Adaptive Space Partitioning for Parallel Bayesian Optimization

Maxime Gobert, Jan Gmys, Nouredine Melab, Daniel Tuyttens

▶ **To cite this version:**

Maxime Gobert, Jan Gmys, Nouredine Melab, Daniel Tuyttens. Adaptive Space Partitioning for Parallel Bayesian Optimization. HPCS 2020 - The 18th International Conference on High Performance Computing & Simulation, Mar 2021, Barcelona / Virtual, Spain. hal-03121209

HAL Id: hal-03121209
https://inria.hal.science/hal-03121209

Submitted on 26 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Adaptive Space Partitioning for Parallel Bayesian Optimization

Maxime Gobert
University of Mons
Mathematics and Operational Research
Mons, Belgium
maxime.gobert@umons.ac.be

Jan Gmys
Inria Lille Nord-Europe
Lille, France
jan.gmys@inria.fr

Nouredine Melab
Université de Lille, CNRS/CRIStAL
Inria Lille - Nord Europe
Lille, France
nouredine.melab@univ-lille.fr

Daniel Tuyttens
University of Mons
Mathematics and Operational Research
Mons, Belgium
daniel.tuyttens@umons.ac.be

***Abstract*—This paper presents a new approach to provide large and well-chosen batches of points to evaluate in parallel in the context of batch-parallel Global Optimization. The method combines Bayesian Optimization and Design Space Partitioning in order to independently select different candidates according to the expected information they could provide if included in the design of experiment. The information in question is given by a figure of merit computed thanks to the chosen bayesian model. The presented algorithm is designed to be flexible and adaptable to various objective functions since it automatically adapts the space partition during the optimization process. Three benchmark functions are investigated in this study and reveal mixed results, but also interesting features for future works.**

***Index Terms*—Parallel Bayesian Optimization, Design Space partitioning, Acquisition Process, Black-Box Optimization**

INTRODUCTION

This paper deals with Global Optimization and more precisely Black-Box Optimization, $\min_{\mathcal{D}} f(\boldsymbol{x})$, where $\mathcal{D}$ is a box constrained domain. The term *Black-Box* refers to the fact that the objective function is not analytically known and therefore neither its gradient, no information is known *a priori*. Furthermore, it is assumed that the evaluation of the objective function comes at a high simulation cost, as it is typically the case when $f(\boldsymbol{x})$ is the outcome of a numerical simulation for an input vector $\boldsymbol{x} \in \mathcal{D}$. A classical approach for dealing with computationally expensive objective functions is to build a cheaper-to-evaluate approximation of the objective function called surrogate (model) or metamodel. The basic principle of surrogate-assisted optimization consists in first building an initial surrogate model using few points obtained with some sampling technique (e.g. Latin Hypercube Sampling). The surrogate is used to determine promising candidate solutions to evaluate with the (real) objective function which, in turn, are used to iteratively refine the surrogate.

Our focus is set on Bayesian Optimization (BO) which has proved to be very efficient in this context. The original concept is introduced by Kushner *et al.* [1] and generalized by Mockus *et al.* [2]. In BO, a prior distribution is set over the sample points, which is commonly a gaussian prior. After the evaluation of the sample, the prior is updated as a posterior distribution used to compute a figure of merit that is designed to provide a valuable candidate through its optimization. This figure of merit is computed thanks to the *Acquisition Function* (AF), also named Infill Criterion (IC) and helps finding the areas to sample in the landscape to be optimized.

Facing time-consuming objective functions, parallel computing can be used as a complementary way, in addition to surrogate models, to further speed up the optimization process. However, the joint usage of parallel processing and surrogate-based optimization is challenging, as it requires to be able to provide a well-chosen batch of candidates for parallel evaluation. We call *Acquisition Process* (AP) the way to provide one or multiple candidates. It is usually constituted of one or several AFs and their respective optimization. Finally, we define a *cycle* as the following sequence of instructions: update the surrogate model, perform AP and evaluate the selected candidate(s) simultaneously.

The objective of this paper is to present a new AP based on space partitioning optimization in order to efficiently select a large and valuable batch of candidates from the surrogate model that will be exactly evaluated in parallel. After presenting related works and precisely defining the objectives of the new approach in Section I, the latter is precisely described in Section II. Then, a benchmark analysis is presented in Section III before concluding on benefits of the approach and future works.

I. RELATED WORKS AND OBJECTIVES

Efficient Global Optimization (EGO) is developed by Jones *et al.* in [3]. It uses Gaussian Processes (GP) surrogate models, which has the ability to provide a measure of uncertainty on

the prediction. The Expected Improvement (EI) is used as AF and is designed to find a trade-off between exploring the regions where the surrogate accuracy is low, and exploiting in area where predicted values are close to the best known solution.

