Analysis of the Bayesian Multi-Scale Optimistic Optimization on the CEC2016 and BBOB Testbeds

Abdullah Al-Dujaili
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: aldujail001@e.ntu.edu.sg

S. Suresh
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: ssundaram@ntu.edu.sg

***Abstract*—This paper provides an empirical analysis of the recently proposed Bayesian Multi-Scale Optimistic Optimization (BaMSOO) algorithm for solving bound-constrained black-box global optimization problems under expensive as well as cheap budgets of function evaluations. The CEC2016 and BBOB benchmarks are used in assessing the algorithm. We also compare BaMSOO with the Simultaneous Optimistic Optimization (SOO) algorithm from which BaMSOO is derived. The results indicate that BaMSOO has a comparable performance with SOO under expensive-budget settings but outperforms it with increasing evaluation budgets. Finally, we provide useful insights regarding the algorithm's relative efficiency and future directions.**

***Index Terms*—BaMSOO, SOO, Benchmarking, black-box optimization, cheap budget**

I. Introduction

Global optimization attempts to find the best solution from a set of solutions for an optimization problem. It has applications in various science and engineering disciplines. Some of these problems are exceptionally difficult to solve exactly, due to the absence of objective function information [1], where the sole source of information about the objective function is available through point-wise evaluation (also known as black-box optimization). In this paper, we are concerned with the solution of bound-constrained black-box deterministic problems of the form

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad x \in \mathcal{X},\ \mathcal{X} = \{\mathbf{x} \in \mathbb{R}^D : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\} \end{aligned} \tag{1}$$

Evaluating $f : \mathcal{X} \to \mathbb{R}$ is typically expensive requiring some computational resources (e.g., time, power, money). More specifically, we are asked to solve (1) using a computational budget of n function evaluations. Several algorithms/techniques have been proposed and studied to solve such problems. With this context, these algorithms are assessed in one of two ways, viz. theoretical, and empirical analysis. In theoretical analysis, a principled methodology is carried out to derive an analytical bound of the (run-time) solution quality. After n evaluations/steps, the quality of the returned solution $\mathbf{x}(n)$ is evaluated by the loss/regret measure:

$$r_n = f(\mathbf{x}(n)) - \inf_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \tag{2}$$

Alternatively, empirical analysis employs experimental simulations of the algorithm on complex problems, gaining an insight on the algorithm's practicality/applicability on real-world problems. With this regard, we are interested in empirically assessing the recently proposed Bayesian Multi-Scale Optimistic Optimization (BaMSOO) algorithm for solving (1) originally introduced in [2].

The foundations of BaMSOO can be found in the machine learning field, combining *Bayesian optimization* (BO) [3] and *optimistic optimization* (OO) [4] techniques in addressing the exploration-vs.-exploitation dilemma in search for the optimum solution. It treats (1) within the framework of sequential decision making of an agent interacting with its environment in a way similar to the *multi-armed bandit problem* [5]. With this context, at step t, the algorithm (or the *agent*) chooses the next candidate solution (*action*) to be sampled $\mathbf{x}_t$ based on previous solutions (*history of actions*) and their corresponding function evaluations (*rewards observed so far*), $\mathcal{D}_{t-1} = \{(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_{t-1}, f(\mathbf{x}_{t-1}))\}$. BaMSOO is a derivative-free deterministic global optimizer, which comes with a proved finite-time analysis, bounding the regret (2) under a very weak assumption about the smoothness of the objective function f.

In order to complement BaMSOO's purely theoretical perspective and demonstrate its effectiveness on complex problems, we aim, in this paper, at conducting an empirical analysis of BaMSOO. We have adopted the CEC2016 and the BBOB testbeds for the analysis. The CEC2016 benchmark [6] has 15 scalable functions; among them are composite and hybrid functions. On the other hand, The BBOB benchmark [7] comes with a testbed of 24 scalable functions. Both sets of functions address such real-world difficulties as ill-conditioning, multi-modality, and dimensionality. Nevertheless, since CEC2016 involves computationally-expensive functions, smaller evaluation budget (less number of function evaluations) are allocated to CEC2016 than to BBOB testbed simulating practical scenarios. Furthermore, to validate its effectiveness as an extension of OO, BaMSOO is compared with the Simultaneous Optimistic Optimization (SOO) [4] algorithm from which BaMSOO is originally developed.

The rest of the paper is organized as follows. Section II provides a brief description of BaMSOO. In Section III, the numerical assessment of the algorithm and its comparison with SOO is presented on the CEC2016 and BBOB testbeds;

discussing the experimental setup, the procedure for evaluating the algorithms' performance, and elaborating the results. Section IV summarizes the main points from this study.

II. Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

Background: When solving (1), BO treats the deterministic function f as a random function and places a prior (e.g., a Gaussian process prior [8]) over it. The aim of the prior is to capture our beliefs about f's behavior from the gathered data $\mathcal{D}_{t-1}$. This builds a posterior distribution on f, on which an auxiliary function (e.g., the probability of improvement, expected improvement, and Bayesian expected loss functions [9]) is constructed. Consequently, the next sample $\mathbf{x}_t$ is determined via optimizing the auxiliary function. The main limitations of BO is that the auxiliary function optimization can be *very hard and computationally complex*. For more details on BO, refer to [3].

On the other hand, OO methods emerge as an efficient alternative to BO. These methods follow the *optimism in the face of uncertainty* principle [5], i.e., choosing the most profitable action in all possible environments that are compatible with the observations $\mathcal{D}_{t-1}$. Instead of optimizing an auxiliary function, OO algorithms dynamically construct search-space-partitioning trees that systematically sample the search space $\mathcal{X}$. Based on a fitness measure (e.g., function value, upper bound), leaf nodes that may contain the optimum are expanded (e.g., [4], [10]). In practical problems where the prior knowledge is available, it is claimed that OO may not be as competitive as BO due to the lack of a posterior [2].

BaMSOO, based on the OO algorithm, Simultaneous Optimistic Optimization (SOO) [4], was introduced by Wang *et al.* [2] to address the limitations of BO and OO, eliminating the need for auxiliary function optimization (within the context of BO), as well as the need for consistent systematic sampling of $\mathcal{X}$ (within the context of OO). BaMSOO enjoys a finite-time convergence rate under a weak assumption that the objective function is locally smooth with respect to (at least) one of the global optima. Similar to optimistic algorithms [4], [11]; BaMSOO does not require the knowledge of the function smoothness.

The Algorithm: BaMSOO can be regarded as a divide-and-conquer k-ary tree search algorithm that iteratively constructs finer and finer partitions of the search spcae $\mathcal{X}$ at multiple scales $h \geq 0$ in looking for the optimum solution. Given a scale $h \geq 0$ and a partition factor $k \geq 2$, $\mathcal{X}$ can be partitioned into a set of k^h cells/hyperrectangle $\mathcal{X}_{h,i}$ such that $\cup_{i\in\{0,\dots,k^h-1\}}\mathcal{X}_{h,i} = \mathcal{X}$. These cells are represented by nodes of a k-ary tree $\mathcal{T}$. A node (h,i) represents the cell $\mathcal{X}_{h,i}$. The set of leaves in $\mathcal{T}$ is denoted as $L \subseteq \mathcal{T}$. Moreover, each node is associated with a representative state $\mathbf{x}_{h,i} \in \mathcal{X}_{h,i}$ at which f may be evaluated as a part of the sequential design framework and out of the n-evaluation budget.

