

Algorithms can depend on the total budget for different reasons. Consider for example surrogate-assisted approaches. They construct surrogate models of the optimization problem and combine actual function evaluations with evaluations on the models. While some algorithms work in a budget-independent way, e.g. [6], others save some true function evaluations for the end (just before the budget is exhausted), making them budget-dependent, e.g. [9]. Similarly, hybrid genetic algorithms that combine genetic algorithms with local search methods can reserve a number of final function evaluations to additionally improve the current best solutions [2]. Another example of budget-dependent algorithms are evolutionary algorithms that set any of their parameters based on the total budget [1].

To address this issue, we propose an approach for benchmarking budget-dependent algorithms that allows anytime performance assessment of their results. It is based on the anytime benchmarking from the Comparing Continuous Optimizers (COCO) platform [4]. The approach is demonstrated on a budget-dependent variant of Differential Evolution (DE) [8] ran on the `bbob` test suite [5].

In the following, we first present some background on anytime benchmarking with the COCO platform (Section 2). The new approach is described in Section 3 followed by a discussion on its time complexity. An illustration with the DE algorithm is shown in Section 4. Section 5 presents some concluding remarks.

2. ANYTIME BENCHMARKING IN COCO

COCO (`https://github.com/numbbo/coco`) is a platform that facilitates benchmarking of optimization algorithms by automatizing this procedure and providing data of previously run algorithms for comparison [4]. An important part of COCO's anytime benchmarking approach [3] is the presentation of algorithm results in the form of data profiles [7].

Consider a single run of algorithm $\mathcal{A}$ on problem p. Given l increasingly difficult targets $\tau_1, \tau_2, \ldots, \tau_l$, it is easy to compute the corresponding runtimes $r_1^p, r_2^p, \ldots, r_l^p$ needed by algorithm $\mathcal{A}$ to reach each of these targets. If the target τ_j was not reached, r_j^p is undefined. A data profile for algorithm $\mathcal{A}$ is then constructed by plotting for each number of evaluations the proportion of targets reached by $\mathcal{A}$ in a runtime equal to or smaller than the number of evaluations. In other words, a data profile is the empirical cumulative distribution function of the recorded runtimes $r_1^p, r_2^p, \ldots, r_l^p$.

Data profiles can be further exploited to show aggregated information over randomized repetitions of running algorithm $\mathcal{A}$ on problem p. Instead of using repeated runs of $\mathcal{A}$ on p (which is sensible only for stochastic algorithms), randomization in COCO is achieved by running the same algorithm $\mathcal{A}$ on different instances of problem p (for example, translated versions of the same problem).

Consider k instances of the problem p, denoted here as $p(\theta_1)$, $p(\theta_2), \ldots, p(\theta_k)$. Like before, the runtime $r_j^{p(\theta_i)}$ at which algorithm $\mathcal{A}$ achieves target τ_j on problem instance $p(\theta_i)$ can be easily calculated for each i and j and is undefined when the target has not been reached. In order to be able to compare algorithms of different success probabilities (for example an algorithm that always reaches difficult targets, but does this slowly, with an algorithm that sometimes reaches a target quickly while other times fails to reach it at all), we simulate restarts of each algorithm via a *bootstrapping procedure*. The N bootstrapped simulated runtimes $r_{j,1}, r_{j,2}, \ldots, r_{j,N}$ of the artificially restarted algorithm to reach a target τ_j are computed from the *recorded* runtimes $r_j^{p(\theta_i)}$ of algorithm $\mathcal{A}$ (for a large N, e.g. $N = 1000$) as:

```
for c ← 1, ..., N do                          ▷ Repeat N times
    r_{j,c} ← 0                               ▷ Initialize runtime
    loop
        i ← random({1, ..., k})               ▷ Choose an instance
        if r_j^{p(θ_i)} is defined then       ▷ Successful
            r_{j,c} ← r_{j,c} + r_j^{p(θ_i)}
            break loop
        else                                  ▷ Unsuccessful
            r_{j,c} ← r_{j,c} + r_max^{p(θ_i)}
        end if
    end loop
end for
return r_{j,1}, r_{j,2}, ..., r_{j,N}
```

if at least one of the recorded runtimes is finite. Note that the total runtime of $\mathcal{A}$ on $p(\theta_i)$, $r_{\max}^{p(\theta_i)}$, is added each time an unsuccessful trial is picked. Runtimes $r_{j,1}, r_{j,2}, \ldots, r_{j,N}$ are undefined for targets τ_j that were not reached in any of the problem instances. The resulting $N \cdot l$ runtimes (of which some undefined) are used to construct the data profile in an analogous way as before, but this time the y axis shows the proportion of targets reached out of $N \cdot l$ ones.