A. Related Works

Dealing with parallel BO, one major difficulty is to provide useful candidates to improve either the surrogate model, either the objective value. BO and in particular EGO provide a way by using an AF. However this process is inherently sequential and must be slightly modified to exploit parallel computing. In [4] Ginsbourger *et al.* several approaches for the parallelization of EGO are presented. To select q candidates, the analytical q-points EI (q-EI) is defined and optimized through time-consuming Monte Carlo simulations. An alternative is proposed to emulate q-EI: each time a candidate is provided by the optimization of the AF, the model is updated with an arbitrary chosen objective value attributed to the candidate (i.e. without any access to the real simulator). In this multi-point AP, q points are sequentially selected by repeating the single-point AP. Due to the relevance of the q-EGO algorithm for the present work, it will be further detailed in Section III-A1.

Motivated by a wider access to parallel machines, several other multi-point APs have been designed. Wang *et al.* [5] proposed another way to select q points using infinitesimal perturbation analysis to construct a gradient estimator of the multi-point EI surface coupled with a multi-start procedure to find the set of points to evaluate exactly. The method appears to be fast while providing the bayes-optimal set of points. Another method from Marmin *et al.* [6] introduces the analytical form of the multi-point EI gradient to be able to optimize the function with gradient information. This approach reduces the computational cost of q-points selection compared to sequential heuristics or Monte Carlo sampling of [4].

Nevertheless, it remains costly in high dimensions because it is still a $d * q$ dimensional problem, d being the number of decision variables. Based on the same idea of providing a batch of points at each cycle, Kandasamy *et al.* [7] use Thompson Sampling to maximize the probability of reward (i.e. improving the target). Shah *et al.* [8] extend the Entropy Search from Hennig *et al.* [9] for parallel computation. In [10] and [11] niching Genetic Algorithms are used to select simultaneously ensemble of points. Another approach consists in using different complementary AFs or ensembles of surrogate models. Wang *et al.* [12] uses n infill criteria coupled to multi-point proposal inspired by the Kriging Believer heuristic from [13] to get $n * q$ candidates per cycle. Lyu *et al.* [14] and Feng *et al.* [15] use multi-objective optimization on different AFs to get a Pareto set of optimal candidates.

B. Objectives

Many methods have been developed trying to either associate different models or AFs to propose several points, either optimizing the multi-point version or a single criterion. However, APs based on partitioning the decision space have only been sparsely addressed in the literature. We can mention Villanueva *et al.* [16] in which a multi-agent strategy is used to explore different subdomains independently, following the machine learning formalism of reinforcement learning. Agents that performs well are rewarded and possibly split while other may merge. Wang *et al.* uses clustering techniques in [17] to locate distinct local optima.

Parallel versions of EGO-like algorithms often suffer from a time-consuming AP or costly metamodel updates, and a major difficulty stays in balancing the optimization process [18]. Trying to push these limits back, we propose a new algorithm which is able to select large batches of points to evaluate in parallel in a moderate amount of time, while keeping a balance between exploration and exploitation. We use a space partition managed by a self-organizing binary tree in order to perform simultaneously different local APs in each subdomain. The Binary Space Partitioning tree (BSP tree) allows to decompose the global AP in several ones. It also balances the optimisation process using a decision heuristic dealing with where to intensify the decomposition process, and where to sample less frequently. The strength of this method lies in its adaptability. It is able to provide as many candidates as needed, remains fast to execute and parallelizable (sub-APs are independent from each other). Finally, the partition tree adapts automatically according to the rule explained in Section II-B2.

II. Presentation of the method

As for many BO algorithm, the method rests on the Expected Improvement (EI) criterion. To be computed, EI requires that the metamodel provides a value of the uncertainty of the prediction. Thanks to its statistical nature, the Kriging equations provide both prediction of the output at the desired location and its variance.

A. Metamodel

Kriging intends to predict the value of a certain quantity according to its location. It uses spatial correlation between the locations and their measure of interest (the output) to build the Best Linear Unbiased Predictor. The theoretical basis of the Kriging model will not be presented here, but can be found for example in [19]. Nevertheless some parameters need to be defined for better understanding. Let us call $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathcal{D} \subset \mathbb{R}^d$ previously mentioned as the location, and $y(\boldsymbol{x}) \in \mathbb{R}$ the objective value at this specific location. The value is assumed to be a random variable that writes as the sum of a trend function $\mu(\boldsymbol{x})$ and a centered gaussian process $z(\boldsymbol{x})$: $y(\boldsymbol{x}) = \mu(\boldsymbol{x}) + z(\boldsymbol{x})$. Denoting $y(\boldsymbol{x}^{(i)}) = y^{(i)}$, we write $\boldsymbol{y} = (y^{(1)}, \dots, y^{(n)})$ the known values of the outputs at locations $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(n)})$. For this study, the trend is chosen to be constant, this kind of kriging is also known as Ordinary Kriging. The spatial correlation is expressed between two points using a kernel function that needs to be defined as an hyper-parameter of the model.

