Initial Design Strategies and their Effects on Sequential Model-Based Optimization

An Exploratory Case Study Based on BBOB

Jakob Bossek
School of Computer Science
The University of Adelaide, Australia

Carola Doerr
Sorbonne University, CNRS, LIP6
Paris, France

Pascal Kerschke
Information Systems and Statistics
University of Münster, Germany

ABSTRACT

Sequential model-based optimization (SMBO) approaches are algorithms for solving problems that require computationally or otherwise expensive function evaluations. The key design principle of SMBO is a substitution of the true objective function by a surrogate, which is used to propose the point(s) to be evaluated next.

SMBO algorithms are intrinsically modular, leaving the user with many important design choices. Significant research efforts go into understanding which settings perform best for which type of problems. Most works, however, focus on the choice of the model, the acquisition function, and the strategy used to optimize the latter. The choice of the initial sampling strategy, however, receives much less attention. Not surprisingly, quite diverging recommendations can be found in the literature.

We analyze in this work how the size and the distribution of the initial sample influences the overall quality of the efficient global optimization (EGO) algorithm, a well-known SMBO approach. While, overall, small initial budgets using Halton sampling seem preferable, we also observe that the performance landscape is rather unstructured. We furthermore identify several situations in which EGO performs unfavorably against random sampling. Both observations indicate that an adaptive SMBO design could be beneficial, making SMBO an interesting test-bed for automated algorithm design.

CCS CONCEPTS

• **Computing methodologies** → *Continuous space search.*

KEYWORDS

Sequential Model-Based Optimization, Design of Experiments, Initial Design, Continuous Black-Box Optimization

1 INTRODUCTION

Sequential Model-Based Optimization (SMBO) algorithms are techniques for the optimization of problems for which the evaluation of solution candidates is resource-intensive, such as problems requiring real physical experiments or problems that require computationally-expensive simulations. The latter are particularly present in almost any application of Artificial Intelligence, most notably in terms of parameter tuning problems – a problem that is also omnipresent in Evolutionary Computation [37]. SMBO-based techniques are among the most successfully applied hyper-parameter tuning methods [1, 18, 28, 35], so that research on this family of iterative optimization heuristics has gained significant traction in the last decade. SMBO forms today an integral part of the state-of-the-art heuristic solvers. Its probably best-known representatives are *Bayesian Optimization* (see surveys by [40, 46, 49]) and, in particular, *Efficient Global Optimization* (EGO, [30]).

The generic SMBO method works as follows. An initial design of points is sampled and evaluated with the true objective function. The eponymous *sequential* part iteratively (1) builds a surrogate of the true objective function (on basis of the already evaluated samples), (2) proposes new samples by optimizing a so-called *infill-criterion* (which is sometimes referred to as *acquisition function*), (3) evaluates these additional samples, and (4) integrates these samples, together with their quality indicators ("function values", "fitness") into the memory. Each of these steps offers a great variety of design choices, which all may affect the performance of the SMBO procedure. Which surrogate model should be used? Which of the countless infill criteria to use? What method should be used to create the initial sample and what proportion of the overall budget should be spent on the initial design? While a large body of works addresses the first two questions (see surveys mentioned above), the latter two questions are treated rather poorly. In this work we aim to shed light on the relevance of a suitably chosen initial sampling strategy. More precisely, we study how the size of the initial design and the strategy used to generate it affects the performance of SMBO. As a well-established benchmark environment offering a great variety of different numerical optimization problems, we chose the 24 noiseless BBOB functions (in different dimensions) as test-bed for our investigation.

Our setup comprises of varying the initial design strategy (classical uniform and Latin-Hypercube-Sampling (LHS) as the most frequently used methods and quasi-random Halton and Sobol' sequences), the total budget, and the fraction of this total budget that is used to build the initial sample. We study a total of 720 problems, which are evaluated against 40 different initial design strategies.

Our general observation is that SMBO performance tends to decrease with increasing initial design ratio, which is in line with the general expectation that adaptive search should outperform non-adaptive sampling. This may justify extreme settings such as the singleton initial design used in the SMAC parameter tuning framework [28]. As always in simulation-based optimization, we are confronted with the important trade-off between the exploitation

GECCO '20, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390155

of already acquired knowledge (through adaptive sampling) and the reduction of uncertainty in regions of the search space that are currently not well covered with already evaluated samples. Sampling in the latter regions of high uncertainty – commonly referred to as exploration – can help to identify other promising regions of the search space. In our experiments, we observe indeed that small initial designs are not always preferable. In fact, we even identify cases in which pure (quasi-)random sampling outperforms any of the tested SMBO-based techniques.

We use our huge database also to investigate advantages of long runs vs. restarted ones. That is, we address the question whether one should use the full budget for one long run, or whether two shorter runs of smaller budget are preferable. We identify several cases in which restarts seem preferable, giving another indication that an adaptive design of SMBO techniques could be preferable.

The evaluation and analysis of the dataset (which comprises more than 500 000 experiments) has been particularly challenging, as no clear pattern between the performance of the different designs and the parameters of the problem (such as its dimension, its high-level features, or even its function ID) were observable. Our data suggests that machine-trained algorithm configuration techniques should be able to outperform state-of-the-art SMBO designs by large margins. The appropriateness of the BBOB dataset for finding generalizable patterns has been shown in [4, 33].

Paper Organization. This work is structured as follows. Below, we continue with an overview of related work and give information about the availability of our data. Section 2 details the SMBO approach. In Section 3 we describe our experimental setup including considered benchmark problems, parameter choices and performance measures. Results are presented in Sections 4 to 6. We conclude with final remarks and visions for incorporating the acquired knowledge into improved SMBO approaches.

Related Work. For surveys on Bayesian optimization and, more generally, SMBO approaches we refer the interested reader to the already mentioned surveys [40, 46, 49]. Our work builds on EGO, originally suggested by Jones, Schonlau, and Welch [30]. EGO is characterized by using a flexible Kriging, i.e., a Gaussian process surrogate model which offers a natural uncertainty estimate and the widely used quasi-standard expected improvement (EI) infill criterion which balances exploitation of the model and exploration of uncertain regions of the model [29].

Our key interest is an analysis of the influence of the initial design's size and distribution. We assess four different distributions: uniform sampling, LHS, Halton points, and Sobol' sequences. For each of these designs we test ten different initial sample sizes. Recommendations on which initial design should be favored vary quite significantly within the community, see [2, 41] for a discussion. In terms of design *size*, SMAC [28] makes an extreme choice in that it uses only one randomly sampled initial design point, whereas other commonly found SMBO implementations typically operate with an initial design of size $10 \cdot d$ [30, 41], where d denotes the search space dimension (i.e., the optimization problem can be modeled as a function $f : S \subseteq \mathbb{R}^d \to \mathbb{R}$). In terms of design *distribution*, LHS and uniform sampling are routinely used in SMBO applications, while quasi-random designs, like Halton and Sobol' designs, are less commonly found – despite several indications that their even distribution may be beneficial for maximizing the initial exploration [48].

