ARTICLE TEMPLATE

# A novel UCB-based batch strategy for Bayesian optimization

C. Domínguez-Bravo[a] and Jose A. Lozano[a,b]

[a]BCAM – Basque Center for Applied Mathematics, Alameda de Mazarredo, 14, Bilbao, Spain
[b]ISG – Intelligent Systems Group, University of the Basque Country UPV/EHU, Donostia, Spain

**ARTICLE HISTORY**
Compiled January 22, 2019

**ABSTRACT**
The optimization of expensive black-box functions appears in many situations. Bayesian optimization methods have been successfully applied to solve these problems using well-known single-point acquisition functions. Nowadays, the developments in technology allow us to perform evaluations of some of these expensive function in parallel. Therefore, there is a need for batch infill criteria to consider this parallelism.

In this paper, a novel batch infill criterion is presented. Our algorithm, at each batch step, restricts the search space into a finite candidate pool and, from it, selects the batch that maximizes the mutual information with respect to the objective function. Depending on the total number of observations available, the candidate pool will consist of a Latin Hypercube Sampling (exploration) or a Pareto Sampling (trade-off between exploration and exploitation). We compare our strategy with some of the state-of-the-art UCB-based batch approaches for different benchmark objectives, outperforming or obtaining very similar results.

**KEYWORDS**
Bayesian optimization; black–box expensive optimization; multi–objective optimization

## 1. Introduction

In this paper, we address the problem of minimizing an expensive black–box function whose evaluations can be performed in parallel. The challenge of the problem is to find a reasonable good solution within a limited evaluation budget, González et al. (2016).

This kind of problem appears, for instance, in the optimization of high-fidelity emulators Sarkar et al. (2016), where the objective function simulates an input-output relationship. One example, in the context of machine learning, is the hyperparameter tuning setting Carrizosa, Martín-Barragán, and Morales (2014); Snoek, Larochelle, and Adams (2012), where the optimal parameters of a learning algorithm are sought. There are many practical applications that fit into this optimization framework, such as drug design González et al. (2015) or the design of renewable energy systems Sarkar et al. (2016).

Given the characteristics of the problem, model-based optimization methods, and, specifically, Bayesian optimization (BO), have been successfully applied Brochu, Cora,

C. Domínguez-Bravo. Email: carmenanadb@gmail.com

and de Freitas (2010); Knowles and Nakayama (2008). In BO, input points are sequentially selected to be evaluated with the black-box expensive function. At each step, the selection is made by maximizing an auxiliary function called the infill criterion. The process continues sampling points until the evaluation budget is reached. The final goal is to find the optimum of the expensive function in spite of having a limited number of evaluations available.

The auxiliary optimization problem involved in the sampling process is assumed to be solvable using standard optimization techniques and the evaluation of the infill criterion is assumed to be inexpensive compared with the cost of the black-box function. The infill function or algorithm is designed to measure the usefulness of observing a new input point given the previous observations.

In single-point BO, one point (the maximum of the infill criterion) is sought at each iteration with the aim of balancing exploration and exploitation. That is, explore regions where the uncertainty values of the model are high, which is desirable in the early stages of the search, and exploit regions with low uncertainty and low mean values, which is desirable in later stages.

Nowadays, the availability of computing resources or specific machines, allows us to perform parallel evaluations at the same cost as single evaluations. Therefore, the need for multi–points or batch infill criteria has increased. When selecting a batch, once again, the aim isto balance exploration and exploitation, along with enforcing dissimilarity between the points in the batch.

In this paper, a novel batch infill criterion is presented called "maximize mutual information from a candidate pool" (MMIP). Our algorithm, at each batch step, considers a search over a finite pool of candidates (restriction rule) and, then, selects from the candidates the batch that maximizes the mutual information with respect to the objective function (selection rule). Depending on the total number of observations available at each step, the candidate pool will consist of a Latin Hypercupe Sampling (exploration) or a Pareto Sampling (trade-off between exploration and exploitation). As the number of observations increases, the Gaussian process information about the unknown objective function is more reliable and the algorithm attempts to take advantage of this.

Both steps, restriction and selection, comprise complex problems and, as we explain in detail in Section 4, in both of them heuristic procedures are applied to find an approximated solution. For instance, Pareto Sampling consists of a finite approximation of the Pareto-optimal set of the bi-objective problem addressed by the upper confidence bound (UCB) single-point acquisition function (optimization of the mean and variance functions given by the Gaussian process) by means of an evolutionary multi-objective algorithm.

We compare our strategy with some of the state-of-the-art UCB-based batch approaches, outperforming the results for some of the benchmark objective functions tested, and obtaining very similar results in the remaining cases.

The rest of the paper is organized as follows. In Section 2, the optimization problem and the Bayesian optimization methodology are described. In Section 3 the state-of-the-art UCB-based batch strategies are presented. Our proposed batch infill algorithm is explained in Section 4 and Section 5 illustrates the algorithm with a 2-dimensional example. In Section 6 we introduce the experimental settings used. We apply our proposed strategy to benchmark optimization functions and compare the results in Section 7. Note that, in the interest of space, the complete list of experiments are included in the Annexes. The last section is devoted to summarizing the conclusions and further work.

## 2. Bayesian optimization and infill criteria

In this section, the optimization problem is detailed and we introduce the Bayesian optimization (BO) method using a Gaussian process (GP) as the surrogate model.

### 2.1. *Optimization problem*

The objective function, $f : \Omega \subset \mathbb{R}^d \longrightarrow \mathbb{R}$, is unknown but it can be pointwise evaluated at an expensive cost. Each input value is a vector of dimension $d$, we use the notation $\boldsymbol{x} = (x_1, \ldots, x_d) \in \Omega$. We assume that $\Omega$ is a compact set and that each $x_i$ is a box-constrained variable $x_i \in [l_i, u_i]$ for $i = 1, 2, \ldots, d$.

As we are using GPs as the surrogate model, the fact of dealing with normally distributed noise in the output does not incorporate any technical issue. Therefore, from now on, we assume that the output values are noisy, that is, once $\boldsymbol{x} \in \Omega$ is selected, the value $y = f(\boldsymbol{x}) + \epsilon$, with $\epsilon \overset{i.i.d.}{\sim} \mathcal{N}\left(0, \sigma_0^2\right)$, is observed, where $\sigma_0^2$ is the variance of the noise.

The optimization problem consists of finding the input point that minimizes the objective function, that is, finding

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\Omega}{\arg\min}\, f(\boldsymbol{x})\,, \tag{1}$$

given that the evaluation budget is limited to an integer value $\mathsf{T} \in [1, +\infty)$.

As optimization method we use a model-based optimization methodology known as Bayesian optimization. The optimization is carried out in a sequential way. At each step a new point (batch of points) is selected to be evaluated and to update the model with it (them), see Section 2.2 and Figure 1.

In order to choose the new point (batch of points) to be observed, instead of performing a local search directly on the mean surface of the surrogate model, a specific infill criterion (function and/or algorithm) is designed to be maximized. See Section 2.3.

In general, at step $t$, when $t$ points have been observed, we denote the next point to be observed from the search space by $\boldsymbol{x}_{t+1}$, and its corresponding (noisy) observation by $y_{t+1}$. In the batch case, we denote the next batch of points to be observed by $\mathsf{B}_b = \{\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+q}\} \subset \Omega$, where $b$ denotes the batch step and $q$ the batch size.

### 2.2. *Bayesian optimization*

In Bayesian optimization (BO) a stochastic process is used as a surrogate model. The stochastic process defines a probability distribution over functions which expresses our prior beliefs about the space of functions where the objective function lies. As we already mentioned, in this paper we use a GP as the surrogate model.

A stochastic process is said to be a GP if any finite number of random variables have a joint Gaussian distribution. A GP is completely defined by its mean $m(\boldsymbol{x})$ and its covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ (kernel) $\forall \boldsymbol{x}, \boldsymbol{x}' \in \Omega$, and it is denoted as $\mathcal{GP}\left(m, k\right)$ . Selecting appropriate mean and kernel functions is a complex task. We do not enter into the details here, see Rasmussen and Williams (2006); Morar, Knowles, and Sampaio (2017) for further information about this topic.

The use of GPs implies smoothness assumptions over the modelled function because the kernel function assumes, in general, that closed locations are highly correlated.

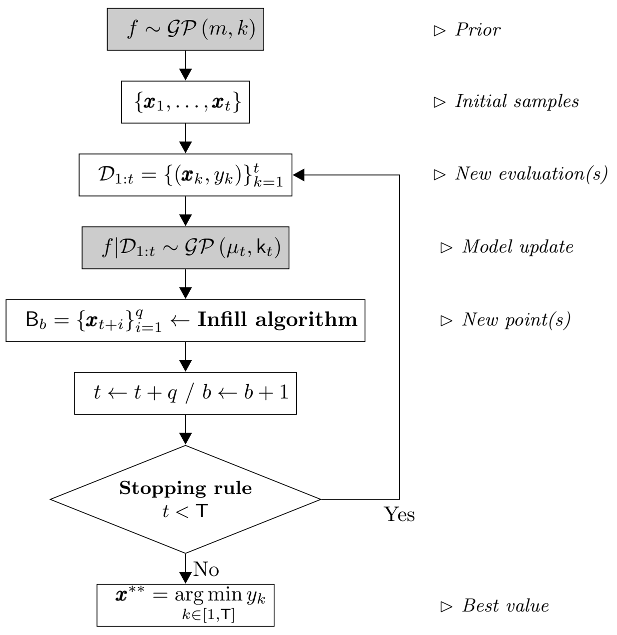
$f \sim \mathcal{GP}(m,k)$
$\triangleright$ Prior
$\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_t\}$
$\triangleright$ Initial samples
$\mathcal{D}_{1:t} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^t$
$\triangleright$ New evaluation(s)
$f|\mathcal{D}_{1:t} \sim \mathcal{GP}(\mu_t, \mathsf{k}_t)$
$\triangleright$ Model update
$\mathsf{B}_b = \{\boldsymbol{x}_{t+i}\}_{i=1}^q \leftarrow$ Infill algorithm
$\triangleright$ New point(s)
$t \leftarrow t+q \;/\; b \leftarrow b+1$
Stopping rule
$t < \mathsf{T}$
Yes
No
$\boldsymbol{x}^{**} = \arg\min_{k\in[1,\mathsf{T}]} y_k$
$\triangleright$ Best value

**Figure 1.** Bayesian optimization

Despite this, GPs have been widely used in BO due to their nice theoretical properties and the good results obtained in many practical applications. See Brochu, Cora, and de Freitas (2010) and Rasmussen and Williams (2006) for an exhaustive description.

We assume that the black-box function is a sample path of the GP, that is, $f \sim \mathcal{GP}(m,k)$. For each input point, the model gives us two quantities: the mean value and the uncertainty value of the prediction, $f(\boldsymbol{x}) \sim \mathcal{N}(m(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ with variance function $\sigma(\boldsymbol{x}) = \sqrt{k(\boldsymbol{x},\boldsymbol{x})}$.

Once some points have been observed, $\mathcal{D}_{1:t} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$, the posterior distribution can be calculated. In our setting the posterior process is, once again, a GP and its corresponding posterior mean and covariance functions can be obtained using the closed formulas given below, which are adapted to handle noisy observations, see Rasmussen and Williams (2006) for more details.

For simplicity we use the notation: $f \,|\, \mathcal{D}_{1:t} \sim \mathcal{GP}(\mu_t, \mathsf{k}_t)$ with $\mu_t(\boldsymbol{x}) = m(\boldsymbol{x} \,|\, \mathcal{D}_{1:t})$ and $\mathsf{k}_t(\boldsymbol{x},\boldsymbol{x}') = k(\boldsymbol{x},\boldsymbol{x}' \,|\, \mathcal{D}_{1:t})$. The posterior mean and variance function are calculated as follows

$$\mu_t(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{X}_{1:t})^\intercal \left(k(\boldsymbol{X}_{1:t}, \boldsymbol{X}_{1:t}) + \sigma_0^2 I_n\right)^{-1} Y_{1:t} \qquad \forall \boldsymbol{x} \in \Omega\,, \tag{2}$$

$$\mathsf{k}_t(\boldsymbol{x},\boldsymbol{x}') = k(\boldsymbol{x},\boldsymbol{x}') - k(\boldsymbol{x}, \boldsymbol{X}_{1:t})^\intercal \left(k(\boldsymbol{X}_{1:t}, \boldsymbol{X}_{1:t}) + \sigma_0^2 I_t\right)^{-1} k(\boldsymbol{x}', \boldsymbol{X}_{1:t}) \quad \forall \boldsymbol{x}, \boldsymbol{x}' \in \Omega\,, \tag{3}$$

where $\boldsymbol{X}_{1:t} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_t]^\intercal \in \mathbb{R}^{t\times d}$ denotes a transposed vector of input values and $Y_{1:t} = [y_1, \dots, y_t]^\intercal \in \mathbb{R}^t$ denotes a vector of output values. The covariance values are denoted in three different ways: scalar $k(\boldsymbol{x},\boldsymbol{x}') \in \mathbb{R}$, vector $k(\boldsymbol{x}, \boldsymbol{X}_{1:t}) = [k(\boldsymbol{x},\boldsymbol{x}_1), \dots, k(\boldsymbol{x},\boldsymbol{x}_t)]^\intercal \in \mathbb{R}^t$ and matrix $k(\boldsymbol{X}_{1:t}, \boldsymbol{X}_{1:t}) \in \mathbb{R}^{t\times t}$ where each matrix entry $(i,j)$ corresponds with the scalar value $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\forall\, i,j \in \{1,\dots,t\}$. Note that $\sigma_0$ denotes the variance of the normally distributed noise and $I_t$ denotes the identity matrix of size $t$.