The Kriging prediction is the Best Linear Unbiased Predictor, hence the prediction of the output $\hat{y}(\boldsymbol{x}^*) = \hat{y}^*$ at

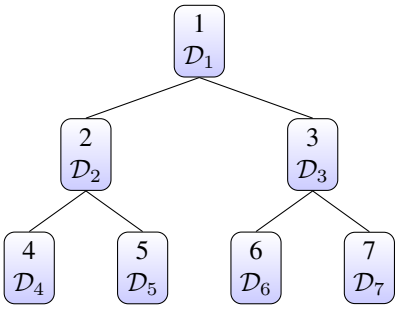


Fig. 1: Partitioning of $\mathcal{D}$ through BSP tree

location $\boldsymbol{x}^*$ must satisfy $\hat{y}^* = argmin(\mathbb{V}ar[\hat{y}^* - y^*])$ such that $\mathbb{E}[\hat{y}^* - y^*] = 0$. Solving this optimization problem corresponds to the fitting of the Kriging model. Finally, the model provides the prediction $\hat{y}(\boldsymbol{x}^*)$ and its associated uncertainty $\sigma(\boldsymbol{x}^*)$.

B. Acquisition Process

The acquisition process is dedicated to find a batch of points to simultaneously evaluate with the objective function (e.g. the simulator). The particularity of our AP is that it is decomposed in several sub-processes responsible for providing one candidate each.

1) Binary Space Partition: In the proposed AP, the search space $\mathcal{D}$ is recursively decomposed according to a binary separation of space. Let us suppose that the whole domain is $\mathcal{D}_1$. The partitions are managed by a binary tree where the root node (at tree level 1) contains $\mathcal{D}_1$. The next level (level 2) has two nodes (node 2 and node 3) that contain $\mathcal{D}_2$ and $\mathcal{D}_3$ such that $\mathcal{D}_1 = \mathcal{D}_2 \cup \mathcal{D}_3$. For each node k, the property is respected such that $\mathcal{D}_k = \mathcal{D}_{2k} \cup \mathcal{D}_{2k+1}$ and thus for each level of the tree, the union preserves the entire domain.

An example is presented in Fig. 1, where the domain is split in four subdomains. Let us call $\mathcal{F}_n$ the family of subdomain indexes at cycle n, such that $\bigcup_{k \in \mathcal{F}_n} \mathcal{D}_k = \mathcal{D}$ and $\forall i, j \in \mathcal{F}_n \; \mathcal{D}_i \cap \mathcal{D}_j = \emptyset$. Regarding Fig. 1, $\mathcal{F}_n = \{2, 6, 7\}$ and $\mathcal{F}_n = \{3, 4, 5\}$ are acceptable sets. Thanks to this kind of decomposition, it is easy to perform one sub-AP in each subdomain, while keeping knowledge of the entire domain. As soon as all the candidates are computed, they are collected and sorted according to the chosen figure of merit. A subset of them will be exactly evaluated while the rest is discarded. We call the number of candidates evaluated in parallel the *batch size* n_{batch}.

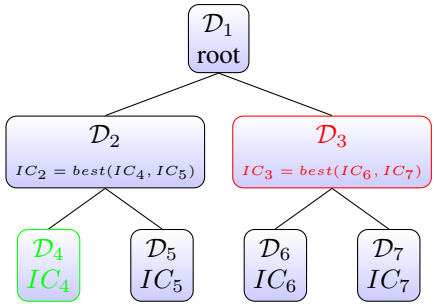
This strategy intends to reinforce the global aspect of optimization by keep sampling in *a priori* less interesting areas of $\mathcal{D}$ (from the IC point of view). Nevertheless, in order to avoid sampling with clearly no gain, and thus waste computational power, the total number of candidates will be greater than the batch size. Furthermore, to balance the exploration and exploitation process, the tree has the possibility to evolve and split further some nodes resulting in intensifying search into the best subdomain - always in terms of the chosen figure of merit.

Even though the subdomains are distinct, it may happen that several candidates are very close to a shared boundary, and thus to each other. In that scenario, the candidates receive a small random perturbation so that the area is still sampled twice, but most of all the stability of the kriging model is preserved.

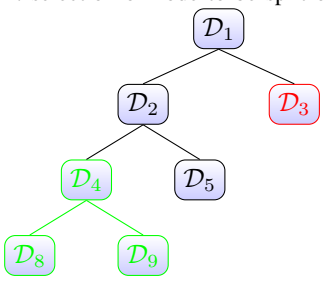
2) Evolution strategy: For this study, the number of candidates provided by the AP before the selection is a multiple of the batch size. Indeed, the batch size is fixed equal to the number of available cores (1 evaluation per core), thus each computing unit performs the same number of sub-APs and the parallel load is balanced. As stated in the previous paragraph, one candidate is chosen in each subdomain, thus we have as many candidates as leaves in the tree. Let us call n_{leaves} that number, and n_{batch} the batch size. Consequently $n_{leaves} = r * n_{batch}$ where $r \in \mathbb{N}_{\setminus\{0\}}$.

The tree is updated once a cycle to take into account the new information. The supposed best subdomain, according to the IC, is decomposed further to intensify the search in this area. Nevertheless, as we decide to keep n_{leaves} constant, this splitting step is only performed if two domains are merged. In terms of the BSP-tree, the leaf with the highest figure of merit is split, and the parent node with the lowest one loses its leaves to become a leaf itself. This process is illustrated in Fig. 2.