We next summarize the main works which explicitly address the question how to chose the initial design.

Bartz-Beielstein and Preuss study in [2] suitable initial designs for SPOT [1], an SMBO algorithm specifically designed to perform well on parameter tuning challenges. From experiments on hyper-parameter tuning of evolutionary computation techniques, they conclude that LHS sampling is, in general, to be preferred over uniform sampling. They thereby disagree with statements previously made in [48], which argues that LHS designs do not gain much over uniform sampling, and that quasi-random sampling strategies should be used instead. The recommendation in [48] is, however, to be understood in terms of general design of experiments setting, and not specifically addressing SMBO initialization.

Brockhoff et al. [11] studied the difference between random sampling and LHS designs for Matlab's MATSuMoTo model-based optimizer [42]. In contrast to our work, they fix the total budget of function evaluations to $n = 50 \cdot d$ (whereas we use $n = 2^4, \ldots, 2^9$) and compared only four initial designs: LHS with $2 \cdot (d + 1) \cdot k$ for $k = 1, 2, 10$ and random sampling with $4 \cdot (d + 1)$ points. Results are compared against SMAC [28] and pure random sampling. Their experiments are also across all 24 BBOB functions in $d = 2, 3, 5, 10, 20$ dimensions (we study $d = 2, 3, 4, 5, 10$). Their performance measure is a fixed-target measure, more precisely they study the expected running time (ERT) for target values that are chosen individually for each function and they also compare the anytime performance in terms of ECDF curves. Based on their experiments, Brockhoff et al. conclude that for this setting, no clear advantage of LHS designs can be observed and that large initial samples seem detrimental.

Morar et al. [41] also compare LHS and uniform sampling, but fix the size of the initial design to $2 \cdot d$ and rather focus on the interplay between initial design distribution and the infill criteria used in the adaptive steps of the SMBO framework. They compare performances on two variants of the Branin function, a classic benchmark in SMBO research, and on two parameter tuning problems. They conclude that the total budget has an important influence on the ranking of the different SMBO algorithms. In line with our observations and conclusions, they recommend tuning of the SMBO design if one is likely to see similar types of problems several times.

More recently, Lindauer et al. [36] analyze the sensitivity of Bayesian optimization heuristics w.r.t. its own hyper-parameters. This study, however, puts a much stronger emphasis on the various design choices, and details for the initial sampling strategy are not explicitly mentioned, although Table 3 in their work suggests that this has been varied as well.

Availability of Project Data. While this report highlights a few of our key findings, and demonstrates which statistics are possible to obtain with the data, the full data base offers much more than we can touch upon in a single conference paper. Not only can our data be used to zoom further into the various settings described below, but it also offers additional information about the function value of the best initial design point and of the first point queried in an adaptive fashion, as well as the distance of these points and of the best solution to the optimal solution (in the decision space $[-5, 5]^d$, measured in terms of the L2 norm).

Please note that most of the results reported below are based on median values per (dimension, function, total budget, initial budget ratio, design) combination. This is to avoid correcting factors for the comparison between the Halton designs (for which we have 5 runs for each of the 7 200 considered settings) and the other three designs (for which we have 25 independent runs per setting, i.e., 5 SMBO runs for each of the 5 random samples from the design). Detailed results for each experiment are available in the data base, so that one can easily perform statistical tests, or use other aggregation methods. An interactive evaluation of the data is possible with the very recently released tool HiPlot [25], which essentially produces parallel coordinate plots through which one can easily navigate by zooming and/or highlighting different parts of the data.

The interested reader can find all our project data on [10].

2 SEQUENTIAL MODEL-BASED OPTIMIZATION

In many real-world applications like production engineering, numerical simulations, or hyper-parameter tuning, the objective function f at hand is often of black-box nature. That is, (a) there is little or no knowledge about the structure of f (in particular, we typically do not have derivatives), and (b) function evaluations are expensive in terms of computational and/or monetary resources (days of computation time or actual physical experiments). As a consequence, in the course of problem solving, one tries to keep the number of true function evaluations low. In such settings, sequential model-based optimization (SMBO, [27]) – also known under the term Bayesian optimization[1] – advanced to the state-of-the-art in recent years and is used extensively in many fields of research, e.g., within versatile tools for automatic algorithm configuration [28].

In a nutshell, the key idea of SMBO is as follows: a regression model, i.e., an approximation $\hat{f}$ to the true optimization problem f, is fitted to the evaluated points of an initial design. Subsequently, the model $\hat{f}$ serves as a cheap surrogate for the expensive true objective function and is used to determine the next point(s) worth being evaluated through the actual problem f. These points are determined by optimizing a so-called *infill criterion* (also referred to as *acquisition function*) which keeps balance between exploiting the model (in the sense of striving to high-quality points) and exploring the search-space regions which lack a good model fit (i.e., regions with a high uncertainty about the quality of approximation $\hat{f}$). Note that the optimization of the acquisition function itself is an (often highly multimodal) optimization problem, which is typically solved by state-of-the art solvers such as CMA-ES [24], Nelder-Mead [43], or simply by standard Newton methods, if the surrogate model $\hat{f}$ allows. The key here is that those algorithms now operate on $\hat{f}$ and not on f, which can be evaluated much more efficiently. The points proposed from the optimization of the acquisition function are then evaluated through f and the surrogate $\hat{f}$ is updated to account for the new information. The process is repeated until the available budget of time or function evaluations is depleted.

Jones [30] was the first who used this approach in his Efficient Global Optimization (EGO) algorithm. Therein, Gaussian processes serve as the surrogate and expected improvement (EI) is adopted as infill criterion. Following Jones' seminal contribution, a plethora of extensions were proposed by the community including multi-point proposal [7] and multi-objective SMBO (e.g., [34]) making SMBO a highly flexible framework with many interchangeable components and facets. We refer the interested reader to [27] (and references therein) for a comprehensive overview.

Our study is based on the classical EGO algorithm by Jones.

[1] Originally, Bayesian Optimization only referred to SMBO approaches with Bayesian priors, but nowadays the term is often used to denote the whole class of SMBO methods.

3 EXPERIMENTAL SETUP

Our study investigates the effect of the total budget, the size of the initial design (i.e., the number of evaluations prior to building the first surrogate), and the distribution of this initial design on the quality of the final recommendation made by an off-the-shelf SMBO algorithm. Below, we summarize the benchmark problems and solution strategies (Section 3.1), as well as the performance measures that we used to evaluate the different strategies (Section 3.2).