As summarized in Algorithm 1, BaMSOO expands several leaves simultaneously. At each round, it expands at most one leaf per scale. To define a trade-off on the exploration-vs.-exploitation dilemma, the algorithm takes as input a function $n \to h_{\max}(n)$ which limits the maximum height/scale at which nodes can be expanded after n expansions. A leaf is expanded only if it has the smallest g-value among all the leaves of the same or lower depths. When a node is expanded, the g-values of its children are evaluated. Typically, in OO methods, the g-value of a node (h,i) is $f(\mathbf{x}_{h,i})$. To cut down the number of function evaluations, BaMSOO eliminates the need for evaluating representative states $\mathbf{x}_{h,i}$ deemed unfit by Gaussian process (GP) posterior bounds. At step t, prior to evaluating f at $\mathbf{x}_{h,i}$, upper and lower confidence bounds ($\mathcal{U}_N$, $\mathcal{L}_N$) on $f(x_{h,i})$ are computed using a GP posterior fitted on $\mathcal{D}_{t-1}$. If a node (h,i)'s $\mathcal{L}_N$ is higher than the best function value found so far f^+, then it is highly likely that this node is sub-optimal. Therefore, its evaluation can be skipped and its g-value is set to be (h,i)'s $\mathcal{U}_N$. Moreover, $x_{h,i}$ is not considered in GP posterior fitting. In summary, BaMSOO makes use of the available information to eliminate the need for sampling from unlikely optimal subspaces.

Upon practical implementation of the algorithm, we faced several key elements (parameters) to decide about, namely: the partition factor K, maximal expandable depth h_{max}, which dimension to split along in a given iteration, the size of the past observation set $|\mathcal{D}_t|$, its update rate (how often it is updated), parameters of the GP bounds, and how many iterations the algorithm can go without sampling the actual function value of a point. These parameters are listed in Table I with the values used when tested with BBOB and CEC2016 functions.

Algorithm 1 BaMSOO pseudocode adapted from [2]

Set $g_{0,0} = f(\mathbf{x}_{0,0})$
Set $f^+ = g_{0,0}$
Initialize the tree $\mathcal{T}_1 = \{0,0\}$
Set $t = 1$, $n = 1, N = 1$, and $\mathcal{D}_t = \{(\mathbf{x}_{0,0}, g(\mathbf{x}_{0,0}))\}$
while true **do**
  Set $\nu_{\min} = \infty$.
  for $h = 0$ to $\min\{\text{depth}(\mathcal{T}_n), h_{\max}(n)\}$ **do**
    Select $(h,j) = \arg\min_{j\in\{j|(h,j)\in L_n\}} g(\mathbf{x}_{h,j})$
    if $g(x_{h,j}) < \nu_{\min}$ **then**
      for $i = 0$ to $k-1$ **do**
        Set $N = N + 1$
        if $\mathcal{L}_N(\mathbf{x}_{h+1,kj+i}|\mathcal{D}_t) \leq f^+$ **then**
          Set $g(\mathbf{x}_{h+1,kj+i}) = f(\mathbf{x}_{h+1,kj+i})$
          Set $t = t + 1$
          $\mathcal{D}_t = \{\mathcal{D}_{t-1}, (\mathbf{x}_{h+1,kj+i}, g(\mathbf{x}_{h+1,kj+i}))\}$
        else
          Set $g(\mathbf{x}_{h+1,kj+i}) = \mathcal{U}_N(\mathbf{x}_{h+1,kj+i}|\mathcal{D}_t)$
        end if
        if $g(\mathbf{x}_{h+1,kj+i}) < f^+$ **then**
          Set $f^+ = g(\mathbf{x}_{h+1,kj+i})$
        end if
      end for
      Add the children of (h,j) to $\mathcal{T}_n$
      Set $\nu_{\min} = g(\mathbf{x}_{h,j})$
      Set $n = n + 1$
    end if
  end for
end while

TABLE I: Parameter settings for BAMSOO on CEC2016 and BBOB testbeds.

Parameter	CEC2016	BBOB
$\lvert\mathcal{D}_t\rvert$	$2 \cdot D$	$2 \cdot D$
$\mathcal{D}_t$'s update rate	$\lvert\mathcal{D}_t\rvert/5$	$\lvert\mathcal{D}_t\rvert/2$
Partition factor (k)	3	3
h_{max}	$\sqrt{\text{FEvals}}$	$\sqrt{\text{FEvals}}$
Length scale (GP kernel)	$\frac{max_{i \in D} u_i - l_i}{6}$	$\frac{max_{i \in D} u_i - l_i}{3}$
Scale factor (GP kernel)	$\frac{max_{i \in D} u_i - l_i}{6}$	$\frac{max_{i \in D} u_i - l_i}{3}$
Number of iterations w.r.t. number of samples	100	10
Selection of splitting coordinate	random	round-robin

III. Numerical Assessment

In this section, we investigate the performance of BAMSOO with respect to SOO using the CEC2016 and BBOB benchmarks.[1] Only algorithms parameters related to the evaluation budget are modified accordingly, SOO's parameters are set to their standard values. With respect to BAMSOO, we have fixed its parameters as in Table I in proportion to the problem dimensionality and the evaluation budget.

A. CEC2016 Testbed

Setup: For 20 runs on CEC2016 functions, we compared BAMSOO's performance with SOO's under expensive-budget settings of $50 \times D$ function evaluations (FEvals) over a search space of $[-100, 100]^D$ for $D \in \{10, 30\}$. The algorithms terminate in one of two settings: when the evaluation budget is exhausted or when the loss r_n (2) is less than 10^{-3}.

Performance Evaluation Procedure: The current best function values are recorded throughout the algorithms runs. Then, the obtained best function values after exhausting the evaluation budget are sorted from the smallest (best) to the largest (worst), and the best, worst, mean, median and standard deviation values for the 20 runs per problem are presented. Error values smaller than 10^{-8} are taken as zero.

Performance Evaluation Discussion: The performance results of SOO and BAMSOO for 10-D search space are reported in Tables II and III, respectively. Whereas 30-D results are reported in Tables IV and V, respectively. From the results, one can note that the algorithms performance is comparable suggesting less advantage of BaMSOO over SOO in expensive-budget settings (a conclusion that is supported by the BBOB testbed as well). This can be attributed to the poor accuracy of the GP posterior bounds in the early phase of search, which gets better towards finer partitions of the decision space after with more allocated function evaluations. Moreover, ill-condition functions (e.g., the bent cigar function f_1) appear to be the most challenging to solve, which can be due to the algorithms coordinate-wise partitioning scheme. Although incorporating randomness in the way BaMSOO selects the next dimension along which partitioning takes place (rather than SOO's round-robin) may improve its performance as suggested by the **Best** column of the tables, it introduces inconsistencies in its performance as shown by the **Std Deviation** column. Besides solution quality evaluation, we also report the algorithms computational complexity in line with guidelines in [6]. While timing results show comparable computational complexity of BAMSOO, one must remember that this is considerably dependent on the sampling decision based on the GP posterior bounds. This factor grows bigger with more function evaluations and may make BAMSOO extremely slow. We experienced this effect on BBOB testbed with cheap-budget settings.