### 2.3. *Infill criterion*

In the single-point case, the infill criterion has to handle the trade-off between the desire of exploration and exploitation. It is usually expressed through a function called acquisition function $\mathsf{a} : \Omega \to \mathbb{R}^+$. Once $t$ points have been observed, the next point to be evaluated is the one that maximizes the acquisition function,

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\Omega} \mathsf{a}\left(\boldsymbol{x} \,|\, \mathcal{D}_{1:t}\right) \tag{4}$$

In the batch case, in order to emphasize the batch diversity, a modified infill criterion which rewards dissimilarity between the points in the batch needs to be defined.

Usually, batch infill criteria are based on well-known single-point acquisition functions. The goal of a batch infill criterion based on a specific single-point acquisition function is to imitate it for the batch case. Given any single-point acquisition function, if one attempts to directly obtain the batch by solving problem (4) $q$ times, all the points will be the same. Similarly, if we calculate directly the best $q$ different points, all the points will be very close due to the smoothness of the GP model.

Many original batch infill criteria and/or algorithms have been proposed in the literature to address this problem. The goal of this paper is to design an efficient batch infill criterion which uses as few evaluations of the costly function as possible in the task of finding the optimum. Before presenting our strategy in Section 4, we describe in Section 3 some single-point and batch infill criteria appearing in the literature.

## 3. State-of-the-art UCB-based strategies

We focus on a well-known single-point acquisition function, the Upper Confidence Bound (UCB) function. In this section the UCB function and some batch infill criteria appearing in the literature and related with the UCB criterion are briefly described.

In practice, the solutions considered are calculated as approximated solutions of the infill criterion using any standard heuristic optimization procedure. Moreover, in some strategies, infill heuristic algorithms are specifically created to find better solutions or to speed up the search. Note that, in some proposals, the infill criterion is inherent in the infill algorithm.

### 3.1. *UCB single-point acquisition function*

The Upper Confidence Bound (UCB) was originally presented in Cox and John (1997), using the notation from Srinivas et al. (2012):

$$\mathsf{a}^{UCB}(\boldsymbol{x} \,|\, \mathcal{D}_{1:t}) = -\mu_t(\boldsymbol{x}) + \kappa_{t+1}\sigma_t(\boldsymbol{x})\,. \tag{5}$$

From now on, we will use the following notation for the UCB one-step lookahead solution,

$$\boldsymbol{x}_{t+1}^{UCB} = \arg\max_{\boldsymbol{x}\in\Omega} -\mu_t(\boldsymbol{x}) + \kappa_{t+1}\sigma_t(\boldsymbol{x})\,. \tag{6}$$

The weight parameter $\kappa_t$ is used to balance the mean and uncertainty. Traditionally, the value of the weight parameter has been empirically fixed. More recently, in Srinivas et al. (2012), the GP-UCB function has been proposed in the framework of multi-armed bandits. In this strategy the weight parameter is adapted at each step in order to maintain conservative confidence intervals (i.e., guarantee, with high probability, that the value of the function lies inside the confidence interval). The authors also develop in Srinivas et al. (2012) theoretical results for this strategy. Note that in Žilinskas and Calvin (2018), the authors developed some general theoretical properties under some specific assumptions for the same problem and propose to apply a bi-objective optimization approach different from the classical UCB strategy.

### 3.2. *Batch UCB-based strategies*

In the following we list 6 UCB-based batch strategies selected from the literature. The first five are sequential strategies and the last one selects the $q$ points in parallel.

From now on, we use the notation $\mathsf{B}_b = \{\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+q}\} \subset \Omega$ for a batch of points calculated at step $b$ where $b = 0, 1, 2, \ldots$ denotes the batch step ($b = \lfloor t/q \rfloor$), $q$ denotes the batch size, and $t = 1, 2, \ldots, \mathsf{T}$ denotes the pointwise step. Note that the size of the last batch depends on the amount of evaluations available.

(1) B-UCB Desautels, Krause, and Burdick (2014). The authors propose the following sequential batch infill criteria:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\Omega} -\mu_t(\boldsymbol{x}) + \kappa_{t+1}\sigma_t(\boldsymbol{x})\,, \tag{7}$$

$$\boldsymbol{x}_{t+i} = \arg\max_{\boldsymbol{x}\in\Omega} -\mu_t(\boldsymbol{x}) + \kappa_{t+1}\sigma_t(\boldsymbol{x} \,|\, \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+i-1}) \quad \text{for } i = 2, \ldots, q\,. \tag{8}$$

This strategy repeats the GP-UCB single-point strategy by exploiting the fact that the GP variance can be updated once a new point is added to the batch without being observed, see Eq. (3).

(2) PE-UCB Contal et al. (2013); Contal (2016). The Pure Exploration-UCB sequentially selects the points of the batch using the following infill criteria:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\Omega} -\mu_t(\boldsymbol{x}) + \kappa_{t+1}\sigma_t(\boldsymbol{x})\,, \tag{9}$$

$$\boldsymbol{x}_{t+i} = \arg\max_{\boldsymbol{x}\in\Omega} \sigma_t(\boldsymbol{x} \,|\, \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+i-1}) \text{ for } i = 2, \ldots, q\,. \tag{10}$$

In order to calculate an approximate solution, the authors propose to reduce the search space by focusing on a region called "relevant region" which is defined as follows:

$$\mathcal{R}_t = \{\boldsymbol{x} \in \Omega \,:\, \mu_t(\boldsymbol{x}) + 2\kappa_{t+1}\sigma_t(\boldsymbol{x}) \geq \hat{y}_t\} \tag{11}$$

where $\hat{y}_t = \max_{\boldsymbol{x}\in\Omega} -\mu_t(\boldsymbol{x}) - \kappa_t\sigma_t(\boldsymbol{x})$.

We will use the following notation for the UCB two-step lookahead solutions,

$$\boldsymbol{x}_{t+2}^{UCB} = \arg\max_{\boldsymbol{x}\in\Omega} -\mu_t\left(\boldsymbol{x} \,|\, \boldsymbol{x}_{t+1}^{UCB}, y_{t+1}^{UCB}\right) + \kappa_{t+2}\sigma_t\left(\boldsymbol{x} \,|\, \boldsymbol{x}_{t+1}^{UCB}, y_{t+1}^{UCB}\right)\,. \tag{12}$$

In the relevant region defined by the authors, the two following points, the two-step lookahead solution $\boldsymbol{x}_{t+2}^{UCB}$ and the unknown optimum $\boldsymbol{x}^*$ (1) (see Eq. (1)), lie inside with high probability. Note that $\boldsymbol{x}_{t+2}^{UCB}$ cannot be calculated explicitly at step $t$ because the value $y_{t+1}^{UCB} = f(\boldsymbol{x}_{t+1}^{UCB}) + \epsilon_{t+1}^{UCB}$ is unknown.

(3) LP-UCB González et al. (2016). The Local Penalization-UCB consists of an infill algorithm where the batch points are sequentially selected as follows:

$$\boldsymbol{x}_{t+i} = \arg\max_{\boldsymbol{x}\in\Omega} g(\mathsf{a}(\boldsymbol{x} \mid \mathcal{D}_{1:t})) \prod_{j=1}^{i-1} \Psi(\boldsymbol{x} \mid \boldsymbol{x}_{t+j}) \text{ for } i = 1, \ldots, q\,, \tag{13}$$

Function $g$ modifies the selected acquisition function $\mathsf{a}(\boldsymbol{x})$ to be always positive and each penalization term $\Psi(\cdot \mid \boldsymbol{x}_{t+j})$ transmits the expected local reduction of the variance due to the previous batch element $\boldsymbol{x}_{t+j} \in \Omega$.

This approach assumes that the function is Lipschitz continuous or that the kernel is twice differentiable.

(4) PRED-UCB González et al. (2016). The Batch Predictive UCB strategy has been used for comparison purposes. The elements of the batch are sequentially selected as follows:

$$\begin{aligned}\boldsymbol{x}_{t+i} = \arg\max_{\boldsymbol{x}\in\Omega} &- \mu_t(\boldsymbol{x} \mid \{(\boldsymbol{x}_{t+j}, \mu_t(\boldsymbol{x}_{t+j}))\}_{j=1}^{i-1}) + \\ &+ \kappa_t \sigma_t(\boldsymbol{x} \mid \{(\boldsymbol{x}_{t+j}, \mu_t(\boldsymbol{x}_{t+j}))\}_{j=1}^{i-1}) \text{ for } i = 1, \ldots, q\end{aligned} \tag{14}$$

That is, once a new point is selected, a fake update of the GP is performed considering the value of the predicted mean as the true value of the function.

(5) RAND-UCB González et al. (2016). In this strategy, at each batch step, the 1st point is selected by maximizing the UCB function and the remaining points are selected uniformly at random from the search space.

(6) $\lambda-$LCB Hutter, Hoos, and Leyton-Brown (2012). The approach consists of selecting solutions of the bi-objective problem defined by minimizing the mean and maximizing the variance (see Section (4.1.1) by applying the single-point UCB function with different weight parameters.

$$\boldsymbol{x}_{t+i} = \arg\max_{\boldsymbol{x}\in\Omega} -\mu_t(\boldsymbol{x}) + \kappa_i \sigma_t(\boldsymbol{x}) \text{ for } i = 1, \ldots, q \text{ and } \kappa_i \sim Exp(1)\,. \tag{15}$$

Note that $\kappa_i$ are sampled uniformly at random from an exponential distribution of mean 1 for $i = 1, \ldots, q$.

Note that, due to the lack of available code or details to reproduce it, we have excluded from the comparison the following three batch strategies that are also related with the multi-objective framework MOI-MBO (Multi-Objective Infill for parallel Model Based Optimization see Bischl et al. (2014)), EGO-MO (Efficient Global Optimization Multi-Objective infill algorithm see Feng et al. (2015)) and SOP (parallel Surrogate global Optimization with Pareto center selection batch infill algorithm see Krityakierne, Akhtar, and Shoemaker (2016)).

## 4. MMIP–Maximize Mutual Information from a candidate Pool

In this section, our proposed strategy called "maximize mutual information from a candidate pool" (MMIP) is presented. This algorithm uses as the batch dissimilarity function the mutual information with respect to the black-box function.

At each batch iteration of the Bayesian process, our infill algorithm has two main steps, the restriction of the search space into a finite pool of candidates (restriction rule) and the selection of the batch that maximizes the mutual information with respect to the objective function (selection rule).

Moreover, the restriction rule has two different possibilities depending on the step of the Bayesian optimization process, that is, number of observations available. When there are a low number of observations, the restriction rule used is a simple discretization method called Latin Hypercube Sampling (LHS). As more evaluations are used to fit the model, a more complex approach to select the candidate pool is used called Pareto set approximation sampling (PS).

The two rules (restriction and selection) are explained in detail in Section 4.1 and 4.2 respectively. In order to illustrate our approach, in Section 5 we use a demonstrative example by applying our method to minimize the well-known Branin 2-dimensional test function.

The MMIP infill algorithm can be written as follows:

$$\mathsf{B}_b = \underset{\mathsf{B}\subset\mathsf{P}_b\,,\,|\mathsf{B}|=q}{\arg\max}\ \mathsf{MI}(f;\mathsf{B}\,|\,\mathcal{D}_{1:t})\,. \tag{16}$$

where $\mathsf{P}_b$ denotes the candidate pool selected from the search space (depending on the phase, it will be a LHS or a PS) and $\mathsf{MI}(f;\mathsf{B})$ denotes the mutual information with respect to the black-box function when observing the set $\mathsf{B}$. The pseudo-code is presented in Alg. 1.

### 4.1. *Restriction rule*

The restriction rule consists of reducing the search space into a finite set of points called the candidate pool denoted as $\mathsf{P}_b$, with $|\mathsf{P}_b| \leq \mathsf{n}$. As previously mentioned, we consider a different approach depending on the number of observations available.

At steps $1 \leq t \leq \mathsf{T}_*$ (initial steps), we assume that exploration is more significant as the Gaussian process is not mature enough. Therefore, we use as the candidate pool a discretization of $\Omega$ which does not use any information given by the model. We propose applying a Latin Hypercube Sampling (LHS) with $\mathsf{n}$ points. Space-filling designs such as LHS were suggested for this kind of problems in Jones, Schonlau, and Welch (1998).