This example directly follows the one from Fig. 1, one candidate is chosen in each leaf (i.e the ones indexed by $\mathcal{F}_0 = \{4, 5, 6, 7\}$). Each node is attributed the best figure of merit of its children, this number is denoted in Fig. 2 by IC_{node}. In Fig. 2a, $\mathcal{D}_4$ possesses the best value among the nodes indexed by $\mathcal{F}_0$. Consequently, the node will be split if



(a) Step 1: selection of node to be split or merged



(b) Step 2: Allowed operations, update of the tree

Fig. 2: Illustration of one tree update

Algorithm 1 BSP-EGO pseudo-code

Input
f: function to optimize
n_{init}: initial sample size
n_{cycle}: number of cycles
$\mathcal{D}$: search space
$\mathcal{T}$: tree

1: $\boldsymbol{X} \leftarrow$ initial_sampling$(\mathcal{D}, n_{init})$
2: $\boldsymbol{y} \leftarrow f(\boldsymbol{X})$
3: $\mathcal{T} \leftarrow$ build_tree$(depth)$
4: **for** i in $1 : n_{cycle}$ **do**
5: $model \leftarrow$ learn_model$(\boldsymbol{X}, \boldsymbol{y})$
6: $\mathcal{B}_{candidates} \leftarrow \varnothing$
7: **for** $leaf$ in $\mathcal{F}_i$ **do** ▷ parallelizable loop
8: $\boldsymbol{c} \leftarrow$ find_best_candidate$(\mathcal{D}_{leaf})$
9: $\mathcal{B}_{candidates} \leftarrow \mathcal{B}_{candidates} \cup \boldsymbol{c}$
10: **end for**
11: $\mathcal{B}_{candidates} \leftarrow$ selection$(\mathcal{B}_{candidates})$
12: $(\mathcal{T}, \mathcal{F}_{i+1}) \leftarrow$ update_tree$(\mathcal{T}, \mathcal{B}_{candidates}, \mathcal{F}_i)$
13: $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \mathcal{B}_{candidates}$
14: $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup f(\mathcal{B}_{candidates})$ ▷ parallel evaluation
15: **end for**
16: **return** $\boldsymbol{x}_{max}$

the merge operation can be performed. Regarding the parents of the leaves, $\mathcal{D}_3$ possesses the worst value, meaning that area does not need as many attention and thus it will be merged. Eventually, $\mathcal{F}_1 = \{3, 5, 8, 9\}$.

In case of a non-allowed operation, the tree is kept identical for the next cycle. For instance, regarding Fig. 2b, it may happen that for the next cycle $\mathcal{D}_3$ is still the worst subdomain and can't be merged with $\mathcal{D}_2$. However, this kind of exception is relatively rare when dealing with large trees.

3) Splitting the subdomain: Without loss of generality, let us set $\mathcal{D}_1 = [0, 1]^d$. We must decide how to separate the hypercube. For practical reasons, the choice is made for now to keep hyper-rectangular domains, so that a splitting operation is characterized by the axis/dimension to be split, and the range at which the section is done. For example, if $\mathcal{D}_1 = [0, 1]^2$ is split according to the first axis in the middle of the segment, it comes to $\mathcal{D}_2 = [0, 0.5] \times [0, 1]$ and $\mathcal{D}_3 = [0.5, 1] \times [0, 1]$. The heuristic for this study is arbitrary splitting: based on the idea that it is preferable to have dimensions of the same order of magnitude, axes are split one after another in a cyclic way. The chosen axis is determined by the level of depth of the tree node, i.e. the axis along which a subdomain is split is given by: $d_{split} = depth(node) \bmod(d)$. The initial tree is formed using this heuristic until it reaches the desired depth.

C. BSP-EGO

Named after EGO algorithm, this methods is called BSP-EGO. It uses a global kriging metamodel and the presented BSP-based AP with one sub-AP per sub-region. BSP-EGO is outlined in Algorithm 1. First, the initial sample is created, namely the couple $(\boldsymbol{X}, \boldsymbol{y})$. Then, the tree $\mathcal{T}$ is initialized at a predefined depth, deduced from the user-defined hyper-parameter n_{leaves}. At the beginning of each cycle, starting at line 4 of Algorithm 1, the model is updated and then, the AP begins in line 7. In each leaf of the tree, marked in $\mathcal{F}_i$, a candidate is proposed using classical optimization on the chosen AF. These AF maximisations are independent and can be performed in parallel. Candidates are gathered and the n_{batch} most promising ones, according to the EI criterion, are selected (line 11). Then the tree and the leaves (i.e. their indexes) are updated. A cycle ends with the parallel evaluation of the selected candidates and their insertion into the database (lines 13 and 14). BSP-EGO is implemented in $C++$, compiled with GCC 8.3.0, using OpenMPI 3.1 for the parallel evaluation of candidate solutions and the BayesOpt C++ library [20] for the Kriging metamodel.