[1]Our MATLAB implementation of BAMSOO is provided at http://ash-aldujaili.github.io/BaMSOO/, whereas SOO's MATLAB source code is available at sequel.lille.inria.fr/Software/StoSOO.

Function	Best	Worst	Median	Mean	Std Deviation
1	$2.809 \cdot 10^8$	$2.809 \cdot 10^8$	$2.809 \cdot 10^8$	$2.809 \cdot 10^8$	$6.115 \cdot 10^{-8}$
2	$1.076 \cdot 10^4$	$1.076 \cdot 10^4$	$1.076 \cdot 10^4$	$1.076 \cdot 10^4$	$1.866 \cdot 10^{-12}$
3	$7.322 \cdot 10^0$	$7.322 \cdot 10^0$	$7.322 \cdot 10^0$	$7.322 \cdot 10^0$	$1.166 \cdot 10^{-13}$
4	$1.346 \cdot 10^3$	$1.346 \cdot 10^3$	$1.346 \cdot 10^3$	$1.346 \cdot 10^3$	$4.666 \cdot 10^{-13}$
5	$2.754 \cdot 10^0$	$2.754 \cdot 10^0$	$2.754 \cdot 10^0$	$2.754 \cdot 10^0$	$1.166 \cdot 10^{-13}$
6	$1.218 \cdot 10^0$	$1.218 \cdot 10^0$	$1.218 \cdot 10^0$	$1.218 \cdot 10^0$	$1.166 \cdot 10^{-13}$
7	$1.793 \cdot 10^0$	$1.793 \cdot 10^0$	$1.793 \cdot 10^0$	$1.793 \cdot 10^0$	$0.000 \cdot 10^0$
8	$2.315 \cdot 10^1$	$2.315 \cdot 10^1$	$2.315 \cdot 10^1$	$2.315 \cdot 10^1$	$0.000 \cdot 10^0$
9	$3.520 \cdot 10^0$	$3.520 \cdot 10^0$	$3.520 \cdot 10^0$	$3.520 \cdot 10^0$	$0.000 \cdot 10^0$
10	$6.364 \cdot 10^5$	$6.364 \cdot 10^5$	$6.364 \cdot 10^5$	$6.364 \cdot 10^5$	$0.000 \cdot 10^0$
11	$6.979 \cdot 10^0$	$6.979 \cdot 10^0$	$6.979 \cdot 10^0$	$6.979 \cdot 10^0$	$0.000 \cdot 10^0$
12	$1.797 \cdot 10^2$	$1.797 \cdot 10^2$	$1.797 \cdot 10^2$	$1.797 \cdot 10^2$	$4.666 \cdot 10^{-13}$
13	$3.792 \cdot 10^2$	$3.792 \cdot 10^2$	$3.792 \cdot 10^2$	$3.792 \cdot 10^2$	$2.333 \cdot 10^{-13}$
14	$2.127 \cdot 10^2$	$2.127 \cdot 10^2$	$2.127 \cdot 10^2$	$2.127 \cdot 10^2$	$4.666 \cdot 10^{-13}$
15	$4.128 \cdot 10^2$	$4.128 \cdot 10^2$	$4.128 \cdot 10^2$	$4.128 \cdot 10^2$	$0.000 \cdot 10^0$

TABLE II: Statistic Performance Results of **SOO** for **CEC16**'s 10-D functions.

Function	Best	Worst	Median	Mean	Std Deviation
1	$9.712 \cdot 10^7$	$9.124 \cdot 10^8$	$4.400 \cdot 10^8$	$4.773 \cdot 10^8$	$2.414 \cdot 10^8$
2	$8.596 \cdot 10^3$	$4.500 \cdot 10^4$	$3.060 \cdot 10^4$	$2.600 \cdot 10^4$	$1.032 \cdot 10^4$
3	$6.391 \cdot 10^0$	$1.046 \cdot 10^1$	$8.148 \cdot 10^0$	$8.115 \cdot 10^0$	$1.191 \cdot 10^0$
4	$1.406 \cdot 10^3$	$1.766 \cdot 10^3$	$1.552 \cdot 10^3$	$1.578 \cdot 10^3$	$1.083 \cdot 10^2$
5	$1.023 \cdot 10^0$	$4.013 \cdot 10^0$	$2.576 \cdot 10^0$	$2.543 \cdot 10^0$	$7.869 \cdot 10^{-1}$
6	$4.146 \cdot 10^{-1}$	$2.786 \cdot 10^0$	$7.787 \cdot 10^{-1}$	$9.221 \cdot 10^{-1}$	$5.292 \cdot 10^{-1}$
7	$4.198 \cdot 10^{-1}$	$1.321 \cdot 10^1$	$4.467 \cdot 10^0$	$5.155 \cdot 10^0$	$3.341 \cdot 10^0$
8	$8.419 \cdot 10^0$	$3.135 \cdot 10^2$	$2.955 \cdot 10^1$	$4.614 \cdot 10^1$	$6.633 \cdot 10^1$
9	$3.381 \cdot 10^0$	$4.320 \cdot 10^0$	$3.812 \cdot 10^0$	$3.824 \cdot 10^0$	$2.789 \cdot 10^{-1}$
10	$7.291 \cdot 10^3$	$5.096 \cdot 10^5$	$1.992 \cdot 10^5$	$2.283 \cdot 10^5$	$1.554 \cdot 10^5$
11	$4.856 \cdot 10^0$	$1.660 \cdot 10^1$	$8.474 \cdot 10^0$	$9.084 \cdot 10^0$	$3.013 \cdot 10^0$
12	$4.291 \cdot 10^1$	$2.685 \cdot 10^2$	$1.714 \cdot 10^2$	$1.790 \cdot 10^2$	$6.176 \cdot 10^1$
13	$3.344 \cdot 10^2$	$4.112 \cdot 10^2$	$3.541 \cdot 10^2$	$3.641 \cdot 10^2$	$2.474 \cdot 10^1$
14	$1.971 \cdot 10^2$	$2.156 \cdot 10^2$	$2.084 \cdot 10^2$	$2.077 \cdot 10^2$	$5.231 \cdot 10^0$
15	$3.431 \cdot 10^1$	$4.665 \cdot 10^2$	$3.881 \cdot 10^2$	$3.050 \cdot 10^2$	$1.520 \cdot 10^2$

TABLE III: Statistic Performance Results of **BaMSOO** for **CEC16**'s 10-D functions.

B. BBOB Testbed

Setup: BAMSOO and SOO are run on the 24 functions (15 runs/instances per function) of the BBOB testbed under cheap-budget settings of $10^4 \times D$ function evaluations (FEvals) to reach a target function value f_t over a search space of $[-5, 5]^D$ for $D \in \{2, 3, 5, 10, 20\}$. The algorithms terminate in one of two cases: exhausting the evaluation budget; or hitting f_t.