As the number of observations increases, at steps $\mathsf{T}_* < t \leq \mathsf{T}$, the Gaussian process is assumed to better represent the shape of the objective function. In this case, the restriction rule method uses the information given by the GP through the mean and variance functions. We propose applying a sampling method that we call Pareto set approximation sampling (PS). The aim is to approximate the Pareto-optimal set of the bi-objective problem defined as the optimization of the conflicting objectives: minimizing the mean and maximizing the variance given by the Gaussian Process. The bi-objective approach and all the concepts related with multi-objective optimization are briefly described in Section 4.1.1.

**Algorithm 1** `MMIP`

```
Require: Batch size $q$, budget $\mathsf{T}$, budget break $\mathsf{T}_*$, candidate pool size $\mathsf{n}$, PS parameters $\theta_{PS}$, GP parameters $\theta_{GP} = (q_0, \mu_0, k, l)$.
 1: $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{q_0}\} \subset \Omega$                              ▷ Initial samples
 2: $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ for $i = 1, \dots, q_0$
 3: $\mathcal{D}_{1:q_0} \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{q_0}$                         ▷ Initial observations
 4: $t \leftarrow q_0$
 5: while $t < \mathsf{T}$ do
 6:     $\mathcal{GP}(\mu_t, \mathsf{k}_t) = \mathcal{GP}(\mu_0, \mathsf{k}_0) \,|\mathcal{D}_{1:t}$                 ▷ GP update
 7:     $b \leftarrow \lfloor (t - q_0)/q \rfloor$
 8:     if $t < \mathsf{T}_*$ then                                     ▷ Two-Phases
 9:         $\{p_1, \dots, p_{\mathsf{n}}\}$ LHS                                ▷ Restriction rule(s)
10:     else
11:         $\{p_1, \dots, p_{\mathsf{n}}\}$ PS with NSGA–II($\theta_{PS}$)
12:     end if
13:     $\mathcal{P}_b \leftarrow \{p_1, \dots, p_{\mathsf{n}}\}$
14:     $\mathsf{B}_b \leftarrow \underset{\mathsf{B} \subset \mathcal{P}_b, |\mathsf{B}| = q}{\arg\max} \ \mathsf{MI}(f; \mathsf{B} \,|\, \mathcal{D}_{1:t})$ with Greedy MI Alg. 2      ▷ Selection rule
15:     $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ with $\boldsymbol{x}_i \in \mathsf{B}_b$ for $i = t+1, \dots, t+q$
16:     $\mathcal{D}_{1:t+q} \leftarrow \mathcal{D}_{1:t} \cup \{(\boldsymbol{x}_i, y_i)\}_{i=t+1}^{t+q}$                ▷ New observations
17:     $t \leftarrow t + q$
18: end while
19: $\boldsymbol{x}^+ \leftarrow \boldsymbol{x}_k$ corresponding with $y_k = \underset{i \in [1, \mathsf{T}]}{\arg\min}\, y_i$              ▷ Best value observed
20: return $\boldsymbol{x}^+$
```

### *4.1.1. Multi-objective optimization*

In the following, we introduce several concepts related with multi-objective optimization that, subsequently, will be used in the explanation of the batch infill criteria.

Multi-objective (MO) optimization involves a set of conflicting evaluation criteria $\{s_1(\cdot), s_2(\cdot), \dots, s_k(\cdot)\}$ where compromise solutions are sought. This paper only addresses bi-objective problems, so we simplify the notation to the case $k = 2$. Let $s_1, s_2 : \Omega \to \mathbb{R}$ be two conflicting criteria to be maximized. We define the vector objective function $s : \Omega \to \mathcal{Z} \subset \mathbb{R}^2$ as $s(\boldsymbol{x}) = (s_1(\boldsymbol{x}), s_2(\boldsymbol{x}))\, \forall \boldsymbol{x} \in \Omega$ where $\mathcal{Z}$ denotes the objective space.

In order to provide a (finite) set of possible alternatives for the MO problem, the concept of Pareto dominance is used. Let $z, z' \in \mathcal{Z}$, we say that $z$ dominates ("is better than") $z'$, denoted by $z \succ z'$, if $z$ is better in at least one objective and worse in no other. We denote by $\mathcal{Z}^*$ the set of non-dominated points in the objective space known as the Pareto-optimal front: $\mathcal{Z}^* = \{z \in \mathcal{Z} \,:\, \nexists\, z' \in \mathcal{Z} \text{ s.t. } z' \succ z\}$.

We denote by $\Omega^*$ the set of non-dominated points in the decision space, which is the pre-image of $\mathcal{Z}^*$, known as the Pareto-optimal set: $\Omega^* = \{\boldsymbol{x} \in \Omega \,:\, s(\boldsymbol{x}) \in \mathcal{Z}^*\}$.

Our interest lies in the bi-objective problem associated with the UCB acquisition function. From now on, we denote the Pareto-optimal set by $\Omega_t^*$ and the vector objective function associated with this problem by $s_t$. That is:

$$\Omega_t^* = \{\boldsymbol{x} \in \Omega \,:\, s_t(\boldsymbol{x}) \in \mathcal{Z}^*\} \text{ with } s_t(\boldsymbol{x}) = (-\mu_t(\boldsymbol{x}), \sigma_t(\boldsymbol{x}))\ . \tag{17}$$

Note that, at each BO step the bi-objective problem will vary, as both objective functions will be updated when new observations are made, and, also note that the UCB single-point optimal solution corresponds with a point of this Pareto set.

In practice, the Pareto-optimal front cannot be calculated exactly because it is very time consuming or even impossible, and, an approximated Pareto front is used instead. Multi-objective evolutionary algorithms (MOEAs) have been developed with the purpose of achieving a uniformly spread approximation of the Pareto-optimal front Greco, Figueira, and Ehrgott (2005). The aim of this kind of algorithm is to evolve the population towards the Pareto-optimal set while maintaining the diversity of the solutions in the population. We will use the MOEA algorithm known as NSGA-II Deb et al. (2002). The NSGA-II, elitist Non-dominated Sorting Genetic Algorithm, considers a population which will always be ranked into Pareto fronts. In order to apply the genetic operators, a partial order to sort the individuals is used by combining two values, the fitness value given by the ranking and the crowding distance value. The crowding distance is an estimate of the density of the solutions surrounding each individual. NSGA-II generates offspring using a specific type of crossover and mutation, and then selects the next generation according to non-dominated sorting and crowding distance comparison.

In this paper, we use the default implementation from Izzo and Biscani (2017) of the MOEA NSGA-II Deb et al. (2002) algorithm to obtain a finite set of non-dominated points among themselves. Note that, any multi-objective algorithm can be applied at this step.

## 4.2. *Selection rule*

The selection rule consists of selecting from the candidate pool the batch to be observed. In order to obtain $q$ different points, we use as the dissimilarity measure the mutual information with respect to the objective function.

As pointed out in Krause, Singh, and Guestrin (2008), maximizing the entropy (which is the same as maximizing the variance in our case) can be seen as an indirect criterion because it only considers the prediction quality of the selected points. Instead of that, by maximizing the mutual information, the quality of the prediction over a region of interest is also taken into account.

In general, the problem of finding the set of points that maximizes the mutual information with respect to a black-box function is known to be NP-complete Krause, Singh, and Guestrin (2008). However, in our setting, as shown also in Krause, Singh, and Guestrin (2008), the mutual information function is submodular and approximately monotonic. Therefore, the batch can be efficiently approximated by applying the Greedy MI rule proposed in Krause, Singh, and Guestrin (2008) and described in Section 4.2.1 and Alg.2. This approximation achieves the following constant factor approximation: $\mathsf{MI}(f;\mathsf{B}^*)-\mathsf{MI}(f;\mathsf{B}) \leq qe-(1-1/e)$ being $\mathsf{B}^*$ the unknown optimal set and $\mathsf{B}$ the set obtained with the algorithm. The algorithm is slightly modified because in our case the UCB one-step-lookahead solution, $\boldsymbol{x}_{t+1}^{UCB}$ (see Eq. 6), is always included as the first point of the batch.

### *4.2.1. Greedy MI algorithm*

The general greedy selection rule consists of, starting from an empty set, adding sequentially the points that maximize the increase of mutual information about the black-box function on the decision space. That is:

$$\boldsymbol{x}_{t+i} = \underset{\boldsymbol{x}\in\Omega}{\arg\max} \Big(\mathsf{MI}\left(f; \{\boldsymbol{x}_{t+j}\}_{j=1}^{i-1} \cup \{\boldsymbol{x}\} \mid \mathcal{D}_{1:t}, \{\boldsymbol{x}_{t+j}\}_{j=1}^{i-1}\right) \\ -\mathsf{MI}\left(f; \{\boldsymbol{x}_{t+j}\}_{j=1}^{i-1} \mid \mathcal{D}_{1:t}, \{\boldsymbol{x}_{t+j}\}_{j=1}^{i-1}\right)\Big) \text{ for } i = 1, \ldots, q\,. \tag{18}$$

As developed in Krause, Singh, and Guestrin (2008), the function used in (18) can be expressed in terms of the entropy function and in our setting, as we assume that $f(\boldsymbol{x})|\mathcal{D}_{1:t} \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t(\boldsymbol{x}))$, maximizing the entropy is equivalent to maximizing the predicted variance.

In Krause, Singh, and Guestrin (2008), they propose an algorithm for maximizing mutual information using the following greedy selection. At each iteration, the algorithm selects the point that maximizes the uncertainty of $f(\boldsymbol{x})$ with respect to the already selected points and minimizes the uncertainty with respect to those not yet selected. They address this bi-objective maximization problem as a single-objective problem using the quotient as follows:

$$\boldsymbol{x}_{t+i} = \underset{\boldsymbol{x}\in\Omega}{\arg\max} \frac{\sigma_t(\boldsymbol{x} \mid \{\boldsymbol{x}_{t+j}\}_{j=1}^{i-1})}{\sigma_t\left(\boldsymbol{x} \mid \Omega\backslash \left(\{\boldsymbol{x}_{t+j}\}_{j=1}^{i-1} \cup \{\boldsymbol{x}\}\right)\right)} \text{ for } i = 1, \ldots, q\,. \tag{19}$$

Note that these calculations are based on the fact that the variance can be updated given an input point and without knowing its observed value. The Greedy MI algorithm used in our approach is described in Alg. 2.

**Algorithm 2** `Greedy MI`

```
Require: Variance function $\sigma$, UCB solution $\boldsymbol{x}^{UCB}$, candidate pool $\mathsf{P}$.
1: $\mathsf{B} \leftarrow \{\boldsymbol{x}^{UCB}\}$
2: $i \leftarrow 1$
3: while $i < q$ do
4:     $\boldsymbol{x}_i \leftarrow \underset{\boldsymbol{x}\in\mathsf{P}}{\arg\max} \frac{\sigma(\boldsymbol{x} \mid \mathsf{B})}{\sigma(\boldsymbol{x} \mid \mathsf{P}\backslash(\mathsf{B} \cup \{\boldsymbol{x}\}))}$
5:     $\mathsf{B} \leftarrow \mathsf{B} \cup \{\boldsymbol{x}_i\}$
6:     $i \leftarrow i + 1$
7: end while
8: return $\mathsf{B}$
```

## 5. Illustrative example

In order to give a visual idea of the procedure we introduce an illustrative example in this section. All the parameters and experimental details are given in Section 6.

In Figure 2, the results obtained by applying our algorithm to minimize the 2-dimensional Branin function are shown. The function has two global optima at $\boldsymbol{x}_1^* = (-\pi, 12.275)$ and $\boldsymbol{x}_2^* = (\pi, 2.275)$, where the function value is zero. In this example the batch size is fixed to five.

The contour plot of the Branin function is used as the background, blue colours represent lower values. The different geometrical figures have the following meaning: black star (observed point), white square (candidate pool point), black square (batch point) and black triangle (UCB one-step look-ahead solution). Note that the UCB one-step look-ahead solution is always included in the batch, but we use a different symbol to differentiate the point.

In the first plot on the left, which corresponds with the first batch step, $q_0 = 10$ initial random points have been observed and the candidate pool corresponds with a LHS of $\mathsf{n} = 100$ points. As the Bayesian process continues, the number of observed points increases and the restriction rule switches to the PS approach. The second plot on the right, corresponds with a step where the Pareto sampling rule is used. In this plot the approximated Pareto set points are shown (decision space) and in the last plot the corresponding points in the bi-objective space are shown. Note that the two objective values are the mean (minimize) and variance (maximize) values.