All our experiments are implemented in the R programming environment [45]. To be more precise: the SMBO framework `mlrMBO` [6] serves as the working horse for our experimental study, the `smoof`-package [8] is used for an interface to the BBOB functions and the interface package `dandy` [9] is used to generate the initial designs. The latter delegates to packages `qrng` [26] and `randtoolbox` [14], which implement quasi-random sequence generators as well as to package `lhs` [13] for the LHS designs.

3.1 Benchmark Problems and Solvers

We use the following setup for our experimental analysis:

- **The objective function f.** As mentioned in the introduction, we focus on the 24 functions from the (noiseless and single-objective) BBOB test suite [22]. An overview of these functions is available in [23]. We consider the first instance of each function, whose d-dimensional variant we denote by $f^{(d)}$. We let F^d be the collection of these 24 functions. We study *minimization* as objective.
- **The problem dimension d.** We consider five different search space dimensions: $d \in D := \{2, 3, 4, 5, 10\}$.
- **Total budget n.** The total number of function evaluations. We consider six different budgets: $n \in N := \{2^x \mid x \in \{4, \ldots, 9\}\}$.
- **Initial design ratio k:** We consider initial designs of size $\lceil k \cdot n \rceil$ with $k \in K := \{0.1, 0.2, \ldots, 1.0\}$.
- **Sampling design s.** We study four different distributions from which the d-dimensional initial design of size $\lceil k \cdot n \rceil$ is sampled:
  - *uniform sampling*: R's default random number generator (Mersenne-Twister [38]) to generate uniform samples.
  - *Latin Hypercube Sampling* (LHS [39]): "improved" LHS design as suggested in [3].
  - *Sobol' sequences* [50]: `randtoolbox` implementation with scrambling as proposed by Owen [44], and Faure & Tezuka [19].
  - *Halton designs* [21] `randtoolbox` implementation with default parameters.

  More detailed definitions, motivations, and applications of these distributions can be found, for example, in [15].
- **Random seed r_i - initial design.** While the Halton point sets are deterministic, the other designs produce random points. To account for this randomness, we sample $R_i = 5$ instances from each of the three random (i.e., non-Halton) designs.

- **Random seed r_A - SMBO randomness.** Finally, to compensate for the randomness of the SMBO algorithm (note that the SMBO process is stochastic itself, e.g., by means of a stochastic procedure used to search the infill-criterion), we do $R_A = 5$ independent runs per each of the settings fixed through the decisions above.

It should be noted that we do not vary the infill criterion (also known as *acquisition function*), nor any other component of the SMBO, but use the default variant of `mlrMBO` v1.1.4 with expected improvement as infill criterion and a Kriging surrogate.

With the notation above, we consider a total number of $|F| \cdot |D| \cdot |N| = 24 \cdot 5 \cdot 6 = 720$ different **problems**, and for each of these problems we consider $|S| \cdot |K| = 4 \cdot 10 = 40$ different solution **strategies**. Here we consider the budget as integral part of a problem, since SMBO algorithms are typically applied when the budget is fixed a priori. We therefore distinguish between the *function* $f^{(d)}$ that is to be optimized, and the *problem* (f, d, n) of minimizing $f^{(d)}$ with a given budget n.

As mentioned above, on each problem we perform 5 runs of each strategy which is based on Halton designs and we perform 25 runs for all other strategies. Our total number of experiments is thus

$$\begin{aligned} &|F| \cdot |D| \cdot |N| \cdot |K| \cdot (1 + (|S| - 1) \cdot R_i)) \cdot R_A \\ =&24 \cdot 5 \cdot 6 \cdot 10 \cdot (1 + (3 \cdot 5)) \cdot 5 = \underline{576\,000}. \end{aligned}$$

Not all of these runs terminated successfully, due to problems with the Kriging implementation used by `mlrMBO`. The problems occur in particular with high total budget and low initial design ratio. Here, the Kriging-routine obviously runs into problems when many points are sampled close to each other as it often is the case when SMBO runs converge into a (local) optimum. While for each $n \leq 128$ there are at least 99.8% successful runs this number reduces to 94% for $n = 256$ and 85.4% for $n = 512$. In total, we had 555 598 (96.5%) successful runs. In all computations below we only consider (f, d, n, k, s) combinations for which at least three runs terminated successfully, i.e., provided their recommendation.

3.2 Performance Measures and VBS

For each of our experiments we record the value of the best solution that has been evaluated during the entire run. We denote this value by $f(d, n, k, s, r_i, r_A)$. Since the BBOB functions have quite diverse ranges of function values, we do not study these function values directly, but rather follow standard practice in BBOB studies and focus on the *target precision*, i.e., the gap to the global optimum,

$$p(f, d, n, k, s, r_i, r_A) := f(d, n, k, s, r_i, r_A) - \inf f^{(d)}.$$

As mentioned above, we will restrict most of our analyses to the median performance of each strategy on each problem. Our main performance criteria is therefore

$$M(f, d, n, k, s) = \mathbb{M}\left(\{p(f, d, n, k, s, r_i, r_A) \mid r_i \in R_i(s), r_A \in [5]\}\right),$$

where $\mathbb{M}$ denotes the median and where we use the convention that $R_i(\text{Halton}) = \{1\}$ and $R_i(s) = \{1, 2, \ldots, 5\} =: [5]$ for the other sampling designs $s \in S \setminus \{\text{Halton}\}$.

Virtual Best Solver and Relative Target Precision. An important concept in comparing portfolios of algorithms is the *virtual best solver* (VBS). This VBS describes a hypothetical algorithm that for each problem (i.e., each (f, d, n) combination in our case) selects an algorithm A from a given portfolio $\mathcal{A}$ that achieves the best performance [32]. In our case, the algorithm portfolio is the collection of all 40 (k, s) combinations. As we consider median performance, the VBS is defined by selecting for each problem (f, d, n) the strategy (k, s) that achieved the best median function value. For notational convenience, we omit the explicit mention of the median and set

$$\text{VBS}(f, d, n) := \min\{M(f, d, n, k, s) \mid s \in S, k \in K\}.$$

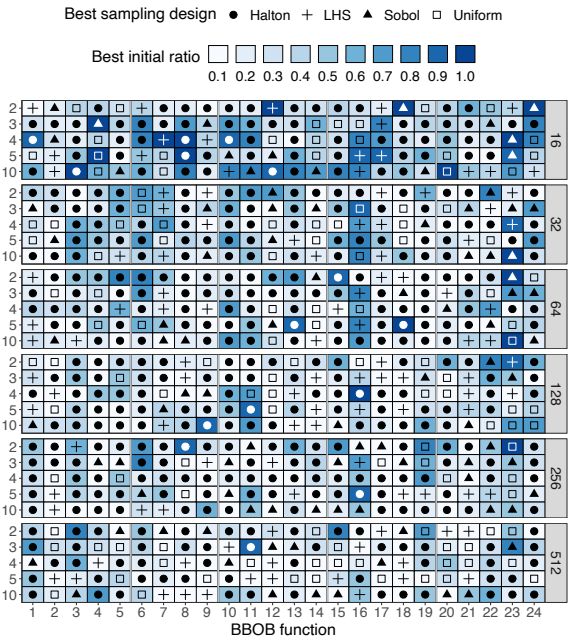

Figure 1: Overview of the virtual best solver (VBS), i.e., the strategy (k, s) that achieved the best median performance on the respective problem (f, d, n).