Performance Evaluation Procedure: The performance experiments are set up according to [12], where each algorithm is run on the functions given in [7] with multiple trials per function. A set of target function values is specified per function. The algorithms are evaluated based on the number of function evalutions required to reach a target. The Expected

Function	Best	Worst	Median	Mean	Std Deviation
1	$5.543 \cdot 10^{9}$	$5.543 \cdot 10^{9}$	$5.543 \cdot 10^{9}$	$5.543 \cdot 10^{9}$	$9.784 \cdot 10^{-7}$
2	$4.997 \cdot 10^{4}$	$4.997 \cdot 10^{4}$	$4.997 \cdot 10^{4}$	$4.997 \cdot 10^{4}$	$7.465 \cdot 10^{-12}$
3	$3.194 \cdot 10^{1}$	$3.194 \cdot 10^{1}$	$3.194 \cdot 10^{1}$	$3.194 \cdot 10^{1}$	$5.832 \cdot 10^{-14}$
4	$6.641 \cdot 10^{3}$	$6.641 \cdot 10^{3}$	$6.641 \cdot 10^{3}$	$6.641 \cdot 10^{3}$	$0.000 \cdot 10^{0}$
5	$4.348 \cdot 10^{0}$	$4.348 \cdot 10^{0}$	$4.348 \cdot 10^{0}$	$4.348 \cdot 10^{0}$	$5.832 \cdot 10^{-14}$
6	$9.356 \cdot 10^{-1}$	$9.356 \cdot 10^{-1}$	$9.356 \cdot 10^{-1}$	$9.356 \cdot 10^{-1}$	$2.333 \cdot 10^{-13}$
7	$1.935 \cdot 10^{1}$	$1.935 \cdot 10^{1}$	$1.935 \cdot 10^{1}$	$1.935 \cdot 10^{1}$	$3.499 \cdot 10^{-13}$
8	$2.487 \cdot 10^{5}$	$2.487 \cdot 10^{5}$	$2.487 \cdot 10^{5}$	$2.487 \cdot 10^{5}$	$1.493 \cdot 10^{-10}$
9	$1.342 \cdot 10^{1}$	$1.342 \cdot 10^{1}$	$1.342 \cdot 10^{1}$	$1.342 \cdot 10^{1}$	$3.499 \cdot 10^{-13}$
10	$2.948 \cdot 10^{7}$	$2.948 \cdot 10^{7}$	$2.948 \cdot 10^{7}$	$2.948 \cdot 10^{7}$	$7.644 \cdot 10^{-9}$
11	$7.589 \cdot 10^{1}$	$7.589 \cdot 10^{1}$	$7.589 \cdot 10^{1}$	$7.589 \cdot 10^{1}$	$0.000 \cdot 10^{0}$
12	$9.878 \cdot 10^{2}$	$9.878 \cdot 10^{2}$	$9.878 \cdot 10^{2}$	$9.878 \cdot 10^{2}$	$4.666 \cdot 10^{-13}$
13	$8.230 \cdot 10^{2}$	$8.230 \cdot 10^{2}$	$8.230 \cdot 10^{2}$	$8.230 \cdot 10^{2}$	$4.666 \cdot 10^{-13}$
14	$3.166 \cdot 10^{2}$	$3.166 \cdot 10^{2}$	$3.166 \cdot 10^{2}$	$3.166 \cdot 10^{2}$	$2.333 \cdot 10^{-13}$
15	$1.142 \cdot 10^{3}$	$1.142 \cdot 10^{3}$	$1.142 \cdot 10^{3}$	$1.142 \cdot 10^{3}$	$9.331 \cdot 10^{-13}$

TABLE IV: Statistic Performance Results of **SOO** for **CEC16**'s 30-D functions.

Function	Best	Worst	Median	Mean	Std Deviation
1	$5.884 \cdot 10^{9}$	$1.710 \cdot 10^{10}$	$8.717 \cdot 10^{9}$	$9.322 \cdot 10^{9}$	$2.717 \cdot 10^{9}$
2	$4.835 \cdot 10^{4}$	$7.567 \cdot 10^{4}$	$5.945 \cdot 10^{4}$	$5.979 \cdot 10^{4}$	$6.124 \cdot 10^{3}$
3	$2.944 \cdot 10^{1}$	$3.951 \cdot 10^{1}$	$3.329 \cdot 10^{1}$	$3.335 \cdot 10^{1}$	$2.243 \cdot 10^{0}$
4	$4.804 \cdot 10^{3}$	$6.395 \cdot 10^{3}$	$5.470 \cdot 10^{3}$	$5.543 \cdot 10^{3}$	$4.250 \cdot 10^{2}$
5	$2.615 \cdot 10^{0}$	$4.280 \cdot 10^{0}$	$3.718 \cdot 10^{0}$	$3.666 \cdot 10^{0}$	$4.635 \cdot 10^{-1}$
6	$1.017 \cdot 10^{0}$	$2.702 \cdot 10^{0}$	$2.160 \cdot 10^{0}$	$2.117 \cdot 10^{0}$	$4.399 \cdot 10^{-1}$
7	$1.588 \cdot 10^{1}$	$2.870 \cdot 10^{1}$	$2.070 \cdot 10^{1}$	$2.180 \cdot 10^{1}$	$3.830 \cdot 10^{0}$
8	$4.899 \cdot 10^{4}$	$5.907 \cdot 10^{5}$	$1.386 \cdot 10^{5}$	$1.658 \cdot 10^{5}$	$1.173 \cdot 10^{5}$
9	$1.263 \cdot 10^{1}$	$1.379 \cdot 10^{1}$	$1.357 \cdot 10^{1}$	$1.346 \cdot 10^{1}$	$3.061 \cdot 10^{-1}$
10	$2.536 \cdot 10^{6}$	$4.199 \cdot 10^{7}$	$1.707 \cdot 10^{7}$	$1.820 \cdot 10^{7}$	$9.110 \cdot 10^{6}$
11	$4.171 \cdot 10^{1}$	$1.567 \cdot 10^{2}$	$1.229 \cdot 10^{2}$	$1.150 \cdot 10^{2}$	$3.195 \cdot 10^{1}$
12	$4.425 \cdot 10^{2}$	$1.812 \cdot 10^{3}$	$1.041 \cdot 10^{3}$	$1.036 \cdot 10^{3}$	$2.912 \cdot 10^{2}$
13	$4.451 \cdot 10^{2}$	$8.195 \cdot 10^{2}$	$6.994 \cdot 10^{2}$	$6.777 \cdot 10^{2}$	$1.167 \cdot 10^{2}$
14	$2.666 \cdot 10^{2}$	$3.157 \cdot 10^{2}$	$2.902 \cdot 10^{2}$	$2.919 \cdot 10^{2}$	$1.543 \cdot 10^{1}$
15	$6.878 \cdot 10^{2}$	$1.247 \cdot 10^{3}$	$1.143 \cdot 10^{3}$	$1.061 \cdot 10^{3}$	$1.743 \cdot 10^{2}$

TABLE V: Statistic Performance Results of **BaMSOO** for **CEC16**'s 30-D functions.

Function	10-Dimension	30-Dimension
1	2.005	7.832
2	2.321	12.246
3	3.281	17.596
4	1.976	7.566
5	3.144	17.905
6	2.068	7.804
7	2.042	8.054
8	2.130	9.419
9	2.526	15.490
10	2.921	11.987
11	2.480	10.724
12	2.561	11.552
13	2.173	8.863
14	2.571	11.929
15	3.024	17.020

TABLE VI: Normalized Average Time Complexity $T1/T0$ of **SOO** for **CEC16**'s functions.