III. Benchmark analysis

A. Experimental protocol

In this section, experimental results obtained with BSP-EGO are presented and compared to the q-EGO algorithm from Ginsbourger *et al.* [4]. One implementation is available in the R package `DiceOptim` presented in Roustant *et al.* [21]. All experimentation are performed using the Chetemi cluster from Grid5000 [22]. The cluster is composed of 15 nodes with 2 Intel Xeon E5-2630 of 10 cores each. Among the 300 available cores, only n_{batch} are reserved for experiments.

1) Reference approach: The used q-EGO algorithm is outlined in Algorithm 2 and differs from BSP-EGO in its AP.

Algorithm 2 q-EGO pseudo-code

Input
f: function to optimize
n_{init}: initial sample size
n_{cycle}: number of cycles
$\mathcal{D}$: search space

1: $\boldsymbol{X} \leftarrow$ initial_sampling$(\mathcal{D}, n_{init})$
2: $\boldsymbol{y} \leftarrow f(\boldsymbol{X})$
3: **for** i in $1 : n_{cycle}$ **do**
4: $model \leftarrow$ learn_model$(\boldsymbol{X}, \boldsymbol{y})$
5: $\mathcal{B}_{candidates} \leftarrow \varnothing$
6: $\boldsymbol{X}_{tmp} \leftarrow \boldsymbol{X}$
7: $\boldsymbol{y}_{tmp} \leftarrow \boldsymbol{y}$
8: **for** k in $1 : q$ **do** ▷ sequential loop
9: $\boldsymbol{x}^{(k)} \leftarrow argmax_{\mathcal{D}}(IC(x))$
10: $\mathcal{B}_{candidates} \leftarrow \boldsymbol{x}^{(k)}$
11: $y^{(k)} \leftarrow$ best_value $(\boldsymbol{y})$
12: $\boldsymbol{X}_{tmp} \leftarrow \boldsymbol{X}_{tmp} \cup x^{(k)}$
13: $\boldsymbol{y}_{tmp} \leftarrow \boldsymbol{y} \cup y^{(k)}$
14: $model \leftarrow$ learn_model$(\boldsymbol{X}_{tmp}, \boldsymbol{y}_{tmp})$
15: **end for**
16: $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \mathcal{B}_{candidates}$
17: $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup f(\mathcal{B}_{candidates})$ ▷ parallel evaluation
18: **end for**
19: **return** $\boldsymbol{x}_{max}$

TABLE I: Characteristics of benchmark functions

f	$\mathcal{D}$	$f(\boldsymbol{x}) =$	y_{min}
Ackley	$[-32, 32]^6$	$-20.exp\big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\big) - exp\big(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\big) + 20 + exp(1)$	0
Rosenbrock	$[-32, 32]^6$	$\sum_{i=1}^{d} \big[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\big]$	0
Alpine	$[0, 10]^6$	$\prod_{i=1}^{d} \sqrt{x_i} sin(x_i)$	-490.21

To be able to provide a batch of distinct candidates without accessing the exact evaluation, q-EGO iteratively performs q maximisations of the AF, each followed by a metamodel update with heuristically assigned objective values. Indeed, assigning a value to a candidate allows to update the model and transform the landscape of the infill criterion so that it knows that the area is sampled (or will be), resulting in a low figure of merit for this location. The used heuristic is named *Constant Liar* (CL). It assigns to all candidates the best current objective value. As shown in lines 8 to 15 of Algorithm 2 this process is repeated q times until the batch $\mathcal{B}_{candidates}$ contains the $q = n_{batch}$ candidates. As a consequence, this method involves q sequential updates of the model (line 14), and a temporary learning set denoted as $(\boldsymbol{X}_{tmp}, \boldsymbol{y}_{tmp})$ in Algorithm 2.

Several alternatives to CL are presented in Ginsbourger *et al.* [13] such as *Kriging Believer* where the model is trusted and the *lie* of CL is replaced with the prediction of the model. According to preliminary experiments, CL seems to perform better on our benchmark test-bed. The creation of the model as well as the optimization process are left to the `DiceKriging` package [21] concerning the q-EGO part.

2) Hyper-parameters settings: For both approaches, the metamodel is an Ordinary Kriging model - i.e with constant trend - with Matern$\frac{5}{2}$ covariance kernel. The study is performed with $n_{batch} = 8, 16, 32$.

Concerning BSP-EGO specific parameters, the initial tree is build with $2 * n_{batch}$ leaves and this number is preserved all through the 48 cycles attributed to each experimentation. Only one tree update is allowed per cycle.

For each experimentation, we compare the average over 10 repetitions of the objective value evolution over the 48 cycles. The comparisons are done using the same initial sample containing 64 points generated randomly so that each approach starts with the same best known value at cycle 0.