Fig. 1 shows which strategy defined the VBS for which problem(s). A first visual interpretation suggests that this data is relatively unstructured; we will come back to this point further below.

By design, some of the BBOB functions are much "harder" than others, so that we see substantial differences in the target precision that can be achieved with a fixed budget n. To compensate for that in our aggregations, we will frequently study the relative performance of a strategy (k, s) compared to the VBS. To this end, we set

$$\mathcal{R}(f, d, n, k, s) := M(f, d, n, k, s)/\text{VBS}(f, d, n)$$

and refer to $\mathcal{R}(f, d, n, k, s)$ as the *relative target precision* of strategy (k, s) on problem (f, d, n). Note that these values are at least one, where $\mathcal{R}(f, d, n, k, s) = 1$ implies that strategy (k, s) achieved the best median target precision among all the 40 different strategies.

4 AGGREGATED RESULTS

As shown in Fig. 1, it is not possible to derive simple rules that define which strategy achieves the best performance on each of the BBOB functions. In Fig. 2 we therefore count how often each strategy forms the VBS. Therein, we observe a clear advantage for Halton designs (it has the most "hits" for any given initial ratio except for $k = 100\%$), and we further observe a clear tendency towards small initial ratios. However, we also see that each strategy "wins" at least

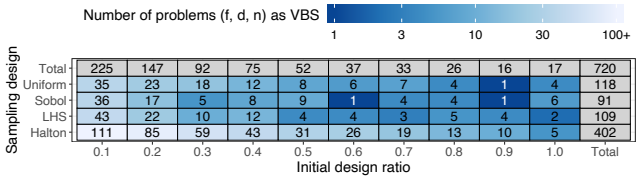

Figure 2: Number of problems (f, d, n) for which the respective strategy forms the VBS.

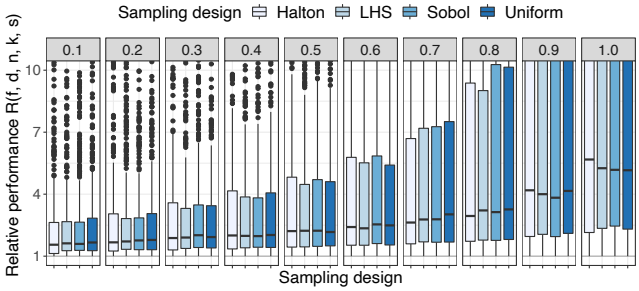

Figure 3: Boxplots of relative performances $\mathcal{R}(f, d, n, k, s)$ across all 600 problems (f, d, n) with budget $n \leq 256$, shown for all 40 different strategies (k, s). The y-axis is capped at 10.

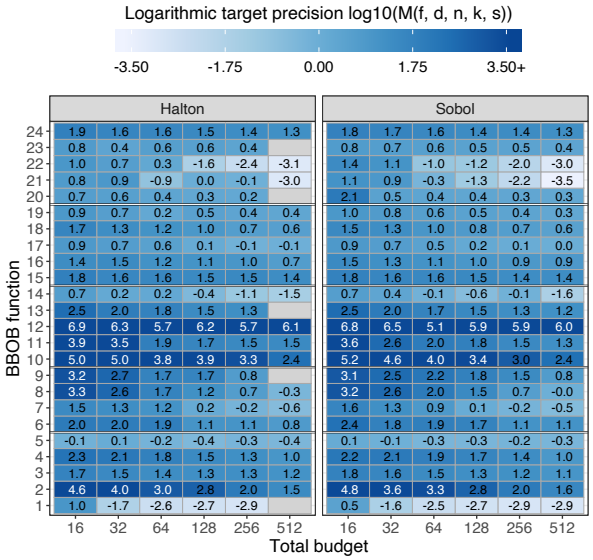

Figure 4: Logarithmic median target precision $\log_{10}(M(f, d, n, k, s))$ depending on the total budget. Results are shown for Halton (left) and Sobol (right) designs with an initial budget of 10% of the total budget and across all 5-dimensional BBOB functions. Gray boxes are due to missing data (less than 3 successful runs, see Section 3.1).

one problem. Neither the simple counting statistics in Fig. 2 nor the more detailed overview in Fig. 1 provide any information about the *magnitude* of the advantage. We thus plot in Fig. 3 the distribution of the relative target precision $\mathcal{R}(f, d, n, k, s)$ of each strategy (k, s), aggregated again over all 720 problems. This plot confirms the tendency that spending a larger ratio of the total budget on generating the initial design results in worse overall performance. We also observe that although the **Halton design generated with k = 10% of the total budget has the best median performance**, the actual differences between the four designs are rather small.

A more detailed picture about the relative performances is provided in Fig. 5. Here, we plot the median (over all 24 BBOB functions) relative performance; i.e., the value in each cell represents $\mathbb{M}(\{\mathcal{R}(f, d, n, k, s) \mid f \in \{1, 2, ..., 24\}\})$ for the given dimension, budget, and strategy. We observe that in most cases the **performance worsens with increasing initial budget ratio k**, and this consistently for each problem dimension d and total budget n.

The values in the rows labeled "Total" are the median values over all budgets (last row per dimension) and dimensions (bottom-most rows), respectively. Noticeably, the **influence of the sampling design vanishes with increasing dimension** – independent from the budget ratio. Aggregated over all dimensions, the differences between the designs are small, as already observed in Fig. 3.

Remember that the values in Fig. 5 are always scaled by the VBS that is specific for problem (f, d, n), but independent from strategy (k, s). This implies that the rows are computed against the same VBS, but different rows compare against different strategies. Values in different rows should therefore only be compared with care.

5 PERFORMANCE BY FUNCTION

After having studied values that were aggregated across all 24 BBOB functions (see Section 4), we now take a closer look at the differences between the different strategies on each of the functions.

Influence of the Total Budget. Fig. 4 reports the median target precision (shown on a log-scale) achieved by Halton and Sobol' designs with $k = 10\%$ initial budget, in dependence of function f and total budget n. The plot reveals the functions that are easy (e.g., functions 1, 21, 22) and difficult (functions 10 and 12) for SMBO. Note that the performance convergence is not always monotonically decreasing with increasing total budget size. This might result from the small number of repetitions (5 for the Halton design, 25 for Sobol'). However, the differences are fairly small. Fig. 6 extends Fig. 4 to all 40 strategies (k, s). That is, for each 5-dimensional problem $(f, 5, n)$ a heatmap of the relative performances $\mathcal{R}(f, d, n, k, s)$ is shown for all pairs of sampling design s and initial design ratio k. We observe that, in particular for functions 15, 19, 23 and 24, the differences between the different initial budgets are comparatively small. This likely results from the functions' highly multimodal landscapes, which hinder SMBO from training reasonable surrogates.