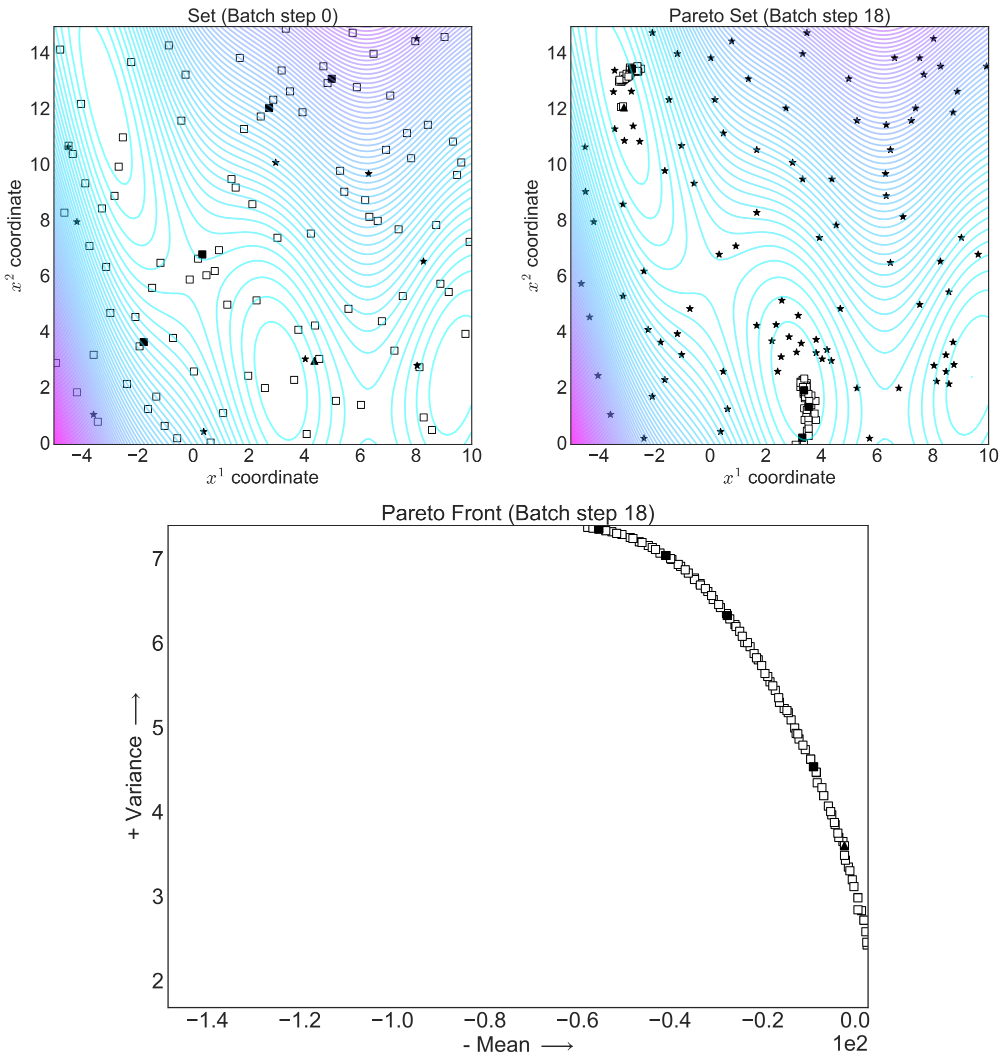

**Figure 2.** Results with the Branin function for batch size $q = 5$.

## 6. Experimental setting

We detail below the setting and parameters used in the experiments. We test the performance of our proposed algorithm in the task of optimizing 12 test functions. We compare the MMIP strategy and 6 variants described below, with the state-of-the-art strategies presented in Section 3 and the UCB single-point strategy as reference.

The 6 variants of the MMIP strategy are listed in detail below and summarized in table 1:

(1) **Rand-MMI-P**. This strategy is the same as MMIP except that, for the first BO steps, no selection rule is applied, the batch is extracted at random from the candidate pool.
(2) **Rand**. The batch is always selected randomly from the corresponding candidate pool.
(3) **MMI-PS**. The restriction rule used is the PS and the selection rule is the Greedy MI rule throughout the whole algorithm.
(4) **Rand-PS**. The batch is selected randomly from the approximated Pareto set.
(5) **MMI-LHS** uses as candidate pool a LHS and the selection rule is the Greedy MI rule.
(6) **Rand-LHS** selects the points randomly from the LHS candidate pool.

Note that, the first two approaches differentiate the restriction and selection rules used based on the number of observations available. The last 4 approaches do not make this differentiation, the same restriction and selection rules are used through the whole BO process.

**Table 1.** MMIP algorithm and variants

| | $1 < t \leq \mathsf{T}_*$ | | $\mathsf{T}_* < t \leq \mathsf{T}$ | |
|---|---|---|---|---|
| | Restriction | Selection | Restriction | Selection |
| MMIP | LHS | MI | PS | MI |
| Rand–MMI–P | LHS | Rand | PS | MI |
| Rand | LHS | Rand | PS | Rand |

| | $1 < t < \mathsf{T}$ | |
|---|---|---|
| | Restriction | Selection |
| MMI–PS | PS | MI |
| Rand–PS | PS | Rand |
| MMI–LHS | LHS | MI |
| Rand–LHS | LHS | Rand |

Our computational development is based on the Python library Bolib (Python library for Bayesian Optimization), Roman et al. (2016b); Roman (2017). The NSGA-II algorithm implementation used is from the Python library PyGMO (Parallel Global Multiobjective Optimizer), Izzo and Biscani (2017). The PE-UCB implementation is based on the available Matlab code from Contal (2016), and a discretization of size $\mathsf{n}$ of the search space is used in order to calculate the relevant region.

Each experiment is repeated 10 independent times with different starting seeds. Therefore, the performance measure used for comparison of a strategy for a given experiment is the average across the 10 different best values found. In our setting the result is the immediate regret at evaluation $t$, that is, $r_t = |f(\tilde{\boldsymbol{x}}_t) - f(\boldsymbol{x}^*)|$ where $\tilde{\boldsymbol{x}}_t$

is the recommendation of the algorithm after $t$ evaluations (the best objective value found so far) and $f(\boldsymbol{x}^*)$ is the minimum value for the objective function (set to zero). A limited number of evaluations, denoted as $\mathsf{T}$, has been used as stopping criteria.

### 6.1. *Parameters*

In order to clarify the parameters involved in the problem, we have grouped them in the following categories: GP, UCB, objective function, infill algorithm optimizer, and batch infill algorithm.

(1) **GP parameters, $\theta_{GP} = (q_0, \mu_0, k, l)$.**
A prior constant mean $\mu_0$ and kernel function $k(\cdot, \cdot)$ have been considered.
Two kernel functions have been tested, the Squared Exponential and the Matern-32 described below.
- $k_{\mathtt{SE}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{r}{2l^2}\right)$,
- $k_{\mathtt{M32}}(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \frac{\sqrt{3}r}{l}\right)\exp\left(-\frac{\sqrt{3}r}{l}\right)$,

where $r = ||\boldsymbol{x} - \boldsymbol{x}'||$ and $l$ denotes the length-scale kernel parameter set to 1 at each dimension, see Rasmussen and Williams (2006).
The value $\mu_0$ has been set as the mean of $q_0 = 10$ uniformly at random selected points $\mathcal{D}_{1:q_0} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{q_0}$, that is, $\mu_0 = \frac{1}{q_0}\sum_{i=1}^{i=q_0} y_i$. Note that, this initial set of observations may or may not be included in the BO process. In our experiments this initial set is included.

(2) **UCB parameter, $\theta_{UCB} = (\kappa_t)$.**
We consider the following adaptative value $\kappa_t = \sqrt{2\log(t^{d/2+2}\pi^2/6)}$ from Brochu, Cora, and de Freitas (2010).

(3) **Objective function parameters: $\theta_f = (f, d, \Omega, k, \mu_\epsilon, \sigma_\epsilon)$.**
Different well-known synthetic test functions with different complexities and dimensions have been used, see Table 2 for a brief description of their parameters. The evaluations are assumed to be noisy with zero-mean Gaussian independent noise, that is, $y = f(\boldsymbol{x}) + \epsilon$ with $\epsilon \overset{i.i.d.}{\sim} \mathcal{N}\left(\mu_\epsilon, \sigma_\epsilon^2\right)$ with $\mu_\epsilon = 0$ and $\sigma_\epsilon^2 = 0.01$.
The closed formula of all the functions and the plots of all the 2-dimensional functions are shown in Annex 9.

(4) **Infill algorithm optimizer: $\theta_{opt} = (\mathsf{n}_{opt})$.**
In our approach, we use a discrete grid optimizer which consists of evaluating the objective function over a random grid and returning the point of the grid where the best objective value is reached. The number of points for the grid is fixed to $\mathsf{n}_{opt} = 20.000$ and the grid is kept constant through the BO process.

(5) **Batch infill algorithm parameters: $\theta_{BI} = (q, \mathsf{T}, \mathsf{T}_*, \mathsf{n}, \theta_{PS})$.**
Three different batch sizes have been tested, $q = 5, 10, 20$, and the number of evaluations is set as $\mathsf{T} = 100\,d$ where $d$ is the dimension of the objective function+. Depending on the batch infill algorithm, additional parameters could be needed. In our approach, the phase break is set as half of the budget $\mathsf{T}_* = \lfloor \mathsf{T}/2 \rfloor$ and the candidate pool size is set to $\mathsf{n} = 100$.
The Pareto Sampling parameters, denoted as $\theta_{PS}$, are those corresponding with the default implementation of the NSGA-II algorithm from the library Izzo and Biscani (2017), that is, $\theta_{PS} = (n_P = \mathsf{n}, n_G = 100, p_c = 0.6, p_m = 0.1, \eta_c = 10, \eta_m = 50)$ where $n_P$ denotes the population size, $n_G$ denotes the number of generations to evolve, $p_c$ and $p_m$ denote the crossover and mutation probabilities, and $\eta_c$ and $\eta_m$ denote the crossover and mutation distribution indexes used to

**Table 2.** Objective functions ($f$ notation, $d$ dimension and $\Omega$ search space).

| Name | $f$ | $d$ | $\Omega$ | $k$ |
|---|---|---|---|---|
| Branin | $f_B$ | 2 | $[-5,10] \times [0,15]$ | $k_{\mathtt{M32}}$ |
| cosines | $f_c$ | 2 | $[0,5]^2$ | $k_{\mathtt{M32}}$ |
| Rastrigin | $f_R$ | 2 | $[-5.12,5.12]^2$ | $k_{\mathtt{SE}}$ |
| gSobol–2 | $f_{gS2}$ | 2 | $[-4,6]^2$ | $k_{\mathtt{SE}}$ |
| gSobol–5 | $f_{gS5}$ | 5 | $[-5,5]^5$ | $k_{\mathtt{M32}}$ |
| gSobol–10 | $f_{gS10}$ | 10 | $[-5,5]^{10}$ | $k_{\mathtt{M32}}$ |
| Levy–2 | $f_{L2}$ | 2 | $[-10,10]^2$ | $k_{\mathtt{SE}}$ |
| Levy–5 | $f_{L5}$ | 5 | $[-10,10]^5$ | $k_{\mathtt{M32}}$ |
| Levy–10 | $f_{L10}$ | 10 | $[-10,10]^{10}$ | $k_{\mathtt{SE}}$ |
| Michalewicz–2 | $f_{M2}$ | 2 | $[0,\pi]^2$ | $k_{\mathtt{M32}}$ |
| Michalewicz–5 | $f_{M5}$ | 5 | $[0,\pi]^5$ | $k_{\mathtt{SE}}$ |
| Michalewicz–10 | $f_{M10}$ | 10 | $[0,\pi]^{10}$ | $k_{\mathtt{SE}}$ |

adapt both binary operators to the real case.

### 6.2. *Parameters selection*

- Objective function kernel selection:
  In order to reduce the number of experiments, we have associated each objective function with the kernel function for which the UCB single-point acquisition function attains better results. The final kernel selection for each objective function is specified in Table 2. The complete list of results can be seen in Annex 10.
- NSGA-II parameters influence:
  A sensitivity study about the population size and number of generation parameters have been performed. No specific pattern is appreciated as the results vary with the objective function. The results are shown in Annex 12.

## 7. Experimental results

In this section, the main results obtained are presented. In the interest of space, the complete list of results (in graphical and tabular formats) can be consulted in Annex 10.

The tabular results presented in this section are separated into three double tables according to the three batch sizes considered, see Tables 3, 6 and 9. In each table, each row corresponds with an infill strategy. Each objective function has three columns: mean, standard deviation and p-value after performing the Wilcoxon signed-rank test against the best strategy. In the p-value column, the batch strategies obtaining the best value is marked with an asterisk. In order to check whether there are significant differences, we consider a significant level of $\alpha = 0.05$ and the values below this level, those with significant differences, are marked in bold. Note that the UCB single-point solution is not included in the test.

In order to see the evolution of the different infill algorithms, some figures are also presented. In the figures, the x-axis corresponds with the number of observations and the y-axis with the objective value obtained (in the interval $[0,1]$). Each curve corresponds to an infill algorithm. Black points on the line represent when the batches are observed, and the line is the interpolated curve. For each objective function, three plots (upper, center and lower) separating the different algorithms are used for reasons of clarity. In the upper plot the state-of-the-art algorithms (lines in blue tones), the MMIP (purple line) and the UCB single-point (black line) are shown. In the center, our variants of the MMIP are plotted (lines in red tones), and, finally, in the lower plot all the curves are plotted together. As we are minimizing, lower values are preferred.