3) Benchmark functions: The performance analysis is performed on three classical benchmark functions. They are chosen to be representative of current challenges in global optimization. The first one is the Ackley function: it is optimized in the domain $\mathcal{D} = [-32, 32]^6$ and its only global minimum is 0. The Ackley function is also known as the "well-function" since it has (in its 2D landscape) a clear minimum in a well in the middle of a noisy plane. The noise can be difficult to manage when dealing with surrogate models and IC. Second choice is the Rosenbrock function, optimized as well in $\mathcal{D} = [-32, 32]^6$ with 0 as unique global minimum. Also named Valley or Banana function because of its 2D representation, the function is unimodal, and the global minimum lies in the valley. Even though the valley is easy to spot, algorithms may have convergence problems. The last one for this study is the Alpine function, named after its multi-modal 2D landscape. It is optimized in $\mathcal{D} = [0, 10]^6$ and has a single global minimum of $-2.808^d \approx -490.21$. Its difficulty lies in the number of local minima of same amplitude in which the process may be stuck. These functions are summarized in Table I.

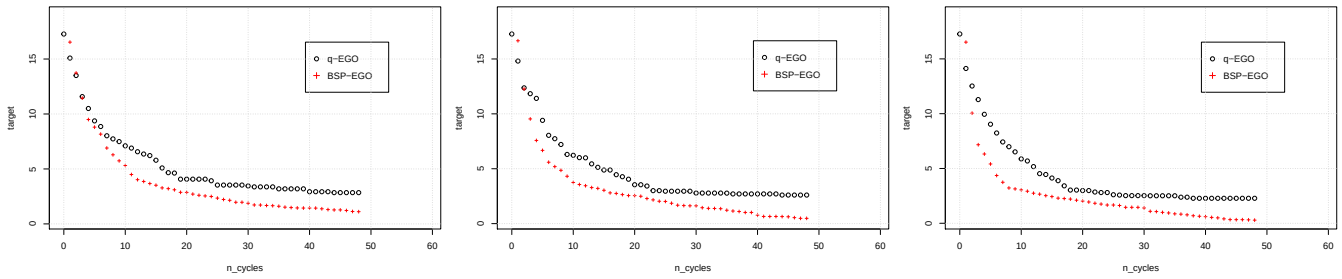
B. Experimental Results

Fig. 3, 4 and 5 show the evolution of the best target (y_{min}) in unction of the current cycle for the three test-functions Ackley, Rosenbrock and Alpine respectively. The black-dotted curves represent the results of the q-EGO reference approach. The results from BSP-EGO approach are labeled as red-crossed curves. Additionally, Fig. 6, 7, 8 show boxplots of the best objective found after 48 cycles.

Fig. 3 shows that the space partitioning approach presents a good behavior regarding the results from the Ackley benchmark function. For each batch size ($n_{batch} \in \{8, 16, 32\}$) the BSP-EGO curve is beneath the q-EGO one, meaning that at any cycle of the optimization, BSP-EGO provides on average a better solution. This is confirmed by the boxplot in Fig. 6 where we can see that almost any execution of BSP-EGO performs better than every q-EGO one. Furthermore, adding parallelism allows to provide a better final value for the same number of cycles. As shown in Fig. 6, while q-EGO stagnates, the BSP approach continues to improve the objective value as n_{batch} increases.

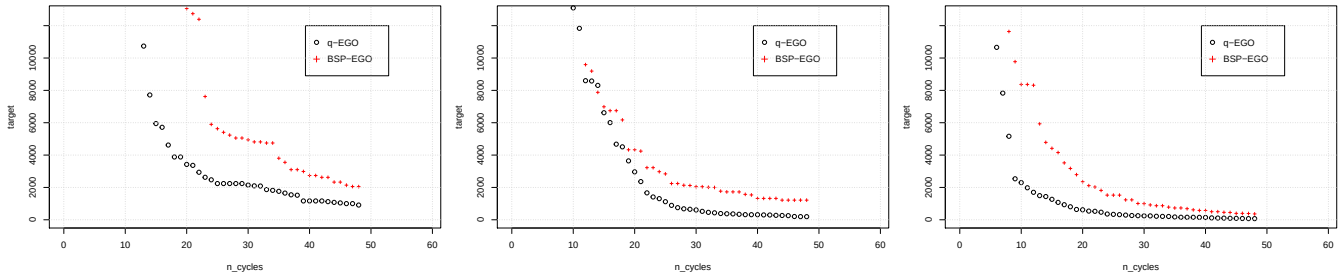
For sake of readability, the figures associated with the Rosenbrock function do not display the entire evolution of the objective value. Indeed the display window is focused on the final values to be able to tell the difference between the curves. Thus, the high output values are not visible for the first cycles.