Influence of the initial sample size k and design s. Fig. 7 shows the relative median target precision $\mathcal{R}(f, d, n, k, s)$ for all 24 BBOB functions, for a fixed budget of 128 function evaluations and variable dimension (columns) and strategies (rows). We recall that the VBS is defined per column, i.e., each column has at least one strategy with $\mathcal{R}(f, d, n, k, s) = 1$ (see Fig. 1).

We observe that the benefit of small initial budgets is important for functions with at most medium-sized indices. This finding is very plausible, as the first 14 functions mainly are separable and/or unimodal – i.e., functions whose structure can be well exploited by SMBO. However, **for the group of multimodal functions** (IDs 15 to 24), with the notable exception of functions 21 and 22, the

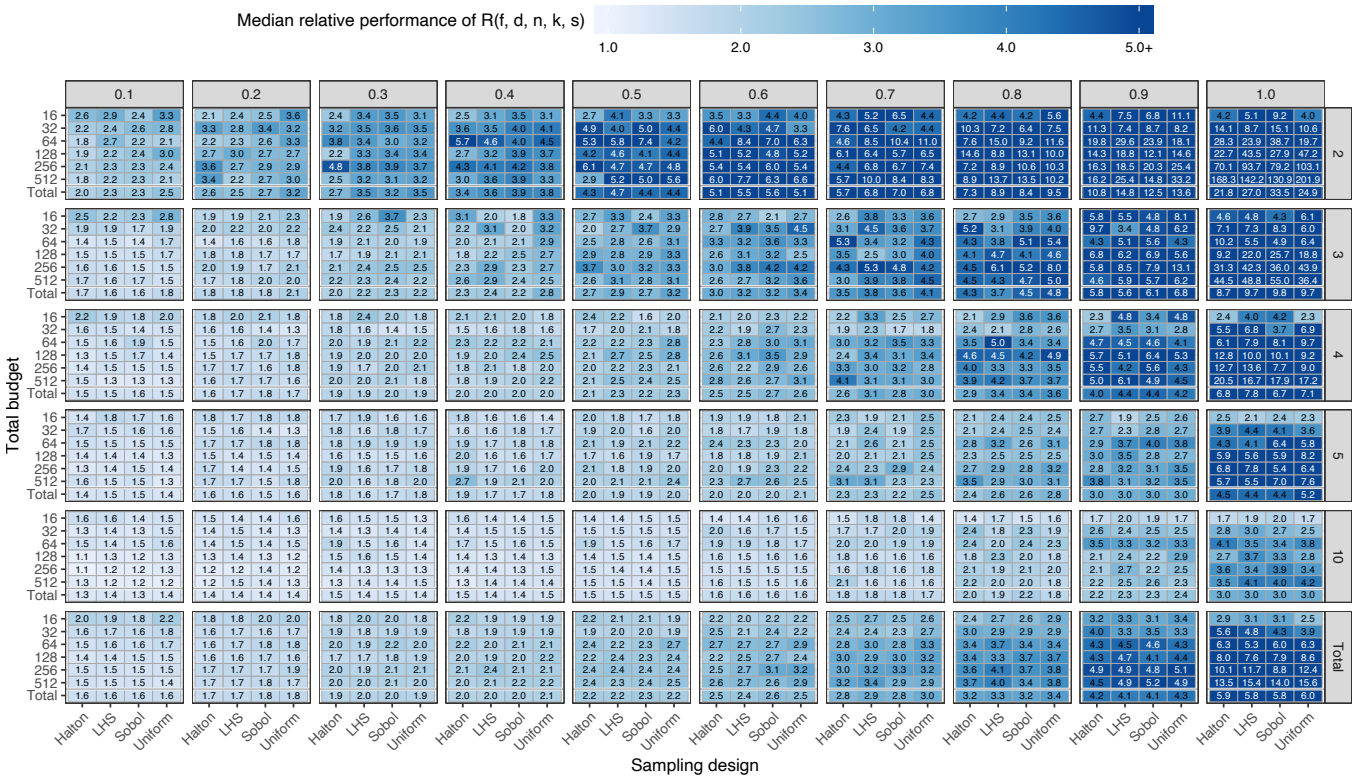

Figure 5: Median (over all 24 BBOB functions) relative performance of $\mathcal{R}(f, d, n, k, s)$, by dimension and budget (rows) and strategy (k, s) (columns).

differences between the different initial ratios are rather small, indicating that **SMBO does not perform much better than (quasi-) random sampling** in the initial phases of the optimization process.

We also see interesting cases in which larger ratios of initial budget result even in better performance than small initial ratios. An extreme case is function 12 in dimension $d = 2$. Its situation is as follows. The VBS is defined by the (30%, Halton) strategy. The differences between the Halton designs with $k > 30\%$ are rather small, whereas for the other strategies smaller initial budgets are preferable. By studying the absolute values in more detail, we find that the Halton strategy identifies a point with absolute target precision 24.9 when $k \geq 30\%$. SMBO does not manage to find a better point in any of its $128 - \lceil 30\% \cdot 128 \rceil = 89$ adaptive evaluations. The best median target precision of any of the other strategies has target precision 58.6 – achieved by the (10%, LHS) strategy. Looking further into the results of the 800 individual runs, we find that 126 of them find a point of target precision smaller than 24.9. The distribution of their initial ratios is not unanimous, as can be see in the following table, which counts how often each initial ratio k appears among these 126 runs. These results show how difficult it is to give a general advice for the optimization of this function – even when the budget is fixed and the function ID known.

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
#	14	12	8	13	21	7	12	14	10	15

6 RESTARTS VS. LONG RUNS

In the previous paragraph, we have started to look into the distribution of the target precisions. We now demonstrate how such information can be used to study whether it is beneficial to use the total budget of n function evaluations for a single long run, or whether one should rather start two shorter runs of budget $n/2$ each, or four runs of budget $n/4$, etc.

Distribution of the Target Precisions. Crucial for the consideration of restarts are the distributions of the function values (or, equivalently, the distributions of the target precisions) achieved by the different strategies (k, s). For reasons of space, we cannot go in much detail here, but Fig. 8 demonstrates how these boxplots look like. Note that this figure is for one specific combination of function ($f = 17$), dimension ($d = 5$) and budget ($n = 128$). It aggregates the target precision of all 40 strategies, i.e., of 800 runs in total. Our data base contains one such plot for each of the 720 (f, d, n) problems.

