Multi-Objective Bayesian Optimization Framework with Random Scalarizations

Artur Souza
Universidade Federal de Minas Gerais
arturluis@dcc.ufmg.br

Leonardo B. Oliveira
Universidade Federal de Minas Gerais
leob@dcc.ufmg.br

Luigi Nardi
*Lund University**
luigi.nardi@cs.lth.se

***Abstract*—Parameter optimization is quickly becoming a challenging problem in modern solutions for science and industry. As technology advances, solutions grow more complex and start demanding the optimization of a large number of parameters in other to be efficient, a task that becomes challenging even for experienced field experts. Bayesian Optimization has emerged as a powerful tool to address this challenge. Bayesian Optimization is an efficient optimization solution that is able to optimize complicated black-box functions with less function evaluations than most other approaches. Given its recent success, many Bayesian Optimization frameworks have been proposed in the literature. However, most of these frameworks are still not flexible enough to meet all the requirements of real world applications. In particular, most frameworks focus on the optimization of single objective problems, while real world applications often must simultaneously optimize multiple conflicting objectives. To address this issue, we propose a novel multi-objective Bayesian Optimization solution based on random scalarizations. Our solution is able to optimize any number of objectives simultaneously and is also flexible to different parameter types and constrained search spaces. In this paper, we provide a detailed presentation of our solution and show that it achieves competitive results on several synthetic and real world benchmarks.**

***Index Terms*—Optimization Methods; Pareto Optimization; Bayesian Optimization; Automated Machine Learning;**

I. Introduction

Parameter optimization is a recurrent problem in both science and industry. Scientists often need to tune parameters for physical and social experiments, while engineers often have to tune parameters of the machines used to accelerate tasks. As both fields advance, the complexity of both machines and experiments escalate, together with the number of parameters to tune. As the complexity of these problems grow, the optimization of problem parameters becomes a daunting task even for field experts [1].

Bayesian Optimization (BO) has emerged as a powerful solution for parameter optimization problem. Optimization problems can be described as optimizing an unknown (black-box) function f subject to a set of parameters X. BO tackles this problem by building a surrogate model $\mathcal{M}$ to mimic the behavior of f and then uses $\mathcal{M}$ to predict promising parameter combinations [2]. The surrogate model allows BO to find good parameter configurations with few function evaluations, making BO particularly well suited to optimize expensive black box functions [3]. Several BO frameworks have been proposed to tackle the parameter optimization problem, with notable success [4]–[6].

This BO formulation is constrained to problems with a single optimization objective, while many applications require the optimization of multiple, often conflicting, objectives. For instance, when tuning algorithms for hard computational problems, experts wish to simultaneously minimize algorithm runtime and maximize the quality of the solutions found by the algorithm [4]. Similarly, for machine learning models, it is sometimes desired to maximize model accuracy, while minimizing model complexity.

In this paper, we present an approach for multi-objective BO. Mathematically, in a multi-objective setting we consider a set of K objectives $f = (f_1, ..., f_k)$ defined over a search space $\mathbb{X}$. Our goal is to encounter the Pareto frontier of this set of objectives; that is, the set $\Gamma \subseteq \mathbb{X}$ of points that are not dominated by any other point in $\mathbb{X}$. Formally, a point $\boldsymbol{x_1}$ is said to dominate another point $\boldsymbol{x_2}$ if $f_i(\boldsymbol{x_1}) \leq f_i(\boldsymbol{x_2}) \, \forall \, i \in \{1, ..., K\}$ and $\exists j \in \{1, ..., K\}$ such that $f_j(\boldsymbol{x_1}) < f_j(\boldsymbol{x_2})$. We denote dominance by $\boldsymbol{x_1} \prec \boldsymbol{x_2}$ and thus $\Gamma = \{\boldsymbol{x} \in \mathbb{X} : \nexists \boldsymbol{x'}$ such that $\boldsymbol{x'} \prec \boldsymbol{x}\}$. Intuitively, the goal is the set Γ of points that can not be optimized further in any objective without making another objective worse.

Our multi-objective approach is based on random scalarizations, similar to the work of Paria *et al.* [7]. Simply put, at each optimization iteration, we randomly sample a set of weights $\boldsymbol{\lambda}$ and use it to scalarize the multiple objectives into a single value through a scalarization function $g(\boldsymbol{\lambda}, \boldsymbol{x})$. Despite similarities, our work extends previous works in multi-objective BO in two key aspects. First, our solution supports discrete parameter types, i.e., integer, ordinal, and categorical parameters (e.g. the type of kernel for a SVM model). Second, we support constrained search spaces [8], i.e., we are able to optimize functions where some parameter configurations lead to impossible scenarios (e.g. a learning model that exceeds the device's memory). These two features are indispensable for several real world application scenarios (e.g. see Section IV).

This paper is organized as follows. Section II presents a summary of BO and a review of existing BO solutions in the literature. Section III presents our solution in detail. Section IV presents our experiments and results on synthetic functions and a real world application scenario. Finally, Section V presents our conclusions from this work.

This work was partially funded by the IEEE Computational Intelligence Society.

* Work partially conducted as a research scientist at Stanford University.