Running Time (ERT) used in the coming figures and tables, depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [12]. Statistical significance is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

Function	10-Dimension	30-Dimension
1	2.733	8.517
2	2.807	8.884
3	3.159	12.459
4	2.733	8.799
5	2.880	9.857
6	2.715	8.668
7	2.697	8.592
8	2.767	8.608
9	2.738	8.944
10	2.760	8.713
11	2.852	9.517
12	2.789	8.829
13	2.788	9.052
14	2.781	8.987
15	3.208	12.820

TABLE VII: Normalized Average Time Complexity $T1/T0$ of **BaMSOO** for **CEC16**'s functions.

Performance Evaluation Discussion: The algorithms' ERT can be examined in Table VIII for 5-D and 20-D as a representative of low- and high-dimension problems, respectively. Overall, BaMSOO outperforms SOO with a shorter ERT and a statistically significant evidence. Nevertheless, the linear slope function f_5 in low dimensions imposes a challenge on BAMSOO as the prior makes no use of the samples $\mathcal{D}_t$ with similar rewards. The attractive section function f_6 is the most difficult function to optimize.

The empirical cumulative distributions of the ERT (FEvals) divided by dimension in 2-, 5-, 10-, and 20-D over different function categories are shown in the Figures 1,2,3, and 4, respectively. These categories capture typical difficulties in continuous domain search. Across all dimensions and categories, BaMSOO is outperforming SOO with a pronounced performance gap in lower dimensions.

IV. CONCLUSION

This paper complements the theoretical analysis of the recently proposed BaMSOO algorithm for solving bound-constrained black-box problems. It provides an extensive evaluation of the algorithm on the CEC2016 and BBOB testbeds under epxpensive-budget and cheap-budget settings, respectively. Furthermore, we compare BaMSOO with SOO from which BaMSOO was derived. Under a very weak assumption on the objective function smoothness, both of the algorithms enjoy a finite-time theoretical convergence. Nevertheless, empirically[2], BaMSOO outperforms SOO over different typical real-world scenarios with more function evaluations. Dimensionality and ill-conditioning appear to be the most challenging factor for both the algorithms.

Practical Issues & Future Directions: BaMSOO comes with several practical issues compared to SOO:

Parameter Tuning for the Prior: BaMSOO comes with several parameters to be tuned for which overfitting is not uncommon. We followed the guidelines given in [13] for setting some of these parameters values.

[2]The data of these experiments will be made available on the CEC2016 and BBOB webpages.

5-D

Δf_{opt}	10	0.1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	11	12	12	12	12	15/15
1: SOO	1.6(0.8)	11288(13320)	∞	∞	∞*5.0e4*	0/15
2: BaM	1.2(0.7)	**31**(143)$^{\star 2}$	**56**(20)$^{\star 3}$	**72**(10)$^{\star 3}$	**98**(19)$^{\star 3}$	15/15
$\mathbf{f_2}$	83	88	90	92	94	15/15
1: SOO	∞	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	**52**(33)$^{\star 3}$	**120**(144)$^{\star 3}$	**177**(2)$^{\star 3}$	**241**(473)$^{\star 3}$	**277**(734)$^{\star 3}$	11/15
$\mathbf{f_3}$	716	1637	1646	1650	1654	15/15
1: SOO	12(20)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	1.6(4)	130(208)	129(163)	129(111)	129(175)	3/15
$\mathbf{f_4}$	809	1688	1817	1886	1903	15/15
1: SOO	87(135)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	12(64)	426(422)	396(406)	∞	∞*5.0e4*	0/15
$\mathbf{f_5}$	10	10	10	10	10	15/15
1: SOO	13(1)	**97**(6)$^{\star 2}$	**264**(13)*	514(8)	840(4)	15/15
2: BaM	**7.6**(0.7)*	3765(3747)	3769(5921)	3773(3703)	3777(10826)	10/15
$\mathbf{f_6}$	114	281	580	1038	1332	15/15
1: SOO	152(111)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	65(85)	∞	∞	∞	∞*5.0e4*	0/15
$\mathbf{f_7}$	24	1171	1572	1572	1597	15/15
1: SOO	4.1(4)	132(107)	∞	∞	∞*5.0e4*	0/15
2: BaM	3.4(3)	**5.0**(3)$^{\star 3}$	**18**(27)$^{\star 3}$	**18**(15)$^{\star 3}$	**25**(43)$^{\star 3}$	10/15
$\mathbf{f_8}$	73	336	391	410	422	15/15
1: SOO	329(513)	971(409)	∞	∞	∞*5.0e4*	0/15
2: BaM	**3.5**(5)*	**34**(33)*	**120**(178)$^{\star 3}$	**245**(430)$^{\star 3}$	**407**(184)$^{\star 3}$	4/15
$\mathbf{f_9}$	35	214	300	335	369	15/15
1: SOO	3.5(1)	1536(1574)	∞	∞	∞*5.0e4*	0/15
2: BaM	5.0(2)	140(140)	**452**(486)$^{\star 2}$	**1068**(701)$^{\star 2}$	∞***5.0e4***$^{\star 2}$	0/15
$\mathbf{f_{10}}$	349	574	626	829	880	15/15
1: SOO	2019(2003)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	165(255)	∞	∞	∞	∞*5.0e4*	0/15
$\mathbf{f_{11}}$	143	763	1177	1467	1673	15/15
1: SOO	25(34)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	22(6)	∞	∞	∞	∞*5.0e4*	0/15
$\mathbf{f_{12}}$	108	371	461	1303	1494	15/15
1: SOO	3022(1386)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	**11**(13)$^{\star 3}$	**20**(20)$^{\star 3}$	**181**(102)$^{\star 3}$	**552**(528)$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15
$\mathbf{f_{13}}$	132	250	1310	1752	2255	15/15
1: SOO	927(743)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	**4.3**(4)$^{\star 3}$	**21**(16)$^{\star 3}$	**28**(39)$^{\star 3}$	**56**(39)$^{\star 3}$	**108**(77)$^{\star 3}$	3/15
$\mathbf{f_{14}}$	10	58	139	251	476	15/15
1: SOO	0.63(1.0)	583(658)	∞	∞	∞*5.0e4*	0/15
2: BaM	0.52(0.4)	5.9(6)	**74**(101)$^{\star 3}$	**1442**(1342)$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15
$\mathbf{f_{15}}$	511	19369	20073	20769	21359	14/15
1: SOO	13(36)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	**1.9**(3)*	**39**(52)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15
$\mathbf{f_{16}}$	120	2662	10449	11644	12095	15/15
1: SOO	1.1(0.6)	18(25)	∞	∞	∞*5.0e4*	0/15
2: BaM	1.1(0.6)	**0.69**(1.0)$^{\star 3}$	**2.6**(3)$^{\star 3}$	**5.5**(8)$^{\star 3}$	**11**(20)$^{\star 3}$	5/15
$\mathbf{f_{17}}$	5.2	899	3669	6351	7934	15/15
1: SOO	1.8(2)	779(1474)	∞	∞	∞*5.0e4*	0/15
2: BaM	1.5(1)	**1.3**(0.9)$^{\star 3}$	**4.9**(4)$^{\star 3}$	**7.3**(10)$^{\star 3}$	**21**(13)$^{\star 3}$	4/15
$\mathbf{f_{18}}$	103	3968	9280	10905	12469	15/15
1: SOO	0.98(0.8)	178(271)	∞	∞	∞*5.0e4*	0/15
2: BaM	0.75(0.6)	**1.1**(2)$^{\star 3}$	**6.7**(9)$^{\star 3}$	**67**(60)$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15
$\mathbf{f_{19}}$	1	242	1.2e5	1.2e5	1.2e5	15/15
1: SOO	1	**1.5**(0.4)$^{\star 3}$	∞	∞	∞*5.0e4*	0/15
2: BaM	1	2.6(0.5)	∞$^{\star 2}$	∞$^{\star 2}$	∞***5.0e4***$^{\star 2}$	0/15
$\mathbf{f_{20}}$	16	38111	54470	54861	55313	14/15
1: SOO	17(4)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	**9.3**(2)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15
$\mathbf{f_{21}}$	41	1674	1705	1729	1757	14/15
1: SOO	1.1(0.9)	19(38)	204(273)	∞	∞*5.0e4*	0/15
2: BaM	0.87(0.5)	1.3(2)	**2.5**(2)$^{\star 2}$	**8.3**(12)$^{\star 3}$	**11**(7)$^{\star 3}$	13/15
$\mathbf{f_{22}}$	71	938	1008	1040	1068	14/15
1: SOO	1.3(0.7)	64(50)	329(620)	∞	∞*5.0e4*	0/15
2: BaM	0.76(0.4)	21(29)	91(74)	155(290)	325(217)	2/15
$\mathbf{f_{23}}$	3.0	14249	31654	33030	34256	15/15
1: SOO	3.4(3)	24(19)	∞	∞	∞*5.0e4*	0/15
2: BaM	3.5(3)	**1.1**(0.8)$^{\star 2}$	**5.2**(4)$^{\star 3}$	∞$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15
$\mathbf{f_{24}}$	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: SOO	207(324)	∞	∞	∞	∞*5.0e4*	0/15
2: BaM	**4.9**(4)$^{\star 2}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***5.0e4***$^{\star 3}$	0/15