Note that the dispersion of Halton designs are smaller, but this is due to the fact that we do not perform resampling for this sequence. For all pairs of (k,Halton) strategies with $k \geq 50\%$ the target precision of the best initial design point is slightly above 3. For $k \geq 80\%$ none of the SMBO runs starting in this best initial design point finds a solution of better target precision. For $k \in \{60\%, 70\%\}$, only one of the five runs each finds a better solution. Note that the length

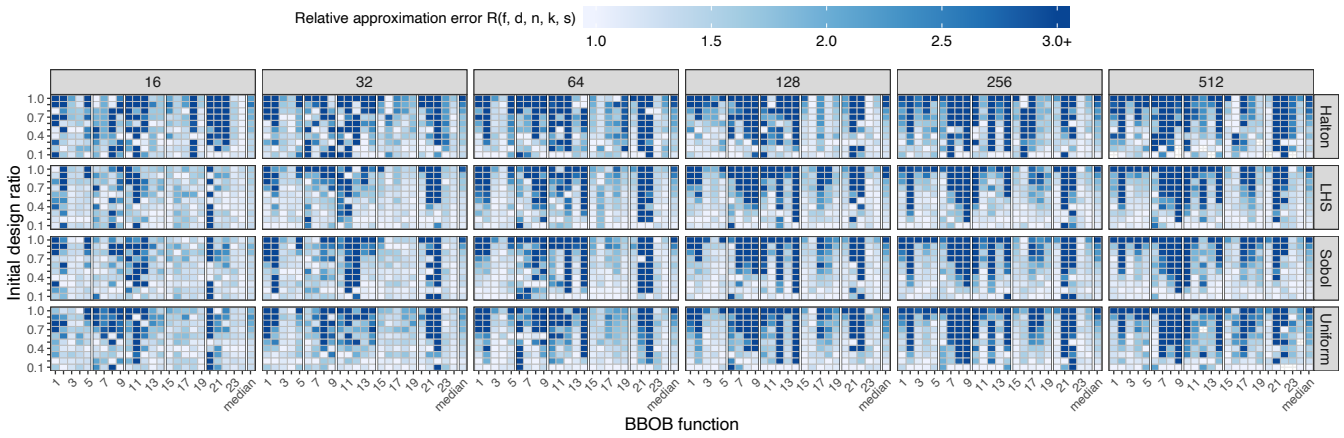

Figure 6: Heatmap visualization of relative performances $\mathcal{R}(f, d, n, k, s)$ by function, total budget, and strategy (k, s) for fixed dimension $d = 5$. Values are capped at 3.

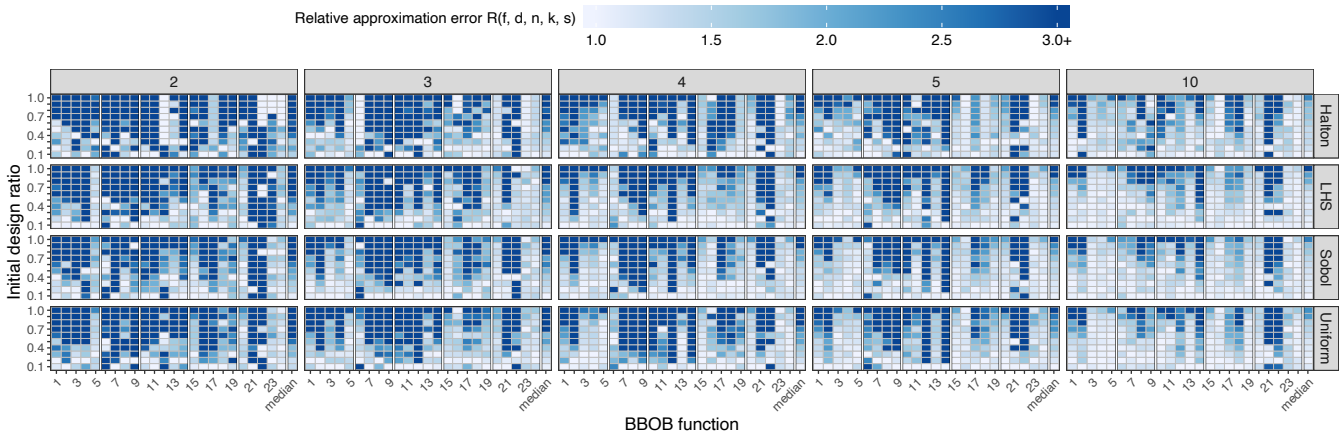

Figure 7: Heatmap visualization of the relative performance $\mathcal{R}(f, d, n, k, s)$ by dimension, function, and design type for a fixed total budget of $n = 128$ function evaluations. Values are capped at 3.

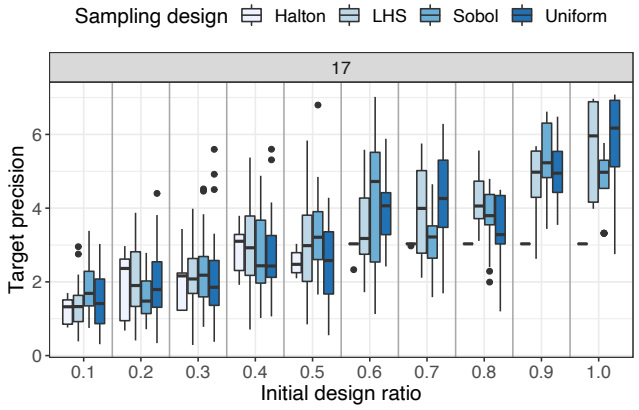

Figure 8: Boxplots of the target precisions $p(17, 5, 128, k, s, r_i, r_A)$-values for function $f = 17$, dimension 5, and total budget $n = 128$, grouped by initial budget ratio k and design s.

of each of these SMBO runs is $n - \lceil k \cdot n \rceil$, which for $k = 0.6$ corresponds to 51 adaptive SMBO steps. Such detailed information could be very **useful to identify weaknesses of the EGO approach**, and, hopefully, contribute towards better SMBO designs.

Computing median target precision of restarting SMBO. To investigate if, for a given problem (f, d, n), a restart strategy is beneficial over a single long run, we need to extend our previous focus on median target precision to different percentiles. To this end, let

$$P_q(f, d, n, k, s) := P_q\left(\{p(f, d, n, k, s, r_i, r_A) \mid r_i \in R_i(s), r_A \in [5]\}\right),$$

the q-th percentile of the target precisions achieved by strategy (k, s) on problem (f, d, n) across all 5 (Halton) or 25 (Sobol', LHS, uniform designs) runs, respectively. For a fair comparison of one run of the full budget n with two runs of budget $n/2$ (of the same strategy), we compare the median $M(f, d, n, k, s)$ (i.e., the 50-th percentile) with the $q := 1 - \sqrt{1/2}$-th percentile $P_q(f, d, n/2, k, s)$. With this value of q, the probability that (at least) one of the two runs achieves a target precision that is at least as good as $P_q(f, d, n/2, k, s)$ equals $1 - (1 - q)^2 = 1/2$. This is identical to the probability that one

long run achieves a target precision that is at least as good as $M(f, d, n, k, s)$. Note that we disregard a small bias in our data, which results from the fact that we do not have 25 completely independent runs. Instead, we use the same initial design sample for five independent SMBO runs each – but, we ignore this effect in the following computations. Also, given the small number of runs, all numbers should be taken with care – the smaller the percentile, the larger the uncertainty around the values. We nevertheless show this example to demonstrate how one could systematically address the question how to split a given budget into possibly parallel runs.