II. Background

A. Bayesian Optimization

Bayesian Optimization is a framework to solve optimization problems. In BO, we wish to find the minimum (or maximum) of an unknown function over a space of parameters. The function being optimized is often unknown and, thus, referred to as a *black-box function*, while the space of possible parameter configurations is often called the problem's *search space*. The main advantage of BO is that it is one of the most efficient approaches considering the number of black-box function evaluations [3].

BO uses an approach for optimization based on Bayes' theorem. Put simply, BO builds a posterior model on the black-box function based on an initial prior and a set of evaluated configurations (the evidence). BO, then, uses the posterior model to make informed decisions on which configurations should be evaluated. BO uses an iterative algorithm, at each iteration, a new configuration is evaluated and the posterior model is updated with the new evidence[1]. It is common, however, for BO solutions to randomly sample configurations from the search space to bootstrap the model, before starting the iterative BO loop. The BO loop is executed for a pre-specified budget, usually on the number of function evaluations.

The model is a key component of BO solutions. The BO model must accurately predict the behavior of the black-box function and be cheap to evaluate. Common model choices for BO are GPs [7], RFs [4], and Tree-structured Parzen Estimators (TPE) [9]. Another desired property for BO models is to accurately estimate the uncertainty of the model itself. This allows BO to reason over which configurations are worth to explore considering the quality of the solution (exploitation) and the uncertainty regarding the black-box function (exploration). This is the fundamental *exploration x exploitation* trade-off in BO.

Another key component of BO is the acquisition function. The acquisition function dictates which configurations should be explored next, considering the exploration and exploitation trade-off. To be effective, the acquisition function must properly balance exploration and exploitation. Too much focus on exploration will lead to a good understanding of the black-box function, but a weak optimized value. Too much focus on exploitation will lead to a poor understanding of the black-box function and likely lead optimization to a local minimum (or maximum). Common choices of acquisition function are Expected Improvement (EI), Upper Confidence Bound (UCB), and Thompson Sampling (TS) [2].

Algorithm 1 shows the BO algorithm. We refer to the initial random sampling phase, to bootstrap the model, as the *Design of Experiments* (or DoE) phase. In the next sections, we review existing BO solutions (Section II-B) and then present our own approach (Section III, including how we extend this BO framework to the multi-objective setting, our choices of model and acquisition function, how we handle constrained search spaces, and other design decisions.

[1]We note that some BO solutions choose to update the model after a number of iterations, instead of every iteration. Usually to save computation time or allow for parallelism.

Algorithm 1 Bayesian optimization.

1: **Input:** Design space $\mathbb{X}$, DoE sampling size N, and optimization budget B.
2: **Output:** Array of explored configurations $\boldsymbol{D}$.
3: $\boldsymbol{X_{doe}} \leftarrow doe\_sample(\mathbb{X}, N)$
4: $\boldsymbol{y} \leftarrow evaluate(\boldsymbol{X_{doe}})$
5: $\boldsymbol{D} \leftarrow (\boldsymbol{X_{doe}}, \boldsymbol{y})$
6: **for** $t = 1$ **to** B **do**
7: $\quad \mathcal{M} \leftarrow fit\_model(\boldsymbol{D})$
8: $\quad \boldsymbol{x^*} \leftarrow \arg\max_x acq(\mathbb{X}, \mathcal{M})$
9: $\quad \boldsymbol{y^*} \leftarrow evaluate(\boldsymbol{x^*})$
10: $\quad \boldsymbol{D} \leftarrow \boldsymbol{D} \cup (\boldsymbol{x^*}, \boldsymbol{y^*})$
11: **end for**
12: **return** $\boldsymbol{D}$

B. Related Work

Several authors have worked on BO, proposed their own BO solutions, and steadily advanced the state-of-the-art over the years. In this section, we review some prominent BO solutions in the literature, with a focus on BO frameworks. For a more comprehensive analysis on the different components of BO, we refer to the survey by Shahriari *et al.* [2].

One of the earliest and most prominent BO solutions in the literature is the work of Jones *et al.* [10]. Jones *et al.* propose a BO solution they call Efficient Global Optimization (EGO), built on top of the DACE model and the EI acquisition function. The essence of the DACE model lies on a correlation function that treats the model's noise as a function of the distance between points in the space. Turns out this correlation function is so effective that the authors can simply assume a constant mean for their stochastic process. Jones *et al.* adopt this model and then use a branch-and-bound algorithm to optimize an EI acquisition function, completing the EGO framework.

The most frequently used model for BO is GP, with several authors having proposed BO frameworks based on GPs. Snoek *et al.* [11], for instance, propose a BO solution based on GPs dubbet Spearmint. The key insight of Spearmint is a bijective warping on the inputs to allow the GP to remove the stationarity of GP models. This allows their solution to handle applications where the black-box function behaves differently in different regions of the space, making it more flexible. Similarly, Gardner *et al.* [8] propose a constrained BO solution based on GPs. Their solution models a separate GP on a cost function and, based on a pre-defined cost threshold, define a "probability of feasibility" for each configuration in the search space. This probability of feasibility allows their solution to handle constrained search spaces, where some parameter configurations are impossible to evaluate.