Finally, data profiles are also able to aggregate runtime results over problems of the same dimensionality that optimize a different function. Imagine a test suite consisting of m such problems with multiple instances. After bootstrapping is performed for each problem separately, there are $m \cdot N \cdot l$ function and target pairs and the same number of bootstrapped runtimes. The aggregated data profile for algorithm $\mathcal{A}$ can thus be constructed by plotting for each number of evaluations the proportion of function and target pairs reached by $\mathcal{A}$ in a runtime equal to or smaller than the number of evaluations.

It is important to note that runtimes are never aggregated over different dimensions since problem dimension is often used as an algorithm parameter. This also allows scalability studies. See [3] for more details on COCO's performance assessment procedure.

3. A BENCHMARKING APPROACH FOR BUDGET-DEPENDENT ALGORITHMS

The idea for benchmarking a budget-dependent algorithm $\mathcal{A}$ is very simple: the algorithm is run with increasing budgets and the resulting runtimes are presented in a single data profile. This is achieved by means of an 'artificial' algorithm $\widetilde{\mathcal{A}}$ that works as follows.

Consider K increasing budgets $b_1, b_2, \ldots, b_K$ and K budget-dependent algorithm variants $\mathcal{A}_{b_1}, \mathcal{A}_{b_2}, \ldots, \mathcal{A}_{b_K}$. The algorithm $\widetilde{\mathcal{A}}$ first works as algorithm $\mathcal{A}_{b_1}$ for budgets $b \leq b_1$, then works as algorithm $\mathcal{A}_{b_2}$ for budgets b, where $b_1 < b \leq b_2$, and so on, finishing by mimicking algorithm $\mathcal{A}_{b_K}$ for budgets b, where $b_{K-1} < b \leq b_K$ (see also Figure 1). In an algorithmic notation (where x_i denotes the ith solution explored by the corresponding algorithm):

```
b_0 ← 0                                   ▷ Initialize a budget preceding b_1
for j ← 1, ..., K do                      ▷ Iterate over budgets
    for i ← 1, ..., b_{j-1} do
        A_{b_j}(x_i)                      ▷ Run A_{b_j} ignoring its output
    end for
    for i ← b_{j-1} + 1, ..., b_j do
        Ã(x_i) ← A_{b_j}(x_i)             ▷ Ã mimics A_{b_j}
    end for
end for
```

Although the first b_{j-1} evaluations of the algorithm $\mathcal{A}_{b_j}$ are ignored by $\widetilde{\mathcal{A}}$, they need to be performed so that $\mathcal{A}_{b_j}$ is in the correct state at evaluation $b_{j-1} + 1$, when it starts to be mimicked by algorithm $\widetilde{\mathcal{A}}$.

As shown with the red run in Figure 1, $\mathcal{A}_{b_j}$ might not contribute to $\widetilde{\mathcal{A}}$ at all if the performance of $\mathcal{A}_{b_i}$ is better for some $b_i < b_j$. On the other hand, if $\mathcal{A}_{b_j}$ is significantly better than $\mathcal{A}_{b_{j-1}}$ (for example, the green vs. the yellow run), this causes a 'jump' in the performance of $\widetilde{\mathcal{A}}$. Note also that the best-so-far profile of $\widetilde{\mathcal{A}}$ does not necessarily follow the overall best-so-far profile of $\mathcal{A}_{b_j}$, but only its best-so-far profile after b_{j-1} function evaluations (notice the yellow and

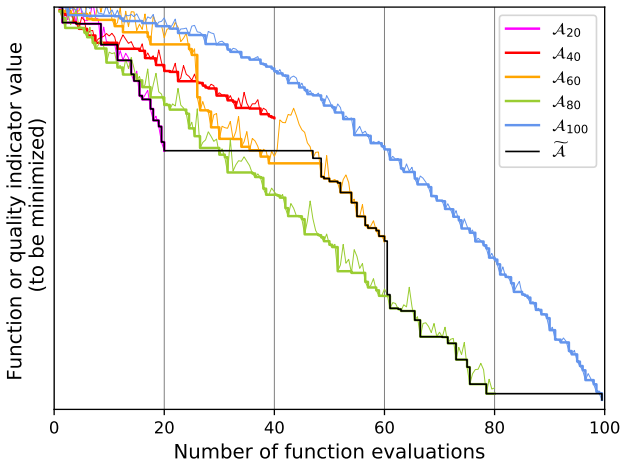

Figure 1: An illustration of the 'artificial' algorithm $\widetilde{\mathcal{A}}$ constructed from five runs of algorithm $\mathcal{A}$ ($\mathcal{A}_b$ means the algorithm was run with the budget of b function evaluations). Thin and thick lines show the actual and best-so-far performance for each run of $\mathcal{A}$, respectively.

black lines after 40 function evaluations).