20-D

Δf_{opt}	10	0.1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	43	43	43	43	43	15/15
1: SOO	17(3)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	**9.5**(4)$^{\star 2}$	**44**(6)$^{\star 3}$	**102**(18)$^{\star 3}$	**183**(23)$^{\star 3}$	**292**(56)$^{\star 3}$	15/15
$\mathbf{f_2}$	385	387	390	391	393	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	**207**(391)$^{\star 3}$	**483**(649)$^{\star 3}$	**494**(391)$^{\star 3}$	**830**(1154)$^{\star 3}$	**1478**(2038)$^{\star 3}$	4/15
$\mathbf{f_3}$	5066	7635	7643	7646	7651	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_4}$	4722	7666	7700	7758	1.4e5	9/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_5}$	41	41	41	41	41	15/15
1: SOO	115(5)	571(13)	1353(6)	2442(4)	3967(23)	15/15
2: BaM	**3.7**(0.0)$^{\star 3}$	**7.8**(0.0)$^{\star 3}$	**12**(0.0)$^{\star 3}$	**16**(0.0)$^{\star 3}$	**20**(0.0)$^{\star 3}$	15/15
$\mathbf{f_6}$	1296	3413	5220	6728	8409	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞*	∞*	∞*	∞*	∞***2.0e5****	0/15
$\mathbf{f_7}$	1351	9503	16524	16524	16969	15/15
1: SOO	453(370)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	**27**(47)$^{\star 2}$	∞$^{\star 2}$	∞$^{\star 2}$	∞$^{\star 2}$	∞***2.0e5***$^{\star 2}$	0/15
$\mathbf{f_8}$	2039	4040	4219	4371	4484	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_9}$	1716	3277	3455	3594	3727	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{10}}$	7413	10735	14920	17073	17476	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞	∞	∞	∞	∞*2.0e5*	0/15
$\mathbf{f_{11}}$	1002	6278	9762	12285	14831	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞	∞	∞	∞	∞*2.0e5*	0/15
$\mathbf{f_{12}}$	1042	2740	4140	12407	13827	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	**221**(60)$^{\star 3}$	**488**(730)$^{\star 3}$	**691**(749)$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{13}}$	652	2751	18749	24455	30201	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	**4412**(3913)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{14}}$	75	304	932	1648	15661	15/15
1: SOO	4.6(2)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	3.6(1)	**326**(509)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{15}}$	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{16}}$	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: SOO	2.3(2)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	1.4(0.3)	**19**(16)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{17}}$	63	4005	30677	56288	80472	15/15
1: SOO	1.7(1)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	1.2(0.7)	**154**(87)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{18}}$	621	19561	67569	1.3e5	1.5e5	15/15
1: SOO	7.9(7)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	5.2(3)	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{19}}$	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: SOO	1	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	1	**1.9**(2)$^{\star 2}$	∞$^{\star 2}$	∞$^{\star 2}$	∞***2.0e5***$^{\star 2}$	0/15
$\mathbf{f_{20}}$	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: SOO	51(5)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	**28**(0.0)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{21}}$	561	14103	14643	15567	17589	15/15
1: SOO	15(52)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	3.4(2)	11(14)	192(99)	∞	∞*2.0e5*	0/15
$\mathbf{f_{22}}$	467	23491	24948	26847	1.3e5	12/15
1: SOO	100(113)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	13(25)	∞	∞	∞	∞*2.0e5*	0/15
$\mathbf{f_{23}}$	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: SOO	1.3(2)	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	1.4(2)	**1.2**(0.5)$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15
$\mathbf{f_{24}}$	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: SOO	∞	∞	∞	∞	∞*2.0e5*	0/15
2: BaM	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞$^{\star 3}$	∞***2.0e5***$^{\star 3}$	0/15

TABLE VIII: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimensions 5 (left) and 20 (right). The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\mathrm{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. 1:SOO is SOO and 2:BaM is BaMSOO. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \dots\}$ is the number following the $\star$ symbol, with Bonferroni correction of 48. A $\downarrow$ indicates the same tested against the best algorithm of BBOB-2009.