As an alternative to GPs, Bergstra *et al.* [9] have proposed the TPE model for BO. The TPE model works by simultaneously building two densities over the search space, one for configurations that lead to good values ($l(\boldsymbol{x})$) and

another for configurations that leads to bad values ($g(\boldsymbol{x})$). Bergstra *et al.* show that an EI acquisition function can be maximized by simply maximizing the ratio $l(\boldsymbol{x})/g(\boldsymbol{x})$, i.e., finding configurations that have high probability under $l(\boldsymbol{x})$ and low probability under $g(\boldsymbol{x})$. The resulting TPE BO framework is competitive with other GP-based solutions, while also being more efficient for high-dimensional problems. The TPE optimization framework is openly available via the Hyperopt library [5].

As another alternative to GPs, Hutter *et al.* propose a BO framework based on a RF model. Their solution, dubbed Sequential Model-Based Algorithm Configuration (SMAC), uses a RF model and the EI acquisition function, optimized with a multi-start local search. The RF model allows SMAC to support discrete and categorical input parameters, a limitation of GPs and TPE. Hutter *et al.* show that their RF-based solution performs better than previous solutions, while also being more flexible.

In a multi-objective setting, Knowles proposes an extension of the EGO algorithm for multiple objectives called ParEGO [12]. ParEGO uses the augmented tchebyshev scalarization function and a set of scalarizing weights to scalarize multiple objectives into a single value. ParEGO then applies the EGO algorithm of Jones *et al.* on the scalarized objectives. The set of different scalarization weights used allows ParEGO to find different configurations on the Pareto front for each weight combination. Knowles compares ParEGO to state-of-the-art multi-objective evolutionary optimization algorithms and show that ParEGO is able approximate the Pareto front better when the number of black-box function evaluations is limited.

At last, Paria *et al.* propose an approach for multi-objective BO using random scalarizations [7]. The approach of Paria is based on GPs and uses a combination of randomly sampled weights and scalarization functions to scalarize objectives. A novelty of Paria *et al.* 's solution is to use scalarized versions of the UCB and TS acquisition functions, rather than scalarizing the model and then applying the acquisition function to the scalarized models. Paria *et al.* prove the convergence of their solution and show that their solution performs better than previous solutions, including ParEGO and multi-objective evolutionary algorithms.

Our approach is similar to that of Paria *et al.* , however, we deviate from their solution in key aspects that render our solution more flexible. First, we replace the GP with a specialized RF model, which allows us to support integer, ordinal, and categorical parameters, besides real parameters. This flexibility is important as many real world applications require the optimization of these discrete parameters. Further, we implement a constrained BO approach, based on the work of Gardner *et al.* [8], to allow us to handle constrained search spaces. Once again, this is desirable in real world applications where some parameter combinations result in impossible configurations. Our model is described in detail in the following section.

III. Bayesian Optimization with Random Scalarizations

This section presents the various components that compose our multi-objective BO approach. Our entire algorithm for BO with random scalarizations is shown in Algorithm 2. A depiction of the algorithm is shown in Figure 1.

A. Random Scalarizations

The essence of our approach lies in the scalarization functions. Given a weight distribution $\mathcal{L}$ defined on the simplex $\lambda \in \mathbb{R}$, $||\lambda||_1 = 1$, a scalarization function is defined as $g(\boldsymbol{\lambda}, \boldsymbol{x})$ where $\boldsymbol{\lambda} \sim \mathcal{L}$ and $g : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a function producing scalar values defined over $\mathbb{X}$ and the support of $\mathcal{L}$. The function $g(\boldsymbol{\lambda}, \boldsymbol{x})$ is chosen such that optimizing $g(\boldsymbol{\lambda}, \boldsymbol{x})$ with respect to $\boldsymbol{x}$ for a given $\boldsymbol{\lambda}$ leads to a point in the Pareto front. Thus, optimizing $g(\boldsymbol{\lambda}, \boldsymbol{x})$ with respect to $\boldsymbol{x}$ for different $\lambda x \sim \mathcal{L}$ leads to different points in the Pareto front. Thus, $\mathcal{L}$ can be seen as a distribution over the Pareto front [7].

We implement three scalarization functions in our BO approach. Two scalarization functions were presented by Paria *et al.* (Equations 1 and 2) and the third is the augmented Tchebyshev scalarization function (equation 3) [13]. We dub the functions *linear scalarization* (Eq. 1), *Tchebyshev scalarization* (Eq. 2), and *augmented Tchebyshev scalarization* (Eq. 3).

The linear scalarization function is defined as:

$$g_{lin}(\lambda, x) = \sum_{k=1}^{K} \lambda_k f_k(x) \tag{1}$$

However, as presented by Paria *et al.* , the linear scalarization function fails to explore non-convex regions of the Pareto front. To address this, Paria *et al.* propose their version of the Tchebyshev scalarization function, defined as:

$$g_{tch}(\lambda, x) = \min_{k=1}^{K} \lambda_k (f_k(x) - z_k^*) \tag{2}$$

Where z_k^* is an ideal reference point, often taken to be the best possible value of the objectives.

We have also implemented the augmented Tchebyshev scalarization function as found in [12], [13]. The augmented Tchebyshev scalarization is defined as:

$$g_{tch_a}(\lambda, x) = \max_{k=1}^{K} \lambda_k f_k(x) + \alpha \sum_{k=1}^{K} \lambda_k f_k(x) \tag{3}$$

Where α is the augmentation constant, set to $\alpha = 0.05$ as in ParEGO [12].