Fig. 4 displays the results for the Rosenbrock function and shows that the q-EGO curves are always on top of BSP-EGO ones. Even though q-EGO performs better on this benchmark function, it has to be noted that the gap shrinks when increasing the batch size. As for the Ackley function, increasing the batch size and thus the degree of parallelism favors BSP-EGO and enables the latter to find better solutions. The evolution of the best found solution with n_{batch} is clearly



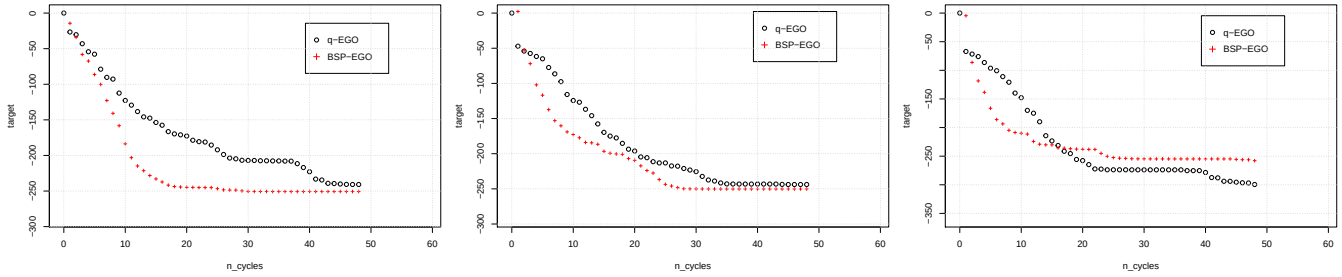
(a) Ackley - $n_{batch} = 8$ (b) Ackley - $n_{batch} = 16$ (c) Ackley - $n_{batch} = 32$

Fig. 3: Evolution of the best found target in function of the cycle - Ackley function



(a) Rosenbrock - $n_{batch} = 8$ (b) Rosenbrock - $n_{batch} = 16$ (c) Rosenbrock - $n_{batch} = 32$

Fig. 4: Evolution of the best found target in function of the cycle - Rosenbrock function



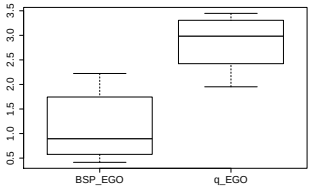
(a) Alpine - $n_{batch} = 8$ (b) Alpine - $n_{batch} = 16$ (c) Alpine - $n_{batch} = 32$

Fig. 5: Evolution of the best found target in function of the cycle - Alpine function

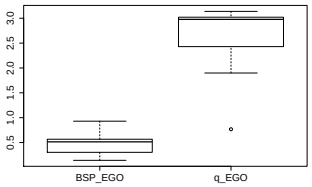
visible in Fig. 7, starting from around few thousands for $n_{batch} = 8$, to a thousand for $n_{batch} = 16$ and finally few hundreds for $n_{batch} = 32$.

Regarding our last benchmark function, Alpine, we can observe in Fig. 5 that BSP-EGO performs better than q-EGO for $n_{batch} = 8, 16$, while for a batch size of 32 q-EGO outperforms its concurrent at the end of the optimization. Regarding the average curves, BSP-EGO appears to be faster than q-EGO to find a local minimum but has difficulties to escape from it. One hypothesis that could explain this behavior is that when increasing the batch size, the number of leaves is also increased, allowing bigger sampling in a relatively small region and therefore over-intensifying the search. Nevertheless, the boxplots in Fig. 8 indicate that the final results of the two approaches are very similar and they both managed to find the global minimum.

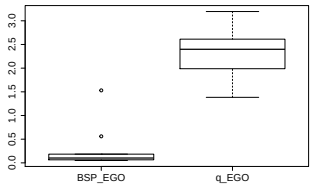
The recursive splitting allows to sample more areas of interest which seems beneficial for the Ackley function where classical BO might face difficulties to exploit the optimal area. Furthermore, increasing the degree of parallelism allows to improve the quality of the best found solution. This is especially visible regarding Fig. 4, which shows that BSP-EGO becomes better with wider parallelism. The two other benchmark functions also seem to benefit from the larger



(a) Ackley - $n_{batch} = 8$

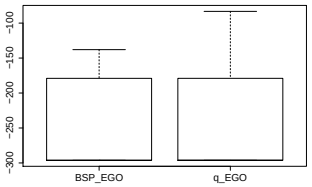


(b) Ackley - $n_{batch} = 16$

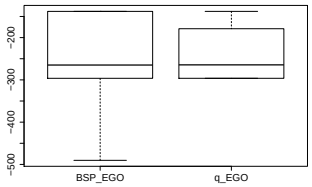


(c) Ackley - $n_{batch} = 32$

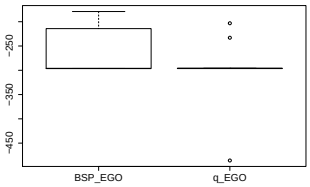
Fig. 6: Box-plots of the final value for BSP-EGO and q-EGO - Ackley function



(a) Alpine - $n_{batch} = 8$

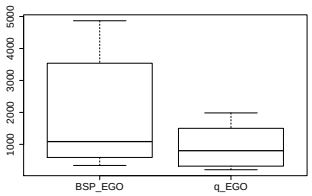


(b) Alpine - $n_{batch} = 16$

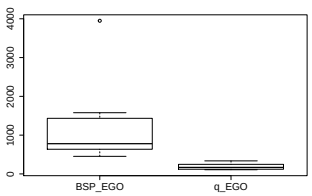


(c) Alpine - $n_{batch} = 32$

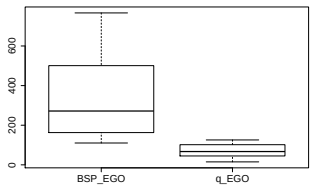
Fig. 8: Box-plots of the final value for BSP-EGO and q-EGO - Alpine function