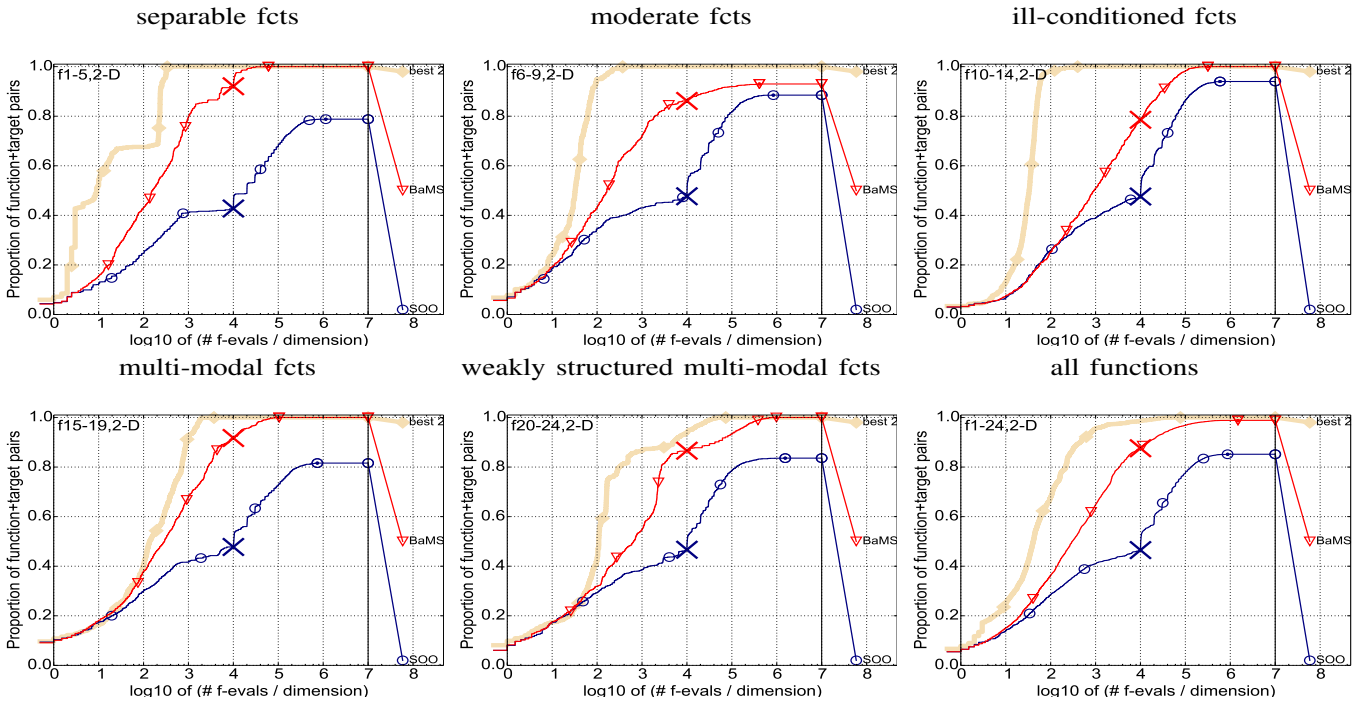

Fig. 1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in 10 [8..2] for all functions and subgroups in 2-D. The best 2009 line corresponds to the best ERT observed during BBOB 2009 for each single target.

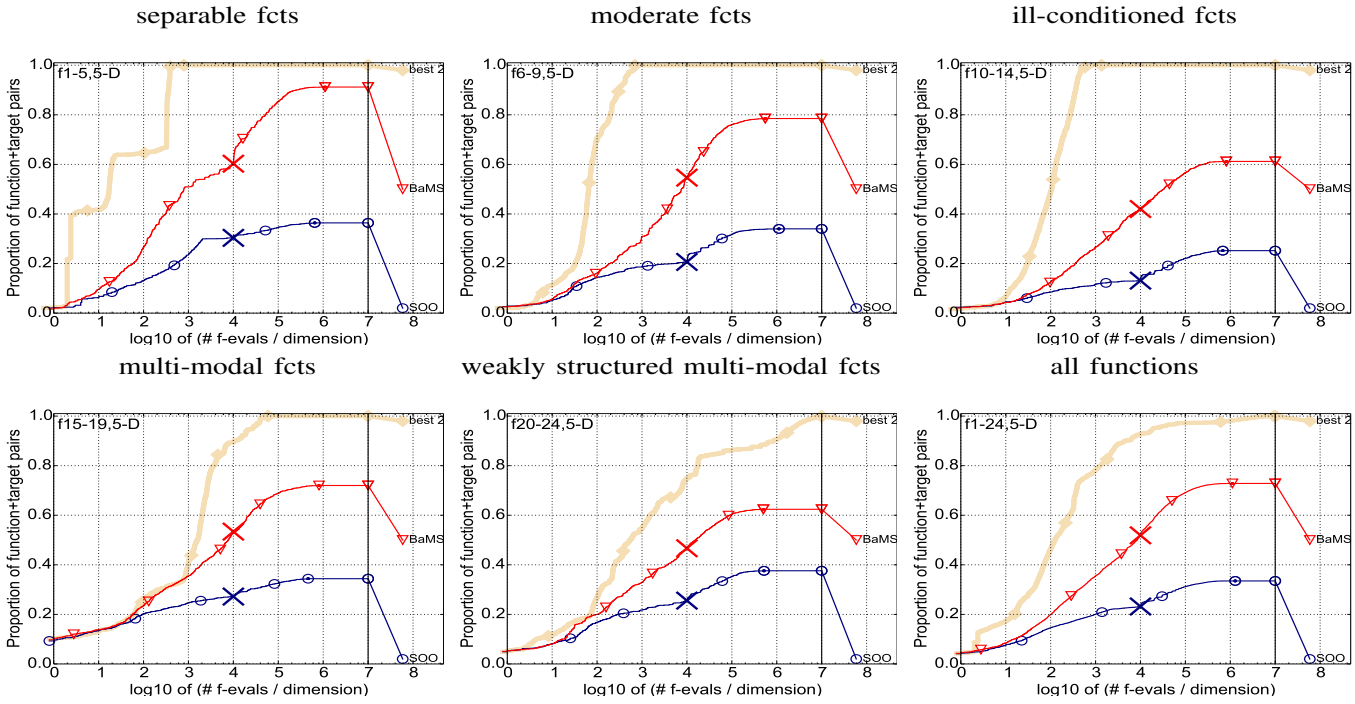

Fig. 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in 10 [8..2] for all functions and subgroups in 5-D. The best 2009 line corresponds to the best ERT observed during BBOB 2009 for each single target.