B. Random Forest Model

We replace Paria *et al.* 's GP with a RF model. Using RFs allow our solution to support ordinal and categorical variables. We use an adapted RF model proposed by Hutter *et al.* [14], which is better suited for BO. Namely, we treat the RF as a mixture model of T trees, compute a mean μ_t and variance σ_t^2 for each tree t, and assume a predictive distribution: $\mathcal{N}(\mu, \sigma^2)$ with:

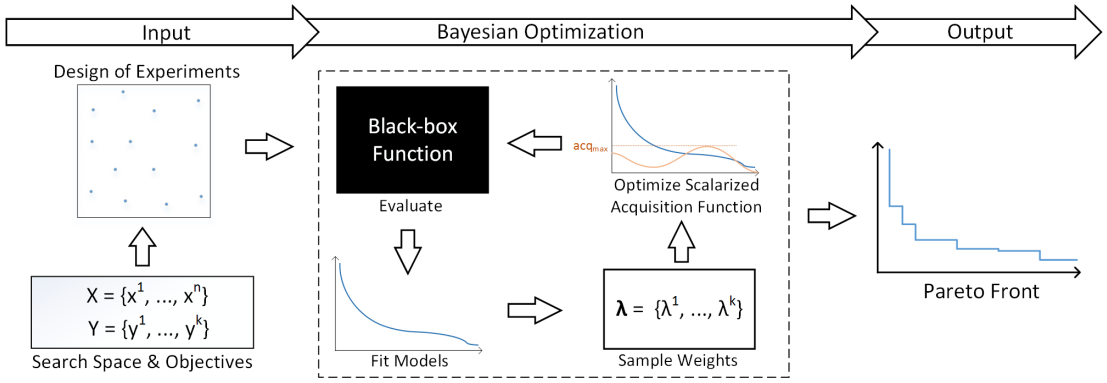

Fig. 1. Multi-objective BO with random scalarizations.

Algorithm 2 Multi-objective optimization with random scalarizations.

Input: Design space $\mathbb{X}$, DoE sampling size N, optimization budget B, weight distribution $\mathcal{L}$. K objectives and C constraints.
Output: Design space exploration array $\boldsymbol{D}$.
$\boldsymbol{X_{doe}} \leftarrow doe\_sample(\mathbb{X}, N)$
$\boldsymbol{y_k} \leftarrow evaluate(\boldsymbol{X_{doe}}) \, \forall \, 1 \leq k \leq K$
$\boldsymbol{D} \leftarrow (\boldsymbol{X_{doe}}, \boldsymbol{y_1}, ..., \boldsymbol{y_K})$
for $t = 1$ **to** B **do**
  $\mathcal{M}_k \leftarrow fit\_model(\boldsymbol{D}) \, \forall \, 1 \leq k \leq K$
  $PF_c \leftarrow fit\_model(\boldsymbol{D}) \, \forall \, 1 \leq c \leq C$
  Sample $\boldsymbol{\lambda^t} \sim \mathcal{L}$
  $\boldsymbol{x^*} \leftarrow \arg\max_x acq(\mathbb{X}, \boldsymbol{\lambda^t}, \mathcal{M}_1, ..., \mathcal{M}_K PF_1, ..., PF_C)$
  $\boldsymbol{y_k^*} \leftarrow evaluate(\boldsymbol{x^*}) \, \forall \, 1 \leq k \leq K$
  $\boldsymbol{D} \leftarrow \boldsymbol{D} \cup (\boldsymbol{x^*}, \boldsymbol{y^*})$
end for
return $\boldsymbol{D}$

$$\mu = \frac{1}{T} \sum_{t=1}^{T} \mu_t \tag{4}$$

$$\sigma^2 = \left(\frac{1}{T} \sum_{t=1}^{T} \sigma_t^2 \right) + \frac{1}{T} \left(\sum_{t=1}^{T} \mu_t^2 \right) - \mu^2 \tag{5}$$

We also change the way splits are selected in the trees, as proposed by Hutter *et al.* . By default, the split point is chosen in the middle point between the two samples at the boundary of the split regions. Instead, we uniformly sample a point for the split in the interval defined by the boundary samples. That is, we sample the split point as $s \; U(x_l, x_u)$, where x_l and x_u are the (lower and upper) boundary samples. This causes the variance to vary smoothly in the space, rather than being divided into constant regions.

C. Acquisition Function

We implement three acquisition functions. Two of them (TS and UCB) are implemented as described by Paria *et al.* , adapted to accommodate our different model. The third is an Expected Improvement acquisition function.