We include in this section Figure 3, where each column corresponds with a gSobol

**Table 3.** Results for batch size $q = 5$

| Name | Branin | | | cosines | | | Rastrigin | | | gSobol-2 | | | Levy-2 | | | Michalewicz-2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ |
| B-UCB | 0.01 | 0.01 | 0.114 | 0.00 | 0.01 | 0.059 | 4.25 | 3.15 | 0.678 | 0.26 | 0.00 | 0.878 | 0.03 | 0.04 | **0.013** | 0.04 | 0.18 | 0.575 |
| PE-UCB | 0.07 | 0.08 | **0.007** | 0.02 | 0.01 | **0.005** | 5.23 | 3.55 | 0.114 | 0.32 | 0.02 | **0.005** | 0.00 | 0.00 | * | −0.02 | 0.00 | 0.139 |
| LP-UCB | 0.39 | 0.07 | **0.005** | 1.38 | 0.72 | **0.005** | 6.33 | 0.01 | **0.017** | 1.42 | 0.39 | **0.005** | 1.29 | 0.74 | **0.005** | 0.18 | 0.01 | **0.005** |
| PRED-UCB | 0.01 | 0.01 | **0.037** | −0.01 | 0.01 | * | 3.23 | 2.77 | * | 0.26 | 0.01 | * | 0.07 | 0.08 | **0.009** | 0.03 | 0.17 | 0.241 |
| RAND-UCB | 0.08 | 0.06 | **0.005** | 0.07 | 0.05 | **0.005** | 4.10 | 1.45 | 0.386 | 0.29 | 0.03 | **0.009** | 0.08 | 0.06 | **0.007** | −0.01 | 0.00 | **0.007** |
| Lambda-UCB | 0.03 | 0.08 | 0.878 | 0.04 | 0.12 | **0.037** | 3.36 | 2.36 | 0.799 | 0.73 | 0.73 | 0.139 | 0.46 | 0.79 | 0.959 | 0.32 | 0.43 | **0.022** |
| MMIP | 0.07 | 0.06 | **0.005** | 0.05 | 0.04 | **0.005** | 3.68 | 1.79 | 0.646 | 0.29 | 0.02 | **0.005** | 0.06 | 0.05 | **0.007** | −0.02 | 0.01 | * |
| Rand-MMI-P | 0.05 | 0.05 | **0.005** | 0.08 | 0.08 | **0.005** | 5.38 | 1.77 | 0.169 | 0.28 | 0.02 | **0.005** | 0.08 | 0.05 | **0.007** | −0.02 | 0.01 | 0.646 |
| Rand | 0.03 | 0.04 | **0.009** | 0.01 | 0.03 | **0.013** | 5.37 | 1.89 | 0.059 | 0.29 | 0.02 | **0.005** | 0.03 | 0.06 | 0.333 | −0.02 | 0.01 | 0.646 |
| MMI-PS | 0.04 | 0.04 | **0.017** | 0.03 | 0.01 | **0.005** | 4.50 | 2.37 | 0.445 | 0.29 | 0.02 | **0.007** | 0.06 | 0.06 | **0.007** | −0.02 | 0.00 | 0.093 |
| Rand-PS | 0.00 | 0.02 | * | 0.00 | 0.01 | 0.074 | 3.30 | 1.42 | 0.799 | 0.28 | 0.02 | **0.028** | 0.00 | 0.01 | 0.959 | −0.02 | 0.00 | 0.646 |
| MMI-LHS | 0.05 | 0.05 | **0.013** | 0.13 | 0.11 | **0.005** | 3.84 | 1.52 | 0.386 | 0.29 | 0.02 | **0.005** | 0.10 | 0.06 | **0.007** | −0.01 | 0.01 | **0.028** |
| Rand-LHS | 0.06 | 0.06 | **0.005** | 0.07 | 0.09 | **0.005** | 4.89 | 2.05 | 0.241 | 0.30 | 0.02 | **0.007** | 0.11 | 0.08 | **0.005** | −0.01 | 0.01 | **0.013** |
| UCB | 0.02 | 0.01 | – | −0.01 | 0.01 | – | 3.29 | 2.99 | – | 0.27 | 0.01 | – | 0.02 | 0.03 | – | −0.02 | 0.00 | – |

| Name | gSobol-5 | | | Levy-5 | | | Michalewicz-5 | | | gSobol-10 | | | Levy-10 | | | Michalewicz-10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ |
| B-UCB | 213.12 | 192.66 | **0.005** | 7.73 | 7.06 | **0.007** | 1.95 | 0.30 | 0.646 | 8.86e+4 | 8.83e+4 | **0.005** | 36.94 | 13.56 | **0.005** | 6.18 | 0.20 | **0.007** |
| PE-UCB | 7.59 | 4.91 | **0.007** | 1.70 | 0.99 | **0.047** | 2.37 | 0.30 | **0.022** | 1.37e+5 | 1.39e+5 | **0.005** | 38.41 | 11.64 | **0.005** | 6.42 | 0.31 | **0.005** |
| LP-UCB | 4.92 | 0.01 | **0.005** | 3.15 | 0.01 | **0.005** | 2.11 | 0.01 | **0.047** | 6.82e+2 | 6.19e−3 | **0.005** | 18.47 | 0.01 | **0.013** | 5.78 | 0.11 | **0.013** |
| PRED-UCB | 135.75 | 203.27 | **0.005** | 7.57 | 6.74 | **0.007** | 1.89 | 0.32 | 0.508 | 5.55e+4 | 3.91e+4 | **0.005** | 30.17 | 10.08 | **0.005** | 6.04 | 0.40 | **0.013** |
| RAND-UCB | 7.90 | 6.45 | **0.007** | 2.89 | 1.54 | **0.007** | 2.01 | 0.31 | 0.285 | 1.34e+3 | 8.22e+2 | **0.005** | 18.99 | 3.22 | **0.009** | 5.53 | 0.36 | 0.333 |
| Lambda-UCB | 213.49 | 191.70 | **0.005** | 14.07 | 5.02 | **0.005** | 1.78 | 0.44 | * | 8.63e+4 | 8.34e+4 | **0.005** | 38.91 | 14.28 | **0.005** | 5.62 | 0.56 | 0.333 |
| MMIP | 0.83 | 0.36 | * | 0.80 | 0.53 | * | 2.11 | 0.41 | **0.017** | 1.90e+1 | 2.02e+1 | * | 18.21 | 3.98 | **0.022** | 5.31 | 0.52 | 0.508 |
| Rand-MMI-P | 0.96 | 0.42 | 0.203 | 1.00 | 0.45 | 0.445 | 2.08 | 0.24 | 0.203 | 1.32e+3 | 1.25e+3 | **0.005** | 16.72 | 2.51 | 0.114 | 5.36 | 0.38 | 0.594 |
| Rand | 1.02 | 0.55 | 0.386 | 1.21 | 0.98 | 0.333 | 2.00 | 0.22 | 0.169 | 2.93e+2 | 2.22e+2 | **0.005** | 16.01 | 2.17 | 0.285 | 5.29 | 0.51 | * |
| MMI-PS | 1.48 | 0.80 | **0.037** | 2.38 | 2.62 | 0.114 | 2.07 | 0.25 | 0.074 | 2.40e+3 | 4.13e+3 | **0.005** | 14.83 | 3.03 | 0.878 | 5.74 | 0.47 | 0.203 |
| Rand-PS | 1.55 | 1.17 | 0.139 | 1.53 | 1.87 | 0.203 | 1.93 | 0.32 | 0.333 | 7.90e+2 | 1.32e+3 | **0.005** | 14.01 | 3.60 | * | 5.42 | 0.52 | 0.646 |
| MMI-LHS | 5.81 | 3.26 | **0.005** | 2.32 | 0.93 | **0.007** | 1.94 | 0.34 | 0.241 | 5.70e+2 | 5.63e+2 | **0.005** | 16.73 | 4.94 | 0.241 | 5.42 | 0.28 | 0.799 |
| Rand-LHS | 13.77 | 6.38 | **0.005** | 3.40 | 0.86 | **0.005** | 1.98 | 0.37 | 0.285 | 1.49e+3 | 1.21e+3 | **0.005** | 17.55 | 3.64 | 0.093 | 5.61 | 0.34 | 0.203 |
| UCB | 121.22 | 198.39 | – | 7.07 | 6.25 | – | 1.82 | 0.36 | – | 5.00e+4 | 3.48e+4 | – | 26.66 | 10.25 | – | 5.91 | 0.40 | – |

**Table 6.** Results for batch size $q = 10$

| Name | Branin | | | cosines | | | Rastrigin | | | gSobol-2 | | | Levy-2 | | | Michalewicz-2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ |
| B-UCB | 0.01 | 0.02 | 0.508 | −0.01 | 0.01 | 0.333 | 4.62 | 3.19 | 0.074 | 0.26 | 0.01 | 0.575 | 0.02 | 0.02 | **0.022** | 0.04 | 0.17 | 0.139 |
| PE-UCB | 0.14 | 0.13 | **0.005** | 0.26 | 0.16 | **0.005** | 6.28 | 2.19 | 0.059 | 0.36 | 0.00 | **0.005** | 0.00 | 0.01 | * | −0.01 | 0.01 | 0.241 |
| LP-UCB | 0.39 | 0.07 | **0.005** | 1.39 | 0.72 | **0.005** | 6.33 | 0.01 | **0.028** | 1.43 | 0.39 | **0.005** | 1.29 | 0.74 | **0.005** | 0.17 | 0.01 | **0.005** |
| PRED-UCB | 0.01 | 0.01 | 0.333 | −0.01 | 0.01 | * | 5.12 | 2.35 | 0.059 | 0.26 | 0.01 | * | 0.04 | 0.04 | **0.017** | 0.08 | 0.30 | 0.139 |
| RAND-UCB | 0.09 | 0.09 | **0.005** | 0.28 | 0.11 | **0.005** | 5.12 | 2.98 | 0.114 | 0.36 | 0.04 | **0.005** | 0.13 | 0.09 | **0.005** | 0.01 | 0.02 | 0.114 |
| Lambda-UCB | 0.06 | 0.10 | 0.169 | 0.07 | 0.16 | 0.074 | 3.21 | 2.76 | * | 0.63 | 0.77 | **0.037** | 0.46 | 0.76 | 0.241 | 0.33 | 0.42 | 0.114 |
| MMIP | 0.01 | 0.01 | 0.059 | 0.29 | 0.19 | **0.005** | 5.25 | 1.84 | 0.114 | 0.34 | 0.05 | **0.005** | 0.05 | 0.06 | **0.013** | −0.01 | 0.00 | 0.646 |
| Rand-MMI-P | 0.03 | 0.03 | 0.074 | 0.28 | 0.16 | **0.005** | 4.64 | 1.95 | **0.047** | 0.32 | 0.04 | **0.005** | 0.04 | 0.07 | **0.022** | −0.01 | 0.01 | 0.386 |
| Rand | 0.01 | 0.02 | 0.721 | 0.03 | 0.06 | **0.005** | 4.81 | 1.93 | 0.059 | 0.31 | 0.03 | **0.005** | 0.01 | 0.02 | 0.139 | −0.01 | 0.00 | 0.575 |
| MMI-PS | 0.01 | 0.01 | 0.285 | 0.25 | 0.17 | **0.005** | 4.47 | 1.27 | 0.139 | 0.31 | 0.03 | **0.005** | 0.03 | 0.03 | **0.007** | −0.01 | 0.01 | 0.374 |
| Rand-PS | 0.00 | 0.01 | * | 0.00 | 0.01 | **0.009** | 4.12 | 2.05 | 0.386 | 0.30 | 0.05 | **0.007** | 0.10 | 0.29 | 0.646 | −0.01 | 0.01 | * |
| MMI-LHS | 0.05 | 0.05 | **0.009** | 0.43 | 0.14 | **0.005** | 4.32 | 1.69 | 0.241 | 0.32 | 0.03 | **0.005** | 0.10 | 0.06 | **0.007** | 0.00 | 0.01 | **0.007** |
| Rand-LHS | 0.04 | 0.04 | **0.047** | 0.27 | 0.16 | **0.005** | 4.60 | 1.76 | 0.074 | 0.33 | 0.06 | **0.005** | 0.16 | 0.11 | **0.005** | 0.00 | 0.02 | **0.009** |
| UCB | 0.02 | 0.01 | – | −0.01 | 0.01 | – | 3.29 | 2.99 | – | 0.27 | 0.01 | – | 0.02 | 0.03 | – | −0.02 | 0.00 | – |