Composing the performances of algorithm $\mathcal{A}$ with different budgets into algorithm $\widetilde{\mathcal{A}}$ results in an estimation of the anytime performance of $\mathcal{A}$. The quality of the estimation depends on the number of budgets K—more budgets enable a better estimation, but make the procedure more time consuming.

One could run the budget-dependent variants of algorithm $\mathcal{A}$ for every budget between 1 and b_K thus obtaining the best possible estimate. However, this would require

$$\sum_{j=1}^{b_K} j = \frac{b_K(b_K+1)}{2}$$

total evaluations. Diluting the budgets by taking only every Mth one does not help to significantly reduce the number of total evaluations. A more promising approach is that of using equidistant budgets in the logarithmic scale. For example, K such budgets between 1 and 10^M require

$$\sum_{j=0}^{MK} 10^{j/K} = \frac{10^{1/K+M}-1}{10^{1/K}-1}$$

total evaluations. Table 1 contains total evaluations for this case for some values of K and M. The actual number of evaluations is likely to be smaller than these numbers due to some consecutive (small) budgets being rounded to the same integer number.

4. EXAMPLE

We present a small example to demonstrate the proposed anytime benchmarking procedure on the COCO platform. The algorithm used in this example is a budget-dependent variant of Differential Evolution (DE) [8], a well-known evolutionary algorithm. While the original DE algorithm is

Table 1: An upper bound of function evaluations required for benchmarking budget-dependent algorithms with K budgets between 1 and 10^M that are equidistant in the logarithmic scale, for some selected values of K and M.

$\left\lceil \frac{10^{1/K+M}-1}{10^{1/K}-1} \right\rceil$		M		
		3	4	5
K	10	4.859	48.618	486.208
	20	9.188	91.947	919.540
	50	22.198	222.165	2.221.835
	100	43.889	439.270	4.393.094

Table 2: Population size of the budget-dependent DE computed for some selected values of budget multipliers m_{budg} and problem dimensions n.

$\lfloor 3\log_{10}^2(n \cdot m_{\text{budg}}) \rfloor$		m_{budg}		
		10	100	1000
n	2	5	15	32
	5	8	21	41
	10	12	27	48
	20	15	32	55

budget-independent, a study shows that setting its parameters, especially population size, in connection to the total budget of evaluations can improve its results [1].

In the experiments we use the DE implementation from the `scipy` Python package (`https://www.scipy.org/`) with the following parameters:

- Initialization = Latin Hypercube sampling
- DE strategy = best/1/bin
- Population size = *variable* (see text)
- Weight F = random in the interval $[0.5, 1)$
- Crossover probability $CR = 0.7$
- No local optimization of the final solution
- Relative tolerance for convergence = 10^{-9}

This implementation computes the population size based on the problem dimension n (for a user-specified multiplier m_{pop}, the population size is calculated as $n \cdot m_{\text{pop}}$). This was bypassed in order to make population size budget-dependent. For a problem with n dimensions and a budget multiplier m_{budg}, the actual budget in COCO is computed as $n \cdot m_{\text{budg}}$ and the population size of DE is calculated as

$$\lfloor 3\log_{10}^2(n \cdot m_{\text{budg}}) \rfloor .$$

Table 2 gathers the values of this formula for selected budget multipliers m_{budg} and problem dimensions n.

The experiment consisted of running five instances of DE with budget multipliers from $\{10, 31, 100, 316, 1000\}$ and at the same time composing their performances into the anytime artificial algorithm called 'DE-anytime'. All algorithms were run on the 24 problems functions from the `bbob` test suite [5] with dimensions n in $\{2, 3, 5, 10, 20\}$. Each problem was instantiated 15 times. The results for dimension 10 are presented in Figure 2. In data profiles plots produced by