The first acquisition function we implement is Thompson Sampling (TS). With GPs, the idea behind TS is to randomly sample a function from the GP and use it as the surrogate model. The randomly sampled function introduces exploration, while optimizing this function favors exploitation. To capture this idea with RFs, we train n RF models with subsets of the data, through bootstrapping, and, then, randomly sample one of these models to use as our surrogate [15][2]. Simply combining this sampled model with the scalarizations defined in Section III leads to the scalarized TS function that we:

$$g_{lin}(\lambda, x) = \sum_{k=1}^{K} \lambda_k f'_k(x) \tag{6}$$

$$g_{tch}(\lambda, x) = \min_{k=1}^{K} \lambda_k (f'_k(x) - z^*_k) \tag{7}$$

$$g_{tch_a}(\lambda, x) = \max_{k=1}^{K} \lambda_k f'_k(x) + \alpha \sum_{k=1}^{K} \lambda_k f'_k(x) \tag{8}$$

Where f' is a RF model fitted through bagging.

The second acquisition function we implement is the Upper Confidence Bound (UCB). The key idea of UCB is to be optimistic and consider the upper confidence of the prediction for each point instead of the mean. In order to compute UCB, we compute the empirical mean and variance for our RF model and use them to compute UCB as:

$$g_{lin}(\lambda, x) = \sum_{k=1}^{K} \lambda_k \mu_k(x) + \sqrt{\beta_t} \sqrt{\sum_{k=1}^{K} \lambda_k^2 \sigma_k^2(x)} \tag{9}$$

$$g_{tch}(\lambda, x) = \min_{k=1}^{K} \lambda_k (\mu_k(x) + \sqrt{\beta_t} \sigma_k(x)) \tag{10}$$

[2]This is equivalent to training a single RF model on a random subset of the data. We follow this simplified approach in our implementations to make it more efficient.

$$g_{tch_a}(\lambda, x) = \max_{k=1}^{K} \lambda_k(\mu_k(x) + \sqrt{\beta_t}\sigma_k(x)) + \alpha \sum_{k=1}^{K} \lambda_k(\mu_k(x) + \sqrt{\beta_t}\sigma_k(x)) \quad (11)$$

β_t is a UCB parameter set as $\beta_t = 0.125 \log(2t+1)$, where t is the iteration number [7].

Finally, the third acquisition function we implement is EI. EI is an improvement-based acquisition function that prioritizes points that are more likely to improve on the best function value found so far. This improvement is encoded in an improvement function:

$$I(x) = (f_{min} - f(x))\mathbb{I}(f_{min} > f(x)) \quad (12)$$

Where f_{min} is the lowest function value found so far and $\mathbb{I}$ is the indicator function. The EI takes the expectation of the improvement function to decide where to explore next. This is computed analytically as:

$$EI(x) = (f_{min} - f(x))\Phi\left(\frac{(f_{min} - f(x))}{\sigma}\right) + \sigma\phi\left(\frac{(f_{min} - f(x))}{\sigma}\right) \quad (13)$$

Where Φ and ϕ are the normal cumulative distribution function and probability distribution function, respectively. After computing the EI for each objective, we scalarize them with:

$$g_{lin}(\lambda, x) = \sum_{k=1}^{K} \lambda_k EI_k(x) \quad (14)$$

$$g_{tch}(\lambda, x) = \min_{k=1}^{K} \lambda_k(EI_k(x) - z_k^*) \quad (15)$$

$$g_{tch_a}(\lambda, x) = \max_{k=1}^{K} \lambda_k EI_k(x) + \alpha \sum_{k=1}^{K} \lambda_k EI_k(x) \quad (16)$$

D. Optimizing the Acquisition Function

We also depart from the reference paper in the optimization of the acquisition function. Paria *et al.* use the DIRECT algorithm to optimize their acquisition functions [16]. However, the DIRECT algorithm does not work well with categorical parameters [17]. Instead, we use a multi-start local search to optimize our acquisition functions.

We implement a best improvement multi-start local search similar to the one proposed in SMAC [4]. First, we compute the scalarization of all the previously evaluated configurations, pick the 10 best performing points, and start local searches on each of these points. Additionaly, we randomly sample 10,000 points, compute their scalarized values, and start local searches on the 10 best performing random points.

Algorithm 3 Multi-start best improvement local search.

Input: Design space $\mathbb{X}$, previously evaluated points $\boldsymbol{D_p} = \{\boldsymbol{X_p}, \boldsymbol{Y_p}\}$, number of random samples N_r, number of local search starting points N_s, function to minimize f.
Output: Best configuration found by the local search.
$\boldsymbol{D_p} \leftarrow get\_best(\boldsymbol{D_p}, N_s)$
$\boldsymbol{X_r} \leftarrow random\_sample(\mathbb{X}, N_r)$
$\boldsymbol{Y_r} \leftarrow f(\boldsymbol{X_r})$
$\boldsymbol{D_r} \leftarrow (\boldsymbol{X_r}, \boldsymbol{y_r})$
$\boldsymbol{D_r} \leftarrow get\_best(\boldsymbol{D_r}, N_s)$
$\boldsymbol{D} \leftarrow D_p \cup D_r$
$\boldsymbol{x_{best}}, \boldsymbol{y_{best}} \leftarrow get\_best(\boldsymbol{D}, 1)$
for each $(\boldsymbol{x}, \boldsymbol{y})$ **in** $\boldsymbol{D}$ **do**
  $end\_of\_search = False$
  while $end\_of\_search = False$ **do**
    $\boldsymbol{X_n} \leftarrow get\_neighbors(\boldsymbol{x})$
    $\boldsymbol{x^*} \leftarrow \arg\min_{\boldsymbol{x} \in \boldsymbol{X_n}} f(\boldsymbol{x})$
    $\boldsymbol{y^*} \leftarrow f(\boldsymbol{x^*})$
    if $\boldsymbol{y^*} < \boldsymbol{y}$ **then**
      $\boldsymbol{x} \leftarrow \boldsymbol{x^*}$
      $\boldsymbol{y} \leftarrow \boldsymbol{y^*}$
    else
      $end\_of\_search = True$
    end if
  end while
  if $\boldsymbol{y} < \boldsymbol{y_{best}}$ **then**
    $\boldsymbol{x_{best}} \leftarrow \boldsymbol{x}$
    $\boldsymbol{y_{best}} \leftarrow \boldsymbol{y}$
  end if
end for
return $\boldsymbol{x_{best}}$