Fig. 9 illustrates an example for the relevant percentiles when comparing one long run of budget n with two short ones of budget $n/2$, and four even shorter ones of budget $n/4$. More precisely, we fix in this figure the strategy to (10%,LHS) and the dimension to $d = 5$, and we show log-scaled relative data. Each 3×6 box corresponds to one of the 24 BBOB functions. As we scale the values within a box by its VBS, and afterwards show the percentile ratios on a log-scale, the field with value 0.0 represents the combination achieving the best target precision (i.e., the VBS) among the displayed combinations. Not surprisingly, for most functions this is the $(1 - \sqrt[4]{1/2})$-th percentile of the full budget $n = 512$. Let P^*_f be the target precision of this (percentile, budget) combination for a given function f. A value φ in field (q, n) is then to be read as follows: the target precision $P_q(n) := P_q(f, d = 5, n, 10\%, LHS)$ satisfies $P_q(n) = 10^{\varphi} \cdot P^*_f$. Smaller values are therefore better. We see that for $f = 5$,for example, our data suggests that a total budget of 512 evaluations (value 0.4 when used as single run) is better used for four runs of budget 128 each (value 0.1). We have marked in this matrix all fields for which the long run compares unfavorably with a restart strategy – the one corresponding to the neighboring field on the lower left diagonal. Overall, we see that several such cases exist, which confirms our previous finding that **EGO does not always compare favorably against quasi-random sampling**.

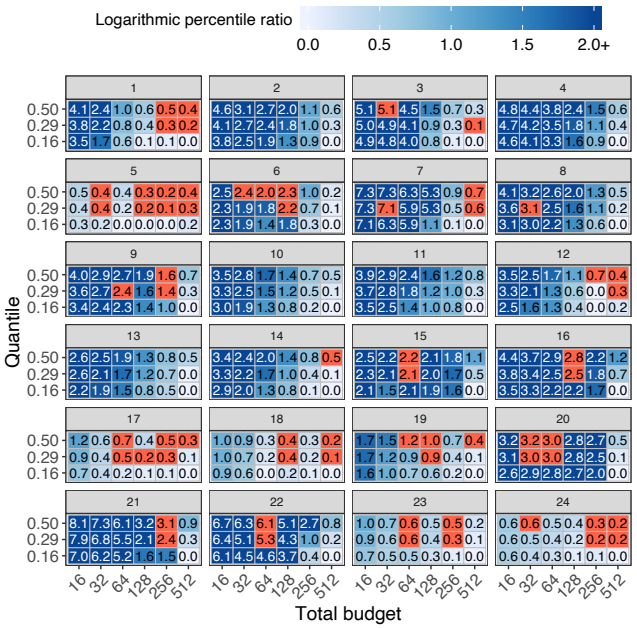

Figure 9: Percentiles $P_q(f, d, n, k, s)$ of target precisions across the 25 SMBO runs per function and dimension using an LHS design with 10% initial budget and for the 2-dimensional problems. The percentiles are scaled by the respective function's best percentile, and the resulting ratios are shown on a capped log10-scale. Red boxes indicate that the corresponding strategy performs unfavorably against a restart strategy (the one to the lower left).

7 CONCLUSIONS

In this paper we have presented a database for data-driven investigations of the sequential model-based optimization (SMBO) strategy EGO [30]. The focus of our work is on analyzing the influence of the (size and type of) initial design on the overall performance of EGO. Our data base contains data for 720 different problems, which are evaluated against a total of 40 different initial design strategies.

While we clearly observed that small initial designs are preferable at a high-level view, we also found that each of the 40 considered combinations of design type and size achieved best performance on at least one of the 720 problems. Our findings thus confirm that an automated strategy selection method – like the proof-of-concept approach presented in [47] – might indeed be profitable. Moreover, we even identified cases in which the usage of EGO does not provide any benefits over the initial (quasi-)random sample – especially in case of highly multimodal problems.

Our long-term vision are SMBO approaches that dynamically decide whether to take the next sample from a (quasi-)random distribution or whether to derive it from the surrogate model. Going one step further, we believe that an adaptive choice of the acquisition function, and possibly even of the solver used to optimize the latter, should bring substantial performance gains – in particular in the case in which the total budget is known in advance. Hence, we need to "train" a final recommendation (last evaluation) instead of achieving good anytime performance. These two mentioned questions fall under the umbrella of dynamic algorithm configuration, which has been an important driver for the field of evolutionary computation for the last decades [12, 16, 17, 31], and which has recently also gained interest in machine learning communities [5].

Typically, the budget of common SMBO applications is too small for a classical a priori (i.e., offline) landscape-aware selection of the optimizer design based on supervised learning approaches (see [32] for a survey). However, in case high-level properties – such as the degree of (multi-)modality or the sizes of the problem's attraction basins – are known for the problem at hand, or can be guessed by an expert, selecting a suitable initial design strategy is feasible.

Finally, we have seen that the performance of the different designs was often quite comparable. To investigate the differences in more detail, we suggest to consider the different strategies as a portfolio of different algorithms. With this viewpoint, one could analyze the marginal contributions [51] or Shapley values [20] of the different designs, and leverage the information contained therein.

ACKNOWLEDGMENTS

This work was supported by the Paris Ile-de-France Region and the *European Research Center for Information Systems (ERCIS)*.

REFERENCES

[1] Thomas Bartz-Beielstein. 2010. SPOT: An R Package For Automatic and Interactive Tuning of Optimization Algorithms by Sequential Parameter Optimization. *CoRR* abs/1006.4645 (2010). arXiv:1006.4645 http://arxiv.org/abs/1006.4645

[2] Thomas Bartz-Beielstein and Mike Preuss. 2006. Considerations of Budget Allocation for Sequential Parameter Optimization (SPO). In *Proc. Workshop on Empirical Methods for the Analysis of Algorithms (EMAA'06)*. 35–40.

[3] Brian Beachkofski and Ramana Grandhi. 2002. Improved Distributed Hypercube Sampling. In *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics.

[4] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. 2017. Per instance algorithm configuration of CMA-ES with limited budget. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*. ACM, 681–688.