(a) Rosenbrock - $n_{batch} = 8$



(b) Rosenbrock - $n_{batch} = 16$



(c) Rosenbrock - $n_{batch} = 32$

Fig. 7: Box-plots of the final value for BSP-EGO and q-EGO - Rosenbrock function

batches. The study suggests that the proposed AP can perform as good as q-EGO while being much faster as it requires only one model update instead of q. However, it has to be noted that the cost of fitting the kriging model still becomes prohibitive with the size of the design of experiment which grows especially fast rising the batch size.

IV. Conclusion

BO and parallel computing are common ways to deal with the optimization of highly time-consuming function/simulators. However making them act in concert still raises difficulties. We propose an approach to provide n_{batch} candidates based on a recursive design space decomposition managed by a binary decomposition tree. A performance analyse is led on three benchmark functions chosen to represent known difficulties in global optimization. The proposed method challenges the q-EGO approach from [21]. According to the presented results in Fig 3, 4, 5, the proposed partition-based AP seems to perform fairly compared to the state-of-the-art method q-EGO. In addition, the developed AP presents interesting qualities such as its much smaller execution time, which can still be reduced using parallelism for the AP. In the current version, BSP-EGO only performs the exact evaluations in parallel. This AP allows to propose arbitrarily big batches of valuable points with only a small extra cost.

BSP-EGO is presented in this paper in its first version, without hyper-parameter calibration. Especially parameters from the tree management need to be further investigated (e.g. number of updates or leaves). Heuristics are being developed based on correlation coefficients to determine the most significant axis to chose as split direction. Regarding the figures of merit coming from the n_{batch} candidates we observe that the value of the last candidates are low. This observation is an argument to explore a multi-criteria approach based on complementary AF. Last, we are planning to investigate the presented BSP-approach with different surrogate models, or with a Kriging model fitted on a subset of points to tackle the prohibitive cost of fitting the Kriging model on large data samples and be able to increase the parallel potential of BSP-EGO. All these ideas remain to be exploited and should hopefully improve the newly presented AP.

Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER

and several Universities as well as other organizations (see https://www.grid5000.fr).

REFERENCES

[1] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Fluids Engineering*, 86(1):97–106, 03 1964.

[2] J. Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[3] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998.

[4] Laurent Carraro David Ginsbourger, Rodolphe Le Riche. A multi-points criterion for deterministic parallel global optimization based on gaussian processes. 2008.

[5] Jialei Wang, Scott Clark, Eric Liu, and Peter Frazier. Parallel bayesian global optimization of expensive functions. 02 2016.

[6] Sébastien Marmin, Clément Chevalier, and David Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. pages 37–48, 2015.

[7] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Asynchronous parallel bayesian optimisation via thompson sampling, 2017.

[8] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions, 2015.

[9] Philipp Hennig and Christian Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13, 12 2011.

[10] Hao Wang, Thomas Bäck, and Michael Emmerich. Multi-point efficient global optimization using niching evolution strategy, 06 2014.

[11] Dawei Zhan, Jiachang Qian, and Yuansheng Cheng. Balancing global and local search in parallel efficient global optimization algorithms. *Journal of Global Optimization*, 67(4):873–892, Apr 2017.

[12] Yuan Wang, Zhong-Hua Han, Yu Zhang, and Wenping Song. Efficient global optimization using multiple infill sampling criteria and surrogate models. 01 2018.

[13] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. 2010.

[14] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3306–3314, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[15] Zhiwei Feng, Qingbin Zhang, Qingfu Zhang, Qiangang Tang, Tao Yang, and Yang Ma. A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization. *J Glob Optim*, 61:1–18, 04 2014.

[16] Diane Villanueva, Rodolphe Le Riche, Gauthier Picard, and Raphael Haftka. Dynamic design space partitioning for optimization of an integrated thermal protection system. 04 2013.

[17] G. Gary Wang and Timothy W. Simpson. Fuzzy clustering based hierarchical metamodeling for design space reduction and optimization. *Engineering Optimization*, 36(3):313–335, June 2004.

[18] Guillaume Briffoteaux, Maxime Gobert, Romain Ragonnet, Jan Gmys, Mohand Mezmaz, Nouredine Melab, and Daniel Tuyttens. Parallel surrogate-assisted optimization: Batched bayesian neural network-assisted ga versus q-ego. *Swarm and Evolutionary Computation*, 57:100717, 2020.

[19] Cressie Noel A. C. *Statistics for spatial data / Noel A. C. Cressie,...* Wiley series in probability and mathematical statistics Applied probability and statistics. J. Wiley & Sons, New York [etc, revised edition edition.

[20] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15:3735–3739, 11 2014.

[21] Olivier Roustant, David Ginsbourger, and Yves Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. June 2010.

[22] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.