We follow SMAC's one-exchange approach to generate the neighborhoods for the local search. For categorical and ordinal parameters, we generate all neighbors that differ in the value of exactly one categorical parameter. For real and integer parameters, we normalize the range of each objective to $[0, 1]$ and randomly sample four neighbors from a truncated gaussian centered at the original parameter value v and standard deviation $\sigma = 0.2$.

E. Constrained Optimization

We also complement our BO solution with the constrained Bayesian Optimization (cBO) approach proposed by Gardner *et al.* [8]. When constraints are present, the optimum is the point that maximizes (or minimizes) the acquisition function weighed by the probability of that point being feasible. Thus, the acquisition functions become:

$$acq_c(\boldsymbol{x}, \boldsymbol{\lambda}) = PF(\boldsymbol{x})acq(\boldsymbol{x}, \boldsymbol{\lambda}) \quad (17)$$

Where $PF(\boldsymbol{x})$ is the probability of point $\boldsymbol{x}$ being feasible. In our solution, we use a RF classifier to model $PF(\boldsymbol{x})$, instead of a GP as originally proposed. The RF classifier allows us to handle mixed parameter types in the feasibility constraint as well as unknown feasibility constraints.

TABLE I
SYNTHETIC FUNCTIONS USED TO EVALUATE AND COMPARE OUR SOLUTION TO SMAC.

Function	Input parameters	Search Space
1d Branin	1	$x = [-5, 10]$
1d CurrinExp	1	$x = [0, 1]$
Branin	2	$x_1 = [-5, 10]$, $x_2 = [0, 15]$
CurrinExp	2	$x_1 = [0, 1]$, $x_2 = [0, 1]$
Ellipsoidal	4	$x_i = [-5, 5] \forall 1 \leq i \leq 4$
10d Counting Ones	10	$x_i = \{0, 1\} \forall 1 \leq i \leq 10$

F. ϵ-greedy

At last, we implement an ϵ-greedy approach during our BO. That means we choose the best configuration (according to our acquisition function) to explore (1 - ϵ)% of the time and randomly choose a configuration to explore ϵ% of the time. By default, we set $\epsilon = 0.05$. The randomly sampled configurations mean our solution will not become exceedingly greedy in its exploitation and also means our solution is guaranteed to converge.

IV. RESULTS

In this section we evaluate our solution on a set of synthetic functions as well as a real world application with the Spatial programming language [18].

A. Synthetic Functions

We evaluate our solution on a set of six synthetic functions. We compare all functions to SMAC, a state-of-the-art tool also based on RF models. To enable comparison to SMAC, we use synthetic functions with a single optimization objective. We compare both solutions based on their simple regret, that is, the difference between the best solution found by each tool and the true global minimum of each function. We also compare our tools with two different DoE approaches, random sampling and latin hypercube sampling. By default, SMAC uses SOBOL for its DoE phase. We used the Tchebyshev scalarization and EI acquisition function on all benchmarks.

The synthethic functions used are summarized in Table I. Branin[3] and CurrinExp[4] are common benchmark functions in the literature. For their monodimensional counterparts, we fixate one of the input parameters at the one of the optima[5] and optimize the other parameter. The ellipsoidal function was taken from the Black-Box Optimization Benchmarking Workshop [19], with $x^{opt} = 1$. The counting ones function is a benchmark where input parameters can be either 0 or 1 and the goal is to minimize the sum of all parameters.

Figure 2 shows the comparison between our solution and SMAC. We allow each method to perform a total of 60 function evaluations for each function. For the monodimensional cases, 10 of the 60 evaluations are used for DoE, for the other cases, 15 of the 60 evaluations are used for DoE. Each figure shows the mean and standard deviation of the regret after 10 executions.

Figure 2 shows that our method is competitive with SMAC on the single-objective optimization problem. Out of six benchmarks, our method performed better in three, tied in one, and lost in two. Further, we note that our function found the exact global optimum in two benchmarks and got extremely close to the global optimum on two others (0.01 and 0.1 regret). We also note that random sampling DoE worked better than, or equal to, Latin Hypercube DoE on five out of six benchmarks.

B. Spatial Programming Language

We next evaluate our method in two of the Spatial Programming Language benchmarks. Spatial [18] is a domain-specific language (DSL) and corresponding compiler for the design of application accelerators on reconfigurable architectures. Both benchmarks are composed solely of ordinal and categorical parameters, the number of input parameters and the total size of the search space (in number of possible configurations) are shown in Table I.