| Name | gSobol-5 | | | Levy-5 | | | Michalewicz-5 | | | gSobol-10 | | | Levy-10 | | | Michalewicz-10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ |
| B-UCB | 249.91 | 199.31 | **0.005** | 8.25 | 7.32 | **0.009** | 1.95 | 0.29 | 0.203 | 1.30$e$+5 | 1.43$e$+5 | **0.005** | 36.98 | 12.30 | **0.005** | 6.13 | 0.31 | **0.013** |
| PE-UCB | 10.76 | 7.68 | **0.005** | 1.57 | 1.12 | 0.799 | 2.38 | 0.49 | **0.022** | 1.84$e$+5 | 2.05$e$+5 | **0.005** | 40.57 | 12.47 | **0.005** | 6.46 | 0.27 | **0.005** |
| LP-UCB | 4.91 | 0.01 | **0.005** | 3.14 | 0.01 | **0.007** | 2.11 | 0.01 | **0.037** | 6.82$e$+2 | 1.00$e$−2 | **0.005** | 18.47 | 0.01 | **0.007** | 5.77 | 0.15 | **0.028** |
| PRED-UCB | 141.77 | 207.20 | **0.005** | 7.43 | 6.99 | **0.009** | 1.94 | 0.27 | 0.646 | 5.97$e$+4 | 4.11$e$+4 | **0.005** | 30.45 | 10.25 | **0.005** | 6.07 | 0.31 | **0.013** |
| RAND-UCB | 11.85 | 6.17 | **0.005** | 2.82 | 1.23 | **0.005** | 2.06 | 0.32 | 0.203 | 1.25$e$+3 | 1.33$e$+3 | **0.022** | 16.30 | 5.43 | 0.074 | 5.41 | 0.76 | 0.169 |
| Lambda-UCB | 250.84 | 199.97 | **0.005** | 14.14 | 5.73 | **0.005** | 2.09 | 0.58 | 0.241 | 1.32$e$+5 | 1.46$e$+5 | **0.005** | 43.13 | 16.21 | **0.005** | 4.91 | 0.98 | * |
| MMIP | 0.78 | 0.28 | * | 1.54 | 0.92 | 0.799 | 2.18 | 0.34 | 0.139 | 6.27$e$+1 | 1.42$e$+2 | * | 15.70 | 4.11 | 0.241 | 5.45 | 0.35 | 0.333 |
| Rand-MMI-P | 0.82 | 0.31 | 0.575 | 1.75 | 0.71 | 0.646 | 2.01 | 0.22 | 0.285 | 9.35$e$+2 | 2.06$e$+3 | **0.005** | 15.41 | 2.82 | 0.374 | 5.63 | 0.19 | 0.093 |
| Rand | 0.97 | 0.50 | 0.333 | 1.58 | 1.13 | 0.799 | 2.02 | 0.17 | 0.333 | 5.58$e$+2 | 1.17$e$+3 | **0.005** | 13.53 | 3.33 | * | 5.48 | 0.52 | 0.203 |
| MMI-PS | 1.89 | 1.25 | **0.022** | 1.00 | 0.73 | 0.074 | 1.97 | 0.33 | 0.575 | 2.17$e$+3 | 2.92$e$+3 | **0.005** | 16.20 | 3.55 | 0.285 | 5.37 | 0.52 | 0.445 |
| Rand-PS | 1.49 | 1.40 | 0.386 | 1.46 | 0.97 | * | 1.83 | 0.34 | * | 1.40$e$+3 | 1.61$e$+3 | **0.007** | 14.67 | 4.35 | 0.386 | 5.58 | 0.42 | 0.241 |
| MMI-LHS | 4.19 | 2.25 | **0.005** | 2.22 | 0.76 | 0.139 | 2.07 | 0.27 | 0.241 | 4.11$e$+2 | 2.08$e$+2 | **0.028** | 17.57 | 4.00 | 0.074 | 5.57 | 0.34 | 0.169 |
| Rand-LHS | 9.20 | 2.65 | **0.005** | 3.00 | 0.85 | **0.013** | 1.88 | 0.44 | 0.799 | 2.02$e$+3 | 1.92$e$+3 | **0.005** | 17.23 | 3.24 | 0.203 | 5.78 | 0.33 | **0.007** |
| UCB | 121.22 | 198.39 | – | 7.07 | 6.25 | – | 1.82 | 0.36 | – | 5.00$e$+4 | 3.48$e$+4 | – | 26.66 | 10.25 | – | 5.91 | 0.40 | – |

**Table 9.** Results for batch size $q = 20$

| Name | Branin | | | cosines | | | Rastrigin | | | gSobol-2 | | | Levy-2 | | | Michalewicz-2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ |
| B-UCB | 0.01 | 0.01 | **0.037** | −0.01 | 0.01 | * | 4.22 | 2.86 | 0.721 | 0.26 | 0.01 | * | 0.04 | 0.04 | **0.017** | 0.03 | 0.18 | 0.173 |
| PE-UCB | 0.09 | 0.05 | **0.009** | 0.42 | 0.15 | **0.005** | 5.42 | 1.86 | 0.059 | 0.36 | 0.00 | **0.005** | 0.00 | 0.01 | 0.721 | 0.01 | 0.01 | **0.005** |
| LP-UCB | 0.39 | 0.07 | **0.005** | 1.36 | 0.73 | **0.005** | 6.34 | 0.01 | **0.005** | 1.34 | 0.30 | **0.005** | 1.10 | 0.64 | **0.005** | 0.17 | 0.01 | **0.005** |
| PRED-UCB | 0.02 | 0.02 | **0.022** | 0.00 | 0.01 | 0.203 | 6.77 | 3.44 | 0.093 | 0.26 | 0.01 | 0.074 | 0.02 | 0.01 | **0.037** | −0.02 | 0.01 | * |
| RAND-UCB | 0.18 | 0.17 | **0.005** | 0.31 | 0.18 | **0.005** | 5.18 | 2.50 | 0.203 | 0.51 | 0.31 | **0.005** | 0.14 | 0.10 | **0.007** | 0.10 | 0.23 | **0.005** |
| Lambda-UCB | 0.29 | 0.51 | 0.169 | 0.12 | 0.18 | **0.005** | 4.31 | 4.34 | 0.799 | 0.33 | 0.16 | **0.028** | 0.51 | 0.75 | 0.114 | 0.45 | 0.40 | **0.007** |
| MMIP | 0.02 | 0.02 | 0.059 | 0.06 | 0.05 | **0.005** | 4.62 | 2.64 | 0.116 | 0.37 | 0.05 | **0.005** | 0.03 | 0.04 | **0.037** | 0.00 | 0.01 | **0.005** |
| Rand-MMI-P | 0.02 | 0.01 | **0.022** | 0.13 | 0.13 | **0.009** | 5.28 | 2.19 | 0.285 | 0.40 | 0.10 | **0.005** | 0.01 | 0.01 | **0.037** | 0.01 | 0.01 | **0.005** |
| Rand | 0.02 | 0.03 | **0.022** | 0.13 | 0.13 | **0.007** | 5.22 | 2.25 | 0.285 | 0.38 | 0.10 | **0.005** | 0.05 | 0.13 | 0.139 | 0.00 | 0.01 | **0.005** |
| MMI-PS | 0.00 | 0.01 | * | 0.11 | 0.13 | **0.005** | 5.16 | 3.13 | 0.386 | 0.37 | 0.05 | **0.005** | 0.01 | 0.03 | 0.241 | 0.00 | 0.01 | **0.017** |
| Rand-PS | 0.00 | 0.01 | 0.646 | 0.09 | 0.13 | **0.013** | 5.07 | 2.67 | 0.093 | 0.32 | 0.04 | **0.007** | 0.00 | 0.02 | * | −0.01 | 0.01 | **0.028** |
| MMI-LHS | 0.12 | 0.12 | **0.007** | 0.38 | 0.21 | **0.005** | 3.67 | 1.80 | * | 0.40 | 0.07 | **0.005** | 0.14 | 0.11 | **0.005** | 0.03 | 0.04 | **0.005** |
| Rand-LHS | 0.12 | 0.06 | **0.005** | 0.31 | 0.17 | **0.005** | 5.33 | 1.62 | 0.114 | 0.40 | 0.14 | **0.005** | 0.19 | 0.16 | **0.005** | 0.01 | 0.01 | **0.005** |
| UCB | 0.02 | 0.01 | – | −0.01 | 0.01 | – | 3.29 | 2.99 | – | 0.27 | 0.01 | – | 0.02 | 0.03 | – | −0.02 | 0.00 | – |

| Name | gSobol-5 | | | Levy-5 | | | Michalewicz-5 | | | gSobol-10 | | | Levy-10 | | | Michalewicz-10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ | $m$ | $\pm sd$ | $p$ |
| B-UCB | 273.40 | 203.59 | **0.005** | 7.92 | 7.59 | **0.022** | 2.06 | 0.23 | 0.508 | 1.78$e$+5 | 2.14$e$+5 | **0.005** | 35.48 | 11.90 | **0.005** | 5.87 | 0.33 | 0.074 |
| PE-UCB | 9.59 | 5.98 | **0.005** | 2.47 | 1.94 | 0.203 | 2.63 | 0.36 | **0.005** | 2.18$e$+5 | 2.69$e$+5 | **0.005** | 41.67 | 12.93 | **0.005** | 6.49 | 0.48 | **0.005** |
| LP-UCB | 4.92 | 0.01 | **0.005** | 3.14 | 0.01 | **0.007** | 2.11 | 0.01 | 0.139 | 6.82$e$+2 | 9.16$e$−3 | **0.013** | 18.47 | 0.01 | **0.007** | 5.78 | 0.14 | 0.074 |
| PRED-UCB | 153.46 | 209.46 | **0.005** | 7.51 | 6.92 | **0.022** | 1.92 | 0.24 | 0.878 | 7.42$e$+4 | 5.49$e$+4 | **0.005** | 30.50 | 10.17 | **0.005** | 5.88 | 0.38 | **0.022** |
| RAND-UCB | 9.01 | 4.93 | **0.005** | 3.30 | 2.36 | 0.093 | 2.12 | 0.12 | 0.139 | 9.96$e$+2 | 8.03$e$+2 | 0.059 | 18.38 | 4.27 | **0.022** | 5.41 | 0.53 | 0.721 |
| Lambda-UCB | 274.38 | 205.58 | **0.005** | 15.04 | 5.26 | **0.005** | 1.96 | 0.39 | 0.959 | 1.78$e$+5 | 2.13$e$+5 | **0.005** | 45.84 | 17.34 | **0.005** | 5.17 | 0.72 | * |
| MMIP | 0.76 | 0.29 | 0.386 | 1.80 | 1.03 | 0.959 | 2.06 | 0.21 | 0.445 | 2.63$e$+2 | 3.09$e$+2 | * | 12.74 | 3.42 | * | 5.49 | 0.36 | 0.285 |
| Rand-MMI-P | 0.96 | 0.64 | 0.139 | 1.61 | 0.94 | * | 2.07 | 0.46 | 0.273 | 1.36$e$+3 | 2.46$e$+3 | 0.241 | 16.32 | 4.74 | 0.169 | 5.56 | 0.40 | 0.093 |
| Rand | 0.66 | 0.28 | * | 1.81 | 1.17 | 0.445 | 1.97 | 0.40 | 0.5 | 1.50$e$+3 | 2.39$e$+3 | **0.009** | 18.80 | 2.77 | **0.017** | 5.39 | 0.44 | 0.333 |
| MMI-PS | 1.67 | 1.03 | **0.022** | 1.07 | 0.65 | **0.028** | 2.24 | 0.44 | 0.114 | 3.45$e$+3 | 4.23$e$+3 | **0.013** | 13.25 | 2.80 | 0.799 | 5.85 | 0.36 | 0.059 |
| Rand-PS | 1.55 | 0.97 | **0.022** | 3.37 | 2.34 | 0.093 | 2.14 | 0.45 | 0.386 | 1.77$e$+3 | 2.94$e$+3 | **0.037** | 14.98 | 3.07 | 0.139 | 5.33 | 0.34 | 0.445 |
| MMI-LHS | 7.47 | 3.33 | **0.005** | 3.09 | 0.97 | **0.013** | 1.95 | 0.22 | 0.721 | 6.63$e$+2 | 3.16$e$+2 | **0.009** | 17.02 | 2.08 | **0.005** | 5.55 | 0.27 | 0.169 |
| Rand-LHS | 8.15 | 2.93 | **0.005** | 3.29 | 1.18 | **0.007** | 1.92 | 0.41 | * | 1.51$e$+3 | 1.33$e$+3 | **0.009** | 19.19 | 4.51 | **0.028** | 5.49 | 0.32 | 0.114 |
| UCB | 121.22 | 198.39 | – | 7.07 | 6.25 | – | 1.82 | 0.36 | – | 5.00$e$+4 | 3.48$e$+4 | – | 26.66 | 10.25 | – | 5.91 | 0.40 | – |