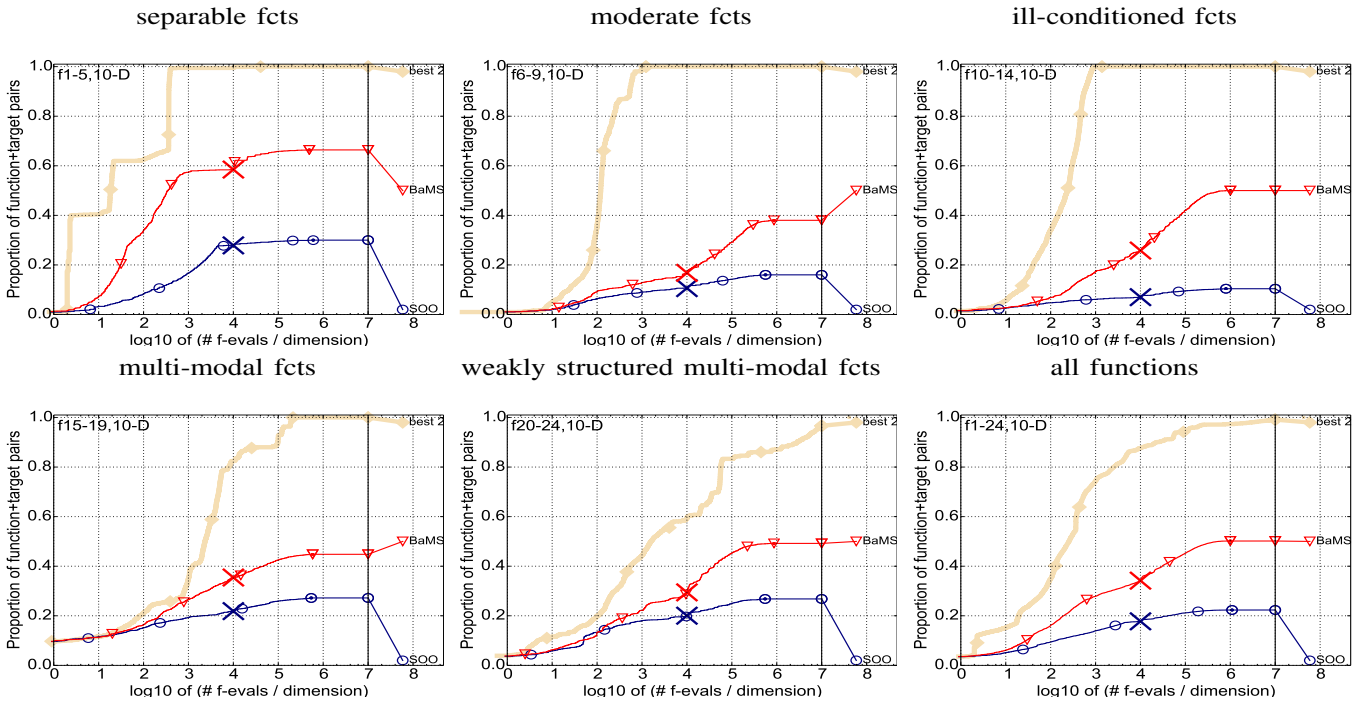

Fig. 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in 10 [8..2] for all functions and subgroups in 10-D. The best 2009 line corresponds to the best ERT observed during BBOB 2009 for each single target.

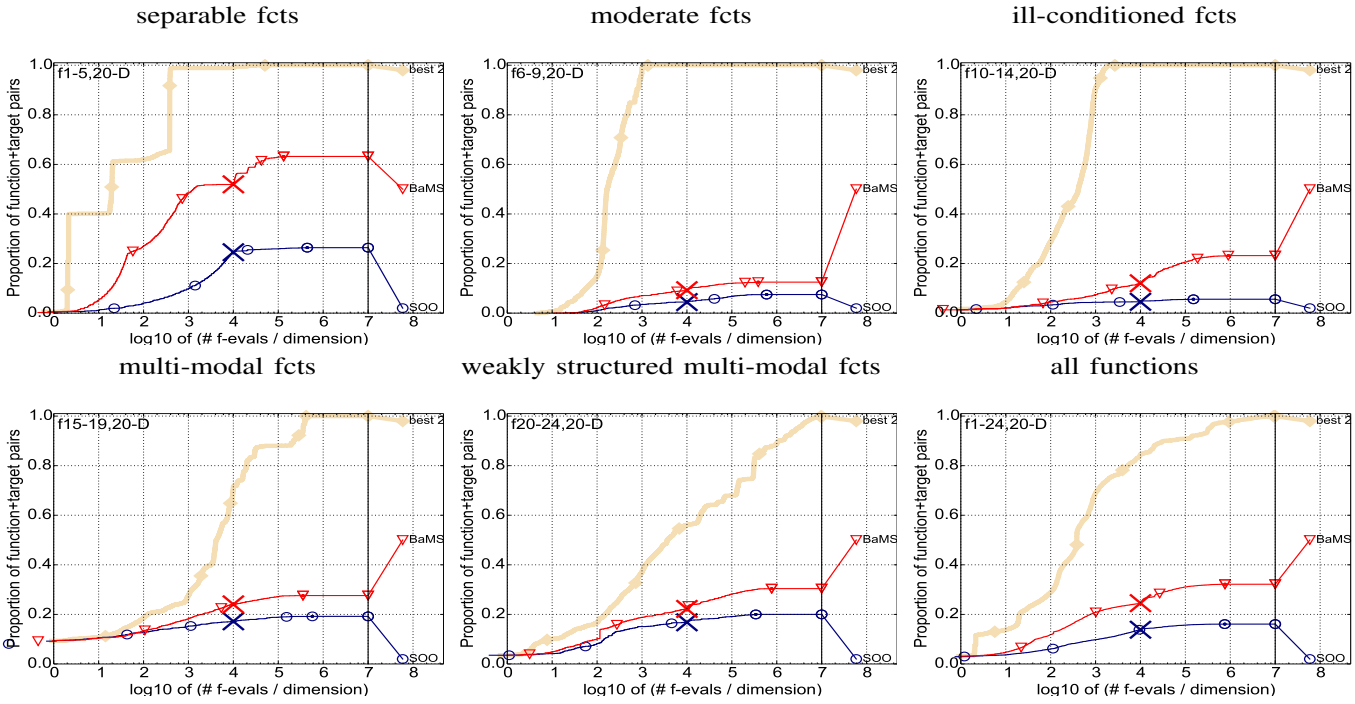

Fig. 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in 10 [8..2] for all functions and subgroups in 20-D. The best 2009 line corresponds to the best ERT observed during BBOB 2009 for each single target.

Prior Update: Computational complexity of updating the prior imposes constrains on how often the computation is performed, and the number of observations $|\mathcal{D}_t|$ to be used. Moreover, for big search spaces, local priors are more promising than a single global prior. We have used a single prior over $\mathcal{X}$ with an update rate and size of $\mathcal{D}_t$ linear in dimension.

Termination Criterion: #FEvals can not be used as a terminating criterion for time-constrained problems as the number of function evaluations per time depends on the prior. If SOO evaluates and expands n, m nodes, respectively. BaMSOO can possibly expand m nodes with far less than n evaluations.

ACKNOWLEDGMENT

The authors wish to extend their thanks to the Ministry of Education (MoE), Singapore, for providing financial support through tier I (No. M4011269) funding to conduct this study.

REFERENCES

[1] J. Hu, Y. Wang, E. Zhou, M. C. Fu, and S. I. Marcus, "A survey of some model-based methods for global optimization," in *Optimization, Control, and Applications of Stochastic Systems*. Springer, 2012, pp. 157–179.

[2] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas, "Bayesian multi-scale optimistic optimization," *AI and Statistics*, pp. 1005–1014, 2014.

[3] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[4] R. Munos, "Optimistic optimization of deterministic functions without the knowledge of its smoothness," in *Advances in neural information processing systems*, 2011.

[5] H. Robbins *et al.*, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[6] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, and B. Qu, "Problem definition and evaluation criteria for cec 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization," *Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore, Tech. Rep*, 2014.

[7] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2009, updated February 2010. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads

[8] C. E. Rasmussen, "Gaussian processes for machine learning." MIT Press, 2006.

[9] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of global optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[10] S. Bubeck, G. Stoltz, C. Szepesvári, and R. Munos, "Online optimization in x-armed bandits," in *Advances in Neural Information Processing Systems*, 2009, pp. 201–208.

[11] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[12] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2012: Experimental setup," INRIA, Tech. Rep., 2012. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads

[13] N. Hansen, "The CMA evolution strategy: A tutorial," 2005. [Online]. Available: http://www.lri.fr/~hansen/cmatutorial.pdf