First, we evaluate the performance of our method on the BlackScholes benchmark. Here, we wish to optimize both the runtime (in Cycles) and the architecture area usage (in logic units used). The search space is also constrained on the area used, since some configurations may exceed the number of logic units available, naturally, an impossible configuration.

Figure 3 shows a comparison between the Pareto front found by our method and the real Pareto front. 1000 random points were used as DoE and 500 points were evaluted during BO. We used the linear scalarization and the TS acquisition function. We note that the Pareto front found is nearly identical to the real Pareto front, thus, our method got close to the true optimum.

Next, we evaluate our solution on the more challenging Molecular Dynamics Grid benchmark. Here, we wish to optimize only the runtime, but still constrain the search space on the area used. Due to the size of the search space, it is unfeasible to compute the true optimum for this benchmark. Instead, we compare the performance of our BO method with random sampling (with up to 8x the budget of BO) and a multi-start local search similar to the one used to optimize our acquisition functions.

Figure 4 shows a comparison of the simple regret for each optimization method. The X-axis shows the number of samples evaluated, where for each Nx random sampling, the method was allowed to evaluate N configurations for each configuration the BO and local search methods evaluated. Our BO method consistently found the best minimum for the benchmark of all methods considered, being considerably superior to all random

TABLE II
SPATIAL BENCHMARKS USED FOR OUR REAL WORLD APPLICATION EXPERIMENTS.

Benchmark	Input parameters	Search Space Size
BlackScholes	4	7.6810^4
Molecular Dynamics	11	$1.6x10^9$

[3]https://www.sfu.ca/ ssurjano/branin

[4]https://www.sfu.ca/ ssurjano/curretal88exp

[5]we set $x_2 = 2.275$ for the Branin and $x_2 = 1$ for CurrinExp

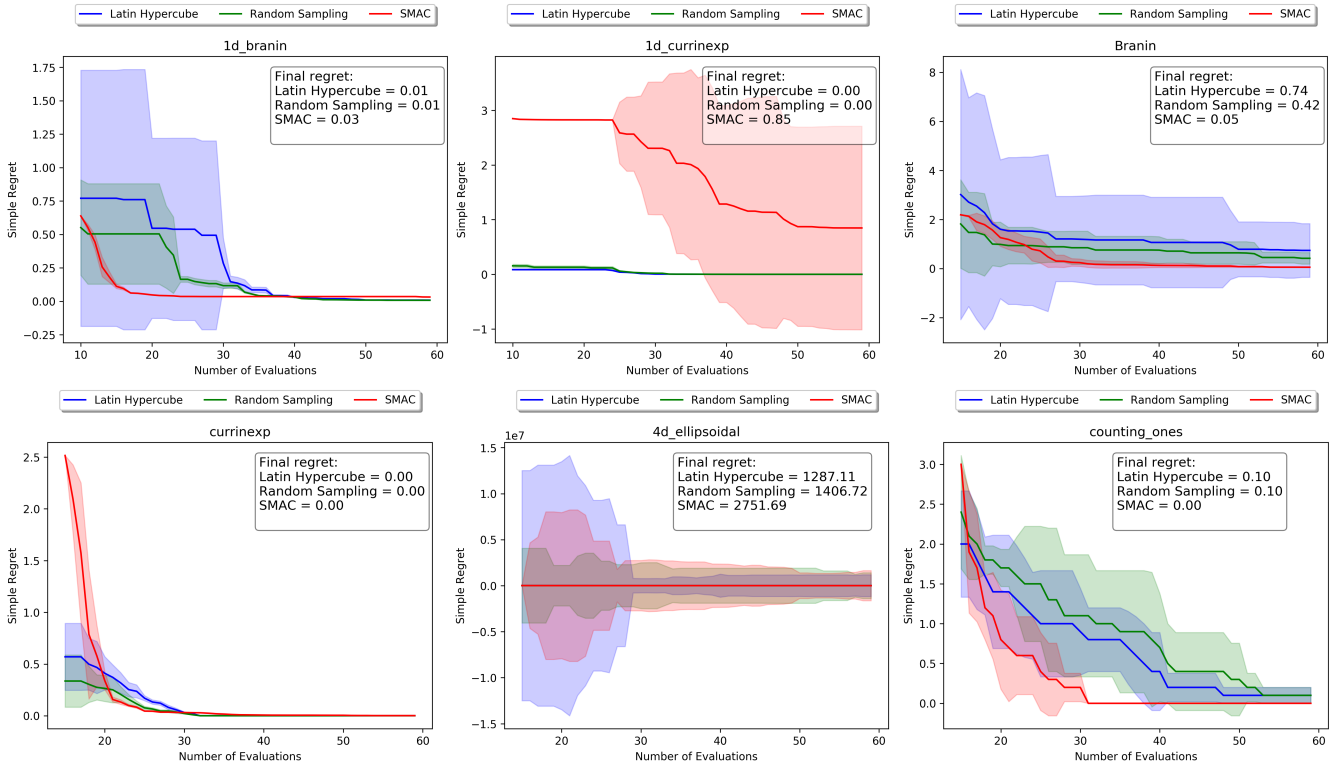

Fig. 2. Simple regret comparison between our solution and SMAC. The solid line shows the mean after 10 executions, while the shaded area shows the standard deviation. Both methods had the same budget for DoE and BO. For our method, we experiment with two DoE types: random sampling and Latin Hypercube sampling. SMAC, by default, uses SOBOL for DoE.

sampling approaches, even with a 8x smaller budget. The local search performed second best, being consistently better than all random sampling approaches.