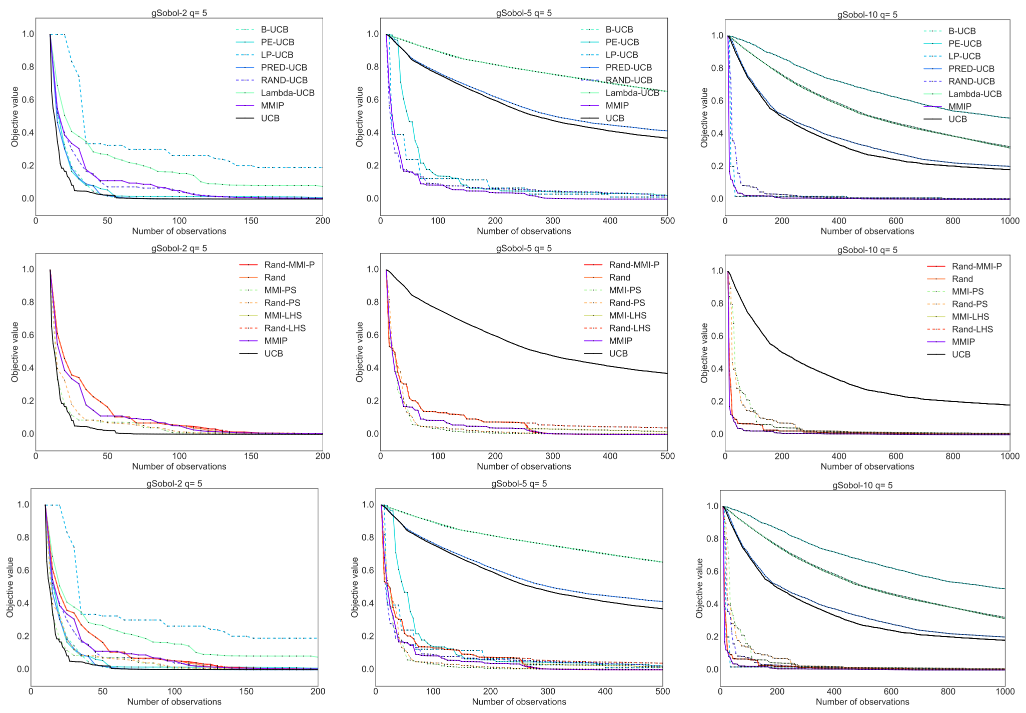

**Figure 3.** Results with $q = 5$ for gSobol-2,5,10.

with our proposed variants and MMIP has no significant differences for batch sizes $q = 10, 20$. For the Rastrigin function, MMIP has no significant differences for all the batch sizes. And, finally, for the Michalewicz–2 function, MMIP is the best or has no significant differences for batch sizes 5 and 10 respectively, and for batch size 10 the best strategy is Pred-UCB.

For higher dimensions, the results obtained with some batch strategies improve the results obtained with the UCB reference function. For gSobol–5 and Levy–5, the MMIP obtains the best result or has no significant differences for all the batch sizes considered. For Michalewicz–5 and Levy–10, only for $q = 5$ does the MMIP strategy have significant differences. For gSobol-10, MMIP attains the best result for all the batch sizes and no other batch strategy attains similar improvements. Finally, for Michalewicz–10, MMIP results have no significant differences for any batch size.

## 7.1. *UCB single-point function influence*

The UCB single-point strategy is used as reference, as the model is updated each time an observation is made, and better results are expected. In some cases, the value obtained with the UCB strategy is in the same range as the value obtained with some of the batch strategies. However, for some cases (gSobol-5, Levy-5, gSobol-10 and Levy-10) the UCB algorithm is outperformed as also pointed out in González et al. (2016). For gSobol-5 and Levy-5, the strategies PE-UCB, LP-UCB, Rand-UCB and our proposals, improve the result obtained with the UCB. For gSobol-10 and Levy-10, the strategies Rand-UCB, LP-UCB and our proposals outperform the results, but the highest improvement is obtained considering our MMIP strategy. This behaviour is maintained for the three batch sizes considered.

In the cases where the single-point UCB strategy seems to get stuck, the strategies capable of escaping from the UCB behaviour are those that apply a more exploratory approach, focused on the uncertainty reduction, and strategies that consider different points from the bi-objective problem, such as our Pareto approach. Note that the UCB solution corresponds with a specific solution of the bi-objective solution. For the strategies that we propose, all of them escape from the UCB behaviour and some of them improve the results significantly when the dimensionality, and therefore the complexity, of the problem increases. The batch algorithms that blindly rely on the UCB behaviour obtain similar or even worse results (B-UCB, Pred-UCB and Lambda-UCB).

### 7.2. *Batch size influence*

In order to appreciate the evolution of the strategies as the batch size increases, the error plots are shown in Figures 4-5. For each batch size ($q = 5, 10, 20$) the corresponding mean and standard deviation values are plotted for the batch infill algorithms. The UCB single-point solution is also included in black for purposes of comparison.

In general, as expected, as the batch size increases, similar or worse results are obtained. However, for the LP-UCB strategy, the results obtained are almost constant and, for our MMIP strategy, in some scenarios better results are obtained as the size increases from 10 to 20 (Rastrigin, cosines, Michalewicz-5, and Levy-10). For Michalewicz-5, the improvement obtained from size 10 to 20 is the same as that obtained with $\lambda$-UCB. Both strategies consider points from the Pareto set.

### 7.3. *Problem dimension influence*

The two dimensional problems, except for Rastrigin function, seem to be very easy to solve. All the batch algorithms, together with the UCB reference strategy, attain near optimal results. We can appreciate from Figure 3 that, as the dimension of the problem increases, the UCB single-point function behaves worse and, therefore, those algorithms that rely strongly only on the information given by the UCB do not behave very well.

The MMIP strategy obtains the best solution for the objective functions gSobol-5 (batch size $q = 5, 10$), Levy-5 (batch size $q = 5$), gSobol-10 (batch size $q = 5, 10, 20$) and Levy-10 (batch size $q = 20$). In the gSobol-10 case, our results are significantly better when compared with the other batch strategies. In the remaining cases, Michalewicz-5 and Levy-10, for batch size 5 our MMIP strategy has significant differences but as the batch size increases the differences become not significant.

### 7.4. *Gaussian process initialization*

In Morar, Knowles, and Sampaio (2017), the authors address the problem of designing the initial sampling plan for the Bayesian optimization method, that is to select the initial samples to initialize the surrogate model, in our case the Gaussian process. They highlight the importance of having sensible default parameters that could adapt to the problem. In our approach, we have initialized the GP constant mean value using $q_0 = 10$ initial samples, that is, $\mu_0 = \frac{1}{q_0} \sum_{i=1}^{i=q_0} y_i$ with $\mathcal{D}_{1:q_0} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{q_0}$ randomly selected observations. In our experiments these initial observations are also included in the BO procedure. We have performed additional experiments using $q_0 = d$ initial samples, being $d$ the dimension of the problem. Note that for those objective functions

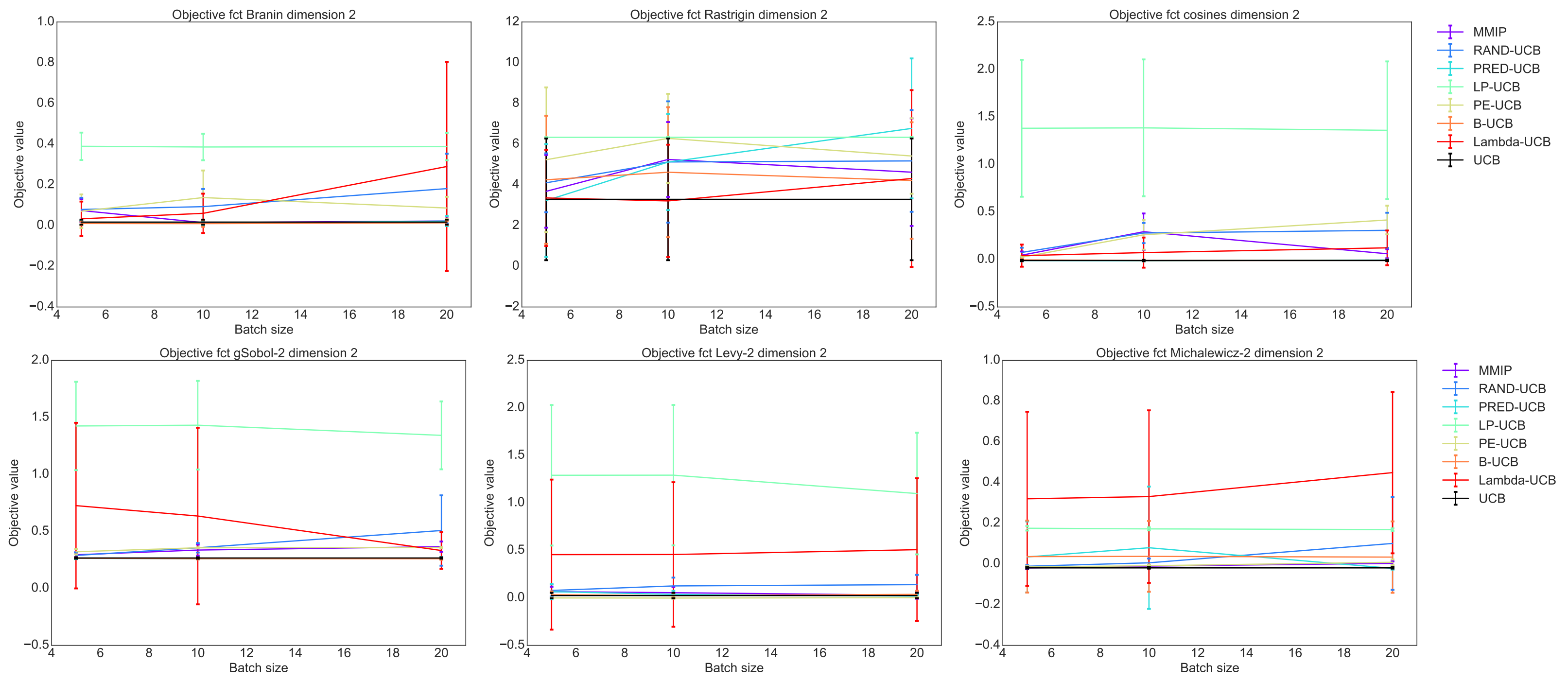

**Figure 4.** Results obtained according to the batch size for dimension 2

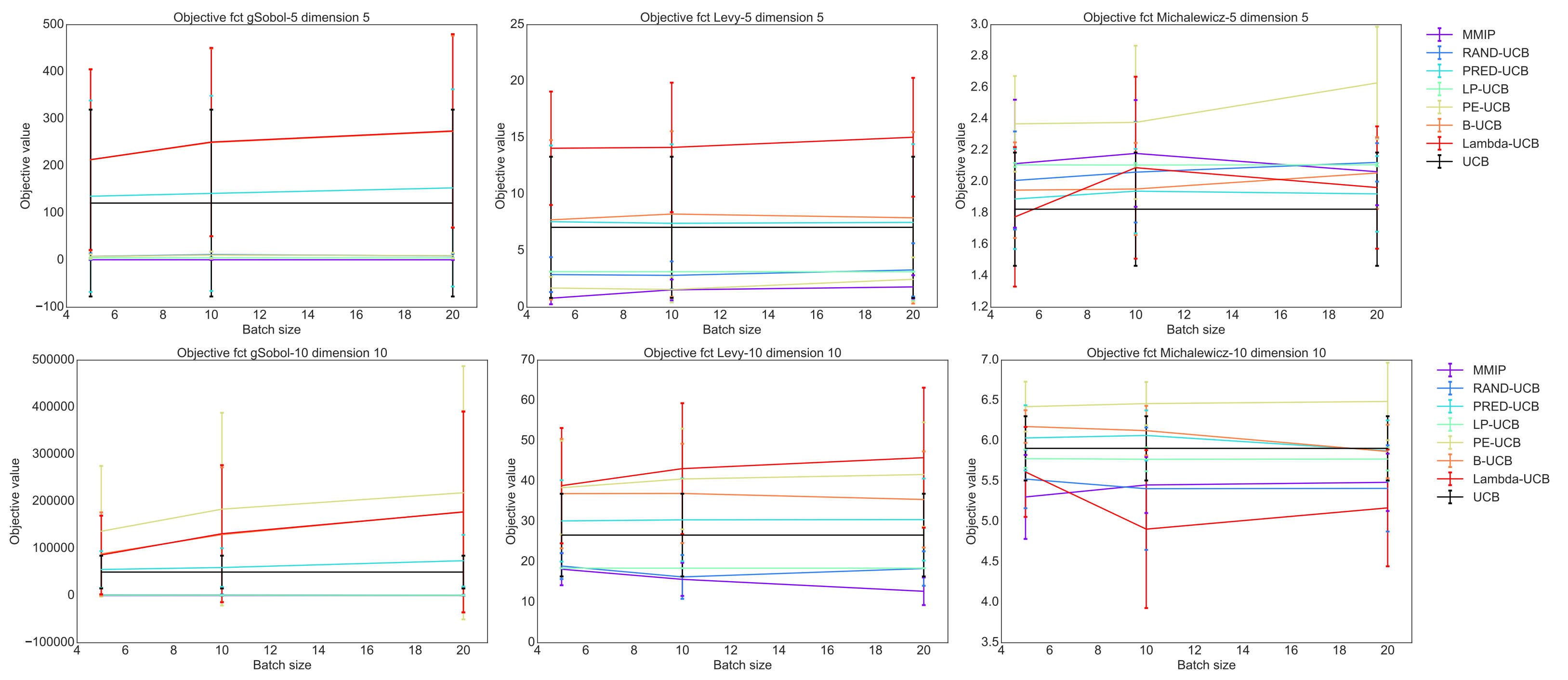

**Figure 5.** Results obtained depending on the batch size for dimension 5 and 10

with dimension $d = 10$, both experiments are the same.