[5] André Biedenkapp, H. Furkan Bozkurt, Frank Hutter, and Marius Lindauer. 2019. Towards White-box Benchmarks for Algorithm Control. *CoRR* abs/1906.07644 (2019). arXiv:1906.07644 http://arxiv.org/abs/1906.07644

[6] Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. 2016. mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. (2016). arXiv:stat/1703.03373 http://arxiv.org/abs/1703.03373

[7] Bernd Bischl, Simon Wessing, Nadja Bauer, Klaus Friedrichs, and Claus Weihs. 2014. MOI-MBO: Multiobjective Infill for Parallel Model-Based Optimization. In *Learning and Intelligent Optimization*, Panos M. Pardalos, Mauricio G.C. Resende, Chrysafis Vogiatzis, and Jose L. Walteros (Eds.). Springer International Publishing, Cham, 173–186.

[8] Jakob Bossek. 2017. smoof: Single-and Multi-Objective Optimization Test Functions. *The R Journal* 9, 1 (2017), 103–113. https://journal.r-project.org/archive/2017/RJ-2017-004/RJ-2017-004.pdf

[9] Jakob Bossek. 2020. *dandy: Designs and Discrepancy*. https://github.com/jakobbossek/dandy R package version 1.0.0.0000.

[10] Jakob Bossek. 2020. Public data repository with project data. https://github.com/jakobbossek/GECCO2020-smboinitial

[11] Dimo Brockhoff, Bernd Bischl, and Tobias Wagner. 2015. The Impact of Initial Designs on the Performance of MATSuMoTo on the Noiseless BBOB-2015 Testbed: A Preliminary Study. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*. ACM, 1159–1166.

[12] Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. 2013. Hyper-heuristics: a survey of the state of the art. *JORS* 64, 12 (2013), 1695–1724.

[13] Rob Carnell. 2019. *lhs: Latin Hypercube Samples*. https://CRAN.R-project.org/package=lhs R package version 1.0.1.

[14] Dutang Christophe and Savicky Petr. 2019. *randtoolbox: Generating and Testing Random Numbers*. R package version 1.30.0.

[15] Josef Dick and Friedrich Pillichshammer. 2010. *Digital Nets and Sequences*. Cambridge University Press.

[16] Benjamin Doerr and Carola Doerr. 2020. Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, 271–321.

[17] Ágoston Endre Eiben, Robert Hinterding, and Zbigniew Michalewicz. 1999. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 3 (1999), 124–141.

[18] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *ICML*. 1436–1445.

[19] Henri Faure and Shu Tezuka. 2002. Another Random Scrambling of Digital (t,s)-Sequences. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, Kai-Tai Fang, Harald Niederreiter, and Fred J. Hickernell (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 242–256.

[20] Alexandre Fréchette, Lars Kotthoff, Tomasz P. Michalak, Talal Rahwan, Holger H. Hoos, and Kevin Leyton-Brown. 2016. Using the Shapley Value to Analyze Algorithm Portfolios. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. AAAI, 3397–3403.

[21] John H. Halton. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2 (1960), 84–90.

[22] Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2016. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* arXiv:1603.08785 (2016).

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

[24] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Computation* 9, 2 (2001), 159–195.

[25] Daniel Haziza, Jérémy Rapin, and Gabriel Synnaeve. 2020. HiPlot - High dimensional Interactive Plotting. https://github.com/facebookresearch/hiplot.

[26] Marius Hofert and Christiane Lemieux. 2019. *qrng: (Randomized) Quasi-Random Number Generators*. https://CRAN.R-project.org/package=qrng R package version 0.0-7.

[27] Daniel Horn, Tobias Wagner, Dirk Biermann, Claus Weihs, and Bernd Bischl. 2015. Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark. In *Evolutionary Multi-Criterion Optimization*. Springer International Publishing, Cham, 64–78.

[28] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *LION*. Springer, 507–523.

[29] Donald R. Jones. 2001. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization* 21, 4 (01 Dec 2001), 345–383.

[30] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13, 4 (1998), 455–492.

[31] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston Endre Eiben. 2015. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation* 19 (2015), 167–187.

[32] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. 2019. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation* 27, 1 (2019), 3–45.

[33] Pascal Kerschke and Heike Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation (ECJ)* 27, 1 (2019), 99 – 127.

[34] Joshua Knowles. 2006. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 10 (2006), 50–66.

[35] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. 2019. Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA. In *Automated Machine Learning - Methods, Systems, Challenges*. Springer, 81–95.

[36] Marius Lindauer, Matthias Feurer, Katharina Eggensperger, André Biedenkapp, and Frank Hutter. 2019. Towards Assessing the Impact of Bayesian Optimization's Own Hyperparameters. In *IJCAI 2019 DSO Workshop*.

[37] Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz (Eds.). 2007. *Parameter Setting in Evolutionary Algorithms*. Studies in Computational Intelligence, Vol. 54. Springer.

[38] Makoto Matsumoto and Takuji Nishimura. 1998. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Trans. Model. Comput. Simul.* 8, 1 (Jan. 1998), 3–30.

[39] Michael D. McKay, Richard J. Beckman, and William J. Conover. 1979. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* 21 (1979), 239–245.

[40] Jonas Mockus (Ed.). 1989. *Bayesian Approach to Global Optimization*. Springer.

[41] Marius Tudor Morar, Joshua Knowles, and Sandra Sampaio. 2017. Initialization of Bayesian Optimization Viewed as Part of a Larger Algorithm Portfolio. In *Proc. of the international workshop in Data Science meets Optimization (DSO at CEC and CPAIOR 2017)*.

[42] Juliane Mueller. 2014. MATSuMoTo: The MATLAB Surrogate Model Toolbox For Computationally Expensive Black-Box Global Optimization Problems. arXiv:math.OC/1404.4261

[43] John Ashworth Nelder and Roger Mead. 1965. A Simplex Method for Function Minimization. *Comput. J.* 7 (1965), 308–313.

[44] Art B. Owen. 1995. Randomly Permuted (t,m,s)-Nets and (t, s)-Sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, Harald Niederreiter and Peter Jau-Shyong Shiue (Eds.). Springer New York, New York, NY, 299–317.

[45] R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[46] Carl Edward Rasmussen and Christopher K. I. Williams (Eds.). 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

[47] Bhupinder Singh Saini, Manuel López-Ibáñez, and Kaisa Miettinen. 2019. Automatic Surrogate Modelling Technique Selection Based on Features of Optimization Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Companion*. ACM, 1765 – 1772.

[48] T.J. Santner, B.J. Williams, and W.I. Notz. 2003. *The Design and Analysis of Computer Experiments*. Springer.

[49] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175.

[50] Ilya Meyerovich Sobol. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *U. S. S. R. Comput. Math. and Math. Phys.* 7, 4 (Jan. 1967), 86–112.

[51] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2012. Evaluating Component Solver Contributions to Portfolio-Based Algorithm Selectors. In *Proc. of Theory and Applications of Satisfiability Testing (SAT'12) (Lecture Notes in Computer Science)*, Vol. 7317. Springer, 228–241.