V. Conclusion

In this paper, we introduced our own multi-objective BO solution based on random scalarizations. Our approach builds on top of previous works in the literature and expands the state-of-the-art by supporting different parameter types and constrained search spaces. We compare our approach to a state-of-the-art optimization framework, showing that our framework is able to produce competitive results, while also being more flexible. We also evaluate our solution on two Spatial DSL benchmarks and show that our solution can be effectively used in real world application scenarios.

References

[1] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.

[2] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

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

[4] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential Model-based Optimization for General Algorithm Configuration. In *International Conference on Learning and Intelligent Optimization*, 2011.

[5] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, 2013.

[6] Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. Gpflowopt: A bayesian optimization library using tensorflow. *arXiv preprint arXiv:1711.03845*, 2017.

[7] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A Flexible Multi-Objective Bayesian Optimization Approach using Random Scalarizations. *CoRR*, abs/1805.12168, 2018.

[8] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian Optimization with Inequality Constraints. In *ICML*, 2014.

[9] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[10] 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, 1998.

[11] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014.

[12] Joshua Knowles. ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. Technical Report TR-COMPSYSBIO-2004-01, University of Manchester, September 2004.

[13] Hirotaka Nakayama, Yeboon Yun, and Min Yoon. *Sequential Approximate Multiobjective Optimization Using Computational Intelligence*. Springer Science & Business Media, 2009.

[14] Holger H. Hoos Kevin Leyton-Brown Frank Hutter, Lin Xu. Algorithm Runtime Prediction: Methods & Evaluation. *CoRR*, abs/1805.12168, 2013.

[15] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian ban-

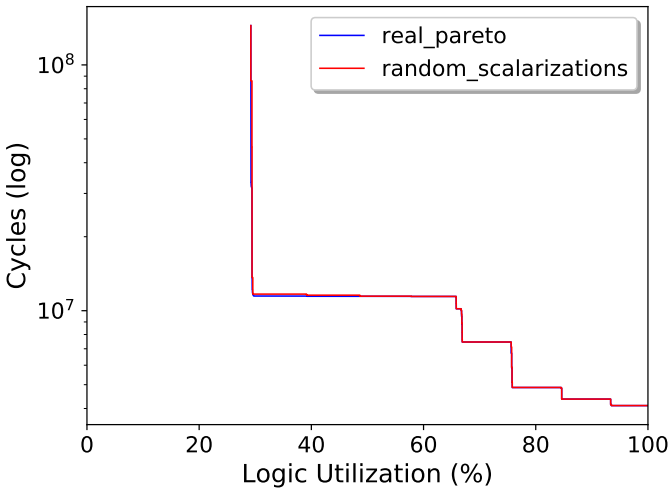

Fig. 3. Comparison between the Pareto front found for the BlackScholes Spatial benchmark and the true Pareto front. The TS acquisition function and linear scalarization function were used. A total of 1500 configurations were evaluated, 1000 of those during DoE.

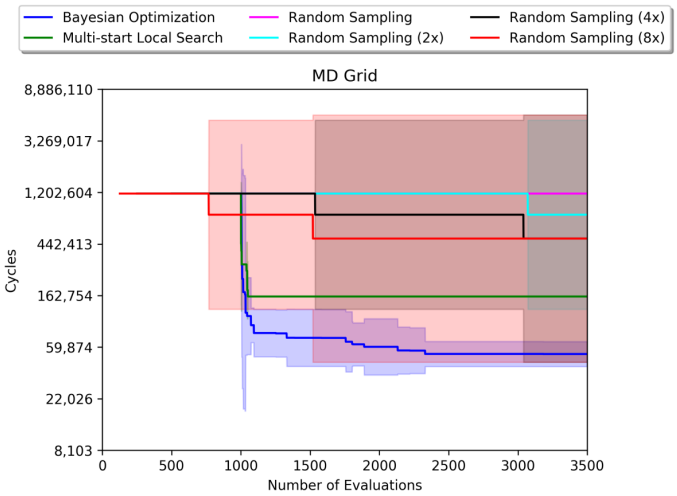

Fig. 4. Regret comparison of different optimization methods on the Molecular Dynamics Grid benchmark. The solid line shows the average regret and the shaded area shows the standard deviation after 10 runs. Each Nx random sampling was allowed to evaluate N samples for each sample the BO/local search methods evaluated.

dits showdown. In *International Conference on Learning Representations*, 2018.

[16] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian Optimization Without the Lipschitz Constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993.

[17] Donald Jones. The direct global optimization algorithm. *Encyclopedia of Optimization*, 1, 01 2001.

[18] David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, et al. Spatial: A language and compiler for application accelerators. In *ACM Sigplan Notices*, volume 53, pages 296–311. ACM, 2018.

[19] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical report, Citeseer, 2010.