In the figures, the plots in the left column are the experiments performed with $q_0 = 10$ initial samples and, in the right column, with $q_0 = d$. Note that the y-axis are not in the same scale as the max and minimal values are different for each case. Figure 6 shows the differences obtained for the 2-dimensional Branin function and batch size $q = 5$ when using the two different initializations. The remaining tabular and graphical results are shown in Annex 11.

With $q_0 = 10$ the UCB single-point algorithm (black line) is the best method. However, when $q_0 = 2$, its behaviour is not good enough and other batch algorithms outperform it. On the contrary, the results for the Levy-5 function improve slightly when using $q_0 = 2$.

Modifying the number of initial samples seems to greatly affect the UCB single-point behaviour. This makes sense because it directly uses the mean and variance functions given by the surrogate model. Those batch acquisition functions that strongly rely on the UCB single-point function will also imitate its bad behaviour when having a low number of initial samples. Those strategies that are capable of extracting information from other sources or are more exploratory approaches can avoid this behaviour and obtain better results, such as, LP-UCB, PE-UCB, Rand-UCB and our proposals.

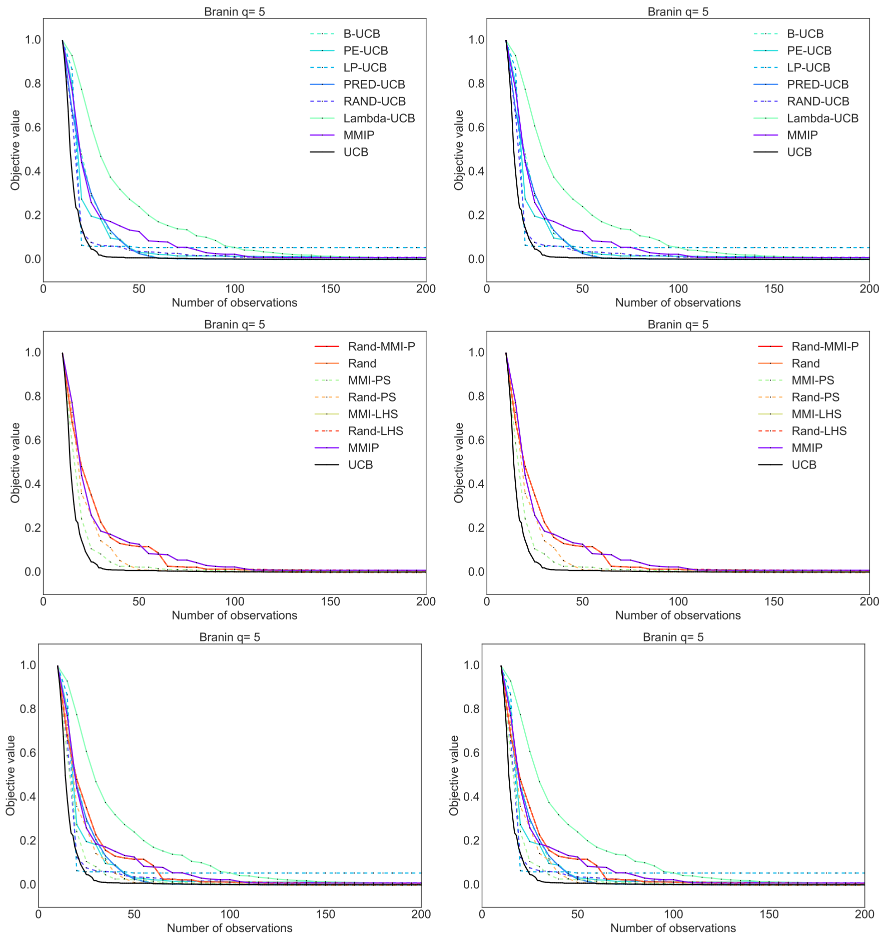

**Figure 6.** Objective function Branin and $q = 4$ (Left $q_0 = 10$, Right $q_0 = d$).

## 8. Conclusions and further work

In this paper a novel batch strategy is proposed called “maximize mutual information from a candidate pool”' (MMIP). In the experimental section, we show that our strategy, when compared with some of the state-of-the-art UCB–based batch approaches, outperforms the results or obtains very similar results depending on the properties of the objective function considered.

For some objective functions, the UCB single-point reference strategy is outperformed by some of the batch strategies tested. In these cases, all the batch algorithms that rely blindly on the UCB behaviour (B-UCB, Pred-UCB and Lambda-UCB) cannot avoid its bad performance. It seems that the mean and uncertainty balance selected by the UCB parameter, even if it is adapted as the number of observations increases, give more weight to the mean before the GP process is mature enough. In general the strategies that apply a more exploratory approach into the batch selection are capable of escaping from this behaviour. For the strategies that we propose, all of them escape from the UCB behaviour, and the MMIP strategy improves the results as the dimensionality, and therefore the complexity, of the problem increases.

Moreover, from the experiments performed with the GP initialization, we conclude that, when the initial GP hyperparameters are not good enough due to the ignorance about the problem properties it could be better to apply strategies that could avoid relying blindly on the UCB balance used. In this work the mean and kernel hyperparameters are fixed at the beginning of the process. An interesting problem would be to adapt these hyperparameters online, see for instance Roman et al. (2016a).

In our approach, the batch size is fixed at the beginning and is constant along the optimization steps. As future work, designing an adaptive batch strategy where the batch size is modified would be interesting. See for instance Azimi, Jalali, and Fern (2012); Ginsbourger et al. (2011); Marmin, Chevalier, and Ginsbourger (2015); Snoek, Larochelle, and Adams (2012).

In Desautels, Krause, and Burdick (2014), the authors introduce, to complement the B-UCB algorithm, the notion of lazy variance calculations. As the authors state, applying the lazy variance computation instead of the traditional one will accelerate the computation of some UCB-based algorithms without any loss of performance. The key idea is that, instead of recomputing $\sigma_t(\boldsymbol{x})$ for all the candidate points at every step $t$, an upper bound can be maintained recomputing its value only at some points. We have not implemented this feature yet, but it would be interesting to apply this in the future and compare the computing times.

In this paper, we assume that the computing time of the objective function is very expensive compared with any of the computing times of the batch algorithms. However, note that different algorithms could reach the same final value but have a different speed in terms of number of evaluations. That is, the number of observations required to obtain a particular result depends on the algorithm. Therefore, depending on the computing time available we might prefer to use one algorithm or another.

## Funding

Carmen-Ana Domínguez-Bravo is partially supported by the Basque Government (BERC 2014-2017 and ELKARTEK), and by the Spanish Ministry of Economy and Competitiveness MINECO (BCAM Severo Ochoa excelence accreditation SEV-2013-0323). Jose A. Lozano is partially supported by the Basque Government (IT609-13), and Spanish Ministry of Economy and Competitiveness MINECO (BCAM Severo Ochoa excellence accreditation SEV-2013-0323 and TIN2016- 78365-R).

## References

Azimi, Javad, Ali Jalali, and Xiaoli Z. Fern. 2012. "Hybrid Batch Bayesian Optimization." In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 315–322. Omnipress.

Bischl, Bernd, Simon Wessing, Nadja Bauer, Klaus Friedrichs, and Claus Weihs. 2014. "MOI-MBO: Multiobjective infill for parallel model-based optimization." In *Learning and Intelligent Optimization*, 173–186. Springer.

Brochu, E., V. M. Cora, and N. de Freitas. 2010. "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning." arxiv.org/abs/1012.2599.

Carrizosa, Emilio, Belén Martín-Barragán, and Dolores Romero Morales. 2014. "A nested heuristic for parameter tuning in support vector machines." *Computers & Operations Research* 43: 328–334.

Contal, Emile. 2016. "Statistical learning approaches for global optimization." PhD diss., Université Paris-Saclay. tel.archives-ouvertes.fr/tel-01396256/.

Contal, Emile, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. 2013. "Parallel Gaussian Process Optimization with Upper Confidence Bound and Pure Exploration." In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science 8188, 225–240. Springer Berlin Heidelberg.

Cox, Dennis D, and Susan John. 1997. "SDO: A statistical method for global optimization." *Multidisciplinary design optimization: state of the art* 315–329.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. "A fast and elitist multiobjective genetic algorithm: NSGA–II." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

Desautels, Thomas, Andreas Krause, and Joel W. Burdick. 2014. "Parallelizing Exploration-exploitation Tradeoffs in Gaussian Process Bandit Optimization." *Journal of Machine Learning Research* 15 (1): 3873–3923.

Feng, Zhiwei, Qingbin Zhang, Qingfu Zhang, Qiangang Tang, Tao Yang, and Yang Ma. 2015. "A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization." *Journal of Global Optimization* 61 (4): 677–694.

Ginsbourger, David, Janis Janusevskis, Rodolphe Le Riche, and others. 2011. "Dealing with asynchronicity in parallel Gaussian Process based global optimization." In *4th International Conference of the ERCIM WG on Computing & Statistics*, Accessed 2014-10-28. hal.archives-ouvertes.fr/hal-00507632/.

González, Javier, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. 2016. "Batch Bayesian optimization via local penalization." In *Artificial Intelligence and Statistics*, 648–657.

González, Javier, Joseph Longworth, David C James, and Neil D Lawrence. 2015. "Bayesian optimization for synthetic gene design." *arXiv:1505.01627 [stat.ML]* arxiv.org/abs/1505.01627.

Greco, Salvatore, J Figueira, and M Ehrgott. 2005. "Multiple criteria decision analysis." *Springer's International series* .

Hutter, Frank, Holger H Hoos, and Kevin Leyton-Brown. 2012. "Parallel algorithm configuration." In *Learning and Intelligent Optimization*, 55–70. Springer.

Izzo, Dario, and Francesco Biscani. 2017. "Pagmo & Pygmo scientific library for massively parallel optimization." esa.github.io/pagmo2/.

Jones, D. R:, M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization* 13: 455–492.

Knowles, Joshua, and Hirotaka Nakayama. 2008. "Meta-modeling in multiobjective optimization." In *Multiobjective optimization*, 245–284. Springer.

Krause, A., A. Singh, and C. Guestrin. 2008. "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies." *Journal of Machine Learning Research* 9: 235–284.

Krityakierne, Tipaluck, Taimoor Akhtar, and Christine A. Shoemaker. 2016. "SOP: parallel surrogate global optimization with Pareto center selection for computationally expensive single objective problems." *Journal of Global Optimization* 1–21.

Marmin, Sébastien, Clément Chevalier, and David Ginsbourger. 2015. "Differentiating the multipoint Expected Improvement for optimal batch design." In *Machine Learning, Optimization, and Big Data*, 37–48. Springer.

Morar, Marius Tudor, J. Knowles, and Sandra Sampaio. 2017. "Initialization of Bayesian Optimization Viewed as Part of a Larger Algorithm Portfolio." In *Proceedinsg of the Data Science meets optimization (DSO) workshop*, .

Rasmussen, C. E., and K. I. Williams. 2006. *Gaussian Processes for Machine Learning.* Massachusetts Institute of Technology.

Roman, I., R. Santana, A. Mendiburu, and J. A. Lozano. 2016a. "An experimental study in adaptive kernel selection for Bayesian Optimization." *Journal of Global Optimization* .

Roman, Ibai. 2017. "Python library for Bayesian Optimization." github.com/ibaidev/bolib.

Roman, Ibai, Josu Ceberio, Alexander Mendiburu, and Jose A Lozano. 2016b. "Bayesian optimization for parameter tuning in evolutionary algorithms." In *IEEE Congress on Evolutionary Computation (CEC)*, 4839–4845.

Sarkar, Dripta, Emile Contal, Nicolas Vayatis, and Frederic Dias. 2016. "Prediction and optimization of wave energy converter arrays using a machine learning approach." *Renewable Energy* 97: 504–517.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams. 2012. "Practical Bayesian optimization of machine learning algorithms." In *Advances in Neural Information Processing systems*, 2951–2959.

Srinivas, N., A. Krause, S. M. Kakade, and M. W. Seeger. 2012. "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting." *IEEE Transactions on Information Theory* 58 (5).

Žilinskas, Antanas, and James Calvin. 2018. "Bi-objective decision making in global optimization based on statistical models." *Journal of Global Optimization* 1–11.