Proceedings of Machine Learning Research 191:38–50, 2022 ECMLPKDD Workshop on Meta-Knowledge Transfer

# Searching in the Forest for Local Bayesian Optimization

**Difan Deng** DENG@TNT.UNI-HANNOVER.DE
**Marius Lindauer** LINDAUER@TNT.UNI-HANNOVER.DE
*Leibniz University Hannover*

**Editors:** P. Brazdil, J. N. van Rijn, H. Gouk and F. Mohr

## Abstract

Because of its sample efficiency, Bayesian optimization (BO) has become a popular approach in dealing with expensive black-box optimization problems, such as hyperparameter optimization (HPO). Recent empirical experiments showed that the loss landscapes of HPO problems tend to be more benign than previously assumed, i.e. in the best case uni-modal and convex, such that a BO framework could be more efficient if it can focus on those promising local regions. In this paper, we propose BOinG, a two-stage approach that is tailored toward HPO problems. In the first stage, we build a scalable global surrogate model with a random forest to describe the overall landscape structure. Further, we choose a promising subregion via a bottom-up approach to the upper-level tree structure. In the second stage, a local model in this subregion is utilized to suggest the point to be evaluated next. Empirical experiments show that BOinG is able to exploit the structure of typical HPO problems and performs particularly well on various problems from synthetic functions and HPO.

## 1. Introduction

Hyperparameter optimization (HPO) is considered to be a tedious, error-prone, and expensive problem, but nevertheless crucial for achieving peak performance of a machine learning algorithm on a given dataset (Bergstra and Bengio, 2012; Feurer and Hutter, 2019). Here we assume that a user-defined cost function $f$ is optimized over a space $\mathcal{X}$ of possible hyperparameter configurations $\boldsymbol{x} \in \mathcal{X}$: $\boldsymbol{x}^* \in \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. Typical cost functions $f$ include, for example, accuracy for classification or RMSE for regression either over a hold-out validation set or a $k$-fold cross validation on the training set (Thornton et al., 2013). Since HPO is often treated as an expensive black-box problem, where only a few function evaluations can be afforded and no gradient is available, Bayesian Optimization (BO; Jones et al. (1998)) is a common and efficient approach for obtaining well-performing hyperparameter configurations (Bergstra et al., 2013; Eriksson et al., 2019; Hutter et al., 2012; Snoek et al., 2012).

However, previous work on HPO showed two important insights: First, the loss landscape of a hyperparameter optimization problem is more benign than what one would expect (Klein et al., 2017; Pimenta et al., 2020; Pushak and Hoos, 2018). In most cases, the loss landscape in well-performing regions is quite flat and the best-performing region is fairly well defined, see for example Figure 1. Given a limited HPO budget, we prefer to focus more on identified best-performing regions and avoid unnecessary exploration in the later stages of the optimization. Second, different surrogate models, used to trade-off exploration and exploitation, perform well depending on the task at hand (Eggensperger et al., 2013).

Inspired by recent advances in exploiting local structures in BO (Eriksson et al., 2019; Wan et al., 2021; Wang et al., 2020), we address these insights by proposing a two-stage BO algorithm. Our approach treats the optimization problem on different scales: in the first stage, i.e. the global level, we execute a full BO iteration with all observed points that are evaluated previously. Then we extract a subregion based on the trained surrogate model and the suggested point given by the global optimizer. In the second stage, i.e. the local level, we perform another BO iteration with only the points inside the subregion and train a local surrogate model. Finally, we propose the sample to be evaluated next on the local model. Since we combine the best of two surrogates by extracting the subregion on the global level from a random forest (RF) and performing BO with a Gaussian Process (GP) inside this subregion, our robust HPO method is hence named as *Bayesian Optimization inside a Grove (BOinG)*.

Our contributions are as follows:

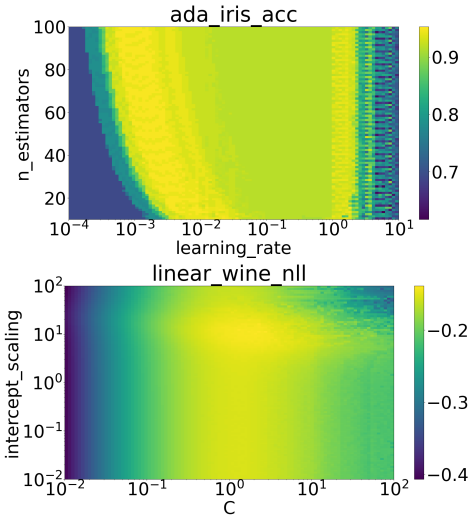


Figure 1: Exemplary hyperparameter loss landscape generated on BayesMark (Turner, 2019). (top) loss landscape of AdaBoost models on the iris datasets. (bottom) loss landscape of linear regression models on the wine datasets

1. We propose the two-stage Bayesian Optimization approach BOinG that allows locating a promising subregion that should be explored more.