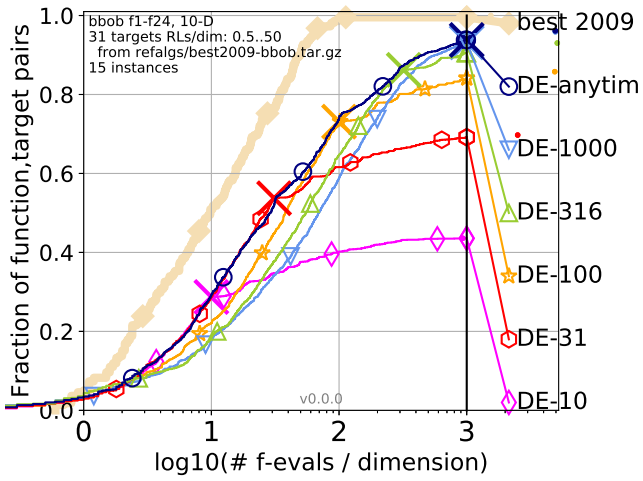

Figure 2: Data profiles for budget-dependent variants of DE run with budgets of $\{10, 31, 100, 316, 1000\}\cdot n$ number of evaluations and the 'artificial' algorithm constructed from these variants estimating anytime performance of DE. See text for further information.

COCO, the function evaluations are always divided by the problem dimension n and shown on a logarithmic scale.

The benchmark setting used in this example is COCO's *expensive setting*, in which the number of evaluations is limited to $1.000n$ and the 31 targets are defined in a relative way—according to the performance of a virtual algorithm denoted with 'best 2009' that is comprised of the best results achieved by 31 algorithms at the Black-box Optimization Benchmarking (BBOB) workshop in 2009. The targets are chosen from $[10^{-8}, \infty)$ such that the 'best 2009' algorithm just failed to reach them within the given budget of nm_{budg} evaluations, with 31 different values of m_{budg} chosen equidistantly in logarithmic scale between 0.5 and 50.

We are showing the results for dimension 10, since the differences among DE instances are best visible for this dimension. Note that the algorithms were stopped at the moment denoted by the large cross, but the data profiles increase also beyond that point due to bootstrapping (see Section 2).

From Figure 2 we can observe that DE variants with a budget of $10n$ and $31n$ evaluations achieve a very similar performance in the first $10n$ evaluations. Other DE variants are noticeably different from the first two and also among themselves, with those with lower budgets converging faster at the beginning of the run. This confirms the findings from [1] that better performance can be achieved by fitting the population size to the total budget.

The dark blue data profile corresponding to the 'DE-anytime' artificial algorithm follows the five underlying algorithms as expected. The accuracy of this estimate could be further improved if a higher number of different budgets was used.

5. CONCLUSIONS

We presented a novel approach for benchmarking budget-dependent algorithms that enables anytime performance assessment of their results. The approach demands repeated runs of an algorithm with increasing budgets. Depending on the number and size of these budgets, it can take a significant amount of time (it is quadratic in the maximal budget in the worst case). By using budgets that are equidistant in the logarithmic scale, the time complexity depends linearly on the maximal budget, making the approach more usable in practice. An example experiment showing how to use this approach in COCO will be available in COCO v2.2.

6. ACKNOWLEDGMENTS

The authors wish to thank Joshua Knowles for helpful discussions on this topic during the SAMCO workshop (Workshop on Surrogate-Assisted Multicriteria Optimization) at the Lorenz Center in Leiden, The Netherlands, 2016.

The authors acknowledge the financial support from the Slovenian Research Agency (project No. Z2-8177). This work is part of a project that has received funding from the *European Union's Horizon 2020 research and innovation program* under grant agreement No. 692286.

This work was furthermore supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

7. REFERENCES

[1] A. S. D. Dymond, A. P. Engelbrecht, and P. S. Heyns. The sensitivity of single objective optimization algorithm control parameter values under different computational constraints. In *Proceedings of CEC 2011*, pages 1412–1419, 2011.

[2] T. A. El-Mihoub, A. A. Hopgood, L. Nolle, and A. Battersby. Hybrid genetic algorithms: A review. *Engineering Letters*, 13:124–137, 2006.

[3] N. Hansen, A. Auger, D. Brockhoff, D. Tušar, and T. Tušar. COCO: Performance assessment. *ArXiv e-prints*, arXiv:1605.03560, 2016.

[4] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *ArXiv e-prints*, arXiv:1603.08785, 2016.

[5] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Research Report RR-6829, INRIA, 2009.

[6] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[7] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[8] R. Storn and K. Price. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[9] V. Volz, G. Rudolph, and B. Naujoks. Surrogate-assisted partial order-based evolutionary optimisation. In *Proceedings of EMO 2017*, pages 639–653, 2017.