2. By using a scalable global model (e.g., RF), BOinG reduces the computational burden on the local model (e.g., GP), which scales better to more iterations than vanilla GP-BO.

3. We propose to augment the local Gaussian Process with global data distribution such that the model captures the local loss landscape while preserving the influence of the global data distribution at a minimal cost.

4. We show the robustness and state-of-the-art performance of BOinG on several synthetic functions and HPO benchmarks for deep learning and reinforcement learning.

## 2. Related Work & Background

Bayesian Optimization (BO) became a promising approach in solving expensive black-box functions (Shahriari et al., 2016). Recent progress on BO focuses on extending BO to large scale and high dimensional search space (Eriksson et al., 2019; Kandasamy et al., 2015; Wang et al., 2020, 2018). BO needs to employ a surrogate model to describe the possible data distribution of the target function. A GP model is the commonly utilized surrogate

model. The predicted mean and variance of a GP for given configuration $\boldsymbol{x}$ is given by

$$\mu(\boldsymbol{x}) = \mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \tag{1}$$

$$\sigma^2(\boldsymbol{x}) = \mathbf{K}(\boldsymbol{x}, \boldsymbol{x}) - \mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_* \tag{2}$$

where $\mathbf{k}_*$ is the covariance vector between $\boldsymbol{x}$ and all previous observations, while $\mathbf{K}$ is the covariance matrix of all previously evaluated points.

One drawback of GPs is its $\mathcal{O}(n^3)$-complexity for fitting the $n$ previous observations and $\mathcal{O}(n^2)$ for predicting means and variances at query points. Several approximate models such as sparse Gaussian processes (Candela and Rasmussen, 2005) have been proposed to alleviate this issue by only using a set of additional points as inducing points to approximate the data distributions. Sparse GP reduces the complexity to $\mathcal{O}(m^2n)$ and $\mathcal{O}(mn)$ for fitting and predicting respectively, where $m$ is the number of the inducing points. Additionally, the introduction of variational inference (Titsias, 2009) brings great benefit to training a sparse GP, e.g., one can optimize a sparse GP w.r.t. each points individually (Hensman et al., 2013), apply natural gradient (Salimbeni et al., 2018) for faster optimization, etc. However, in exchange, sparse GP leads to poor variance estimation that could mislead the optimizer (Shahriari et al., 2016). Despite its various complexity issues, GPs remain the most widely used surrogate model in BO frameworks.

An alternative surrogate model is a random forest (RF) (Breimann, 2001; Hutter et al., 2011). An RF predicts the mean and variance values from the empirical mean and variance of each individual tree's predictions. Thereby, the computation complexity for an RF to fitting and prediction only scales to $\mathcal{O}(n \log n)$ and $\mathcal{O}\log n$ respectively. Additionally, the tree structure allows it to easily deal with various data types as well as conditional hyperparameters, which is a common case in complex hyperparameter spaces. Still, RF's empirical mean and variance predictions might cause poor variance estimation when extrapolation is required (Shahriari et al., 2016). Other GP-free methods include Tree Panzer Estimator (TPE) (Bergstra et al., 2013; Tiao et al., 2021), Bayesain neural network (Perrone et al., 2018) or likelihood-free methods (Song et al., 2022).

Partitioning the entire search space into several subregions with trees has been applied to reduce the computational complexity (Wang et al., 2018) or work with heteroscedasticity (Assael et al., 2014; Gramacy et al., 2004) problems. The inborn hierarchy structure of trees makes them especially suitable for selecting the region that satisfies our requirements (Wang et al., 2014). We follow the same idea in BOinG to build a single local model on the most promising subregion that is obtained by the trees in an RF model.

Similar to BOinG, trust region BO (TuRBO) (Eriksson et al., 2019) maintains a local model and expands or shrinks the subregion based on the evaluated result of the suggested point. TuRBO discards all previously evaluated observations if the subregion shrinks to be smaller than a threshold. Thus, TuRBO does not make full use of the previous evaluations and might sample repeatedly in the same region.

While TuRBO only considers BO as a local optimizer, McLeod et al. (2018) rely on BO to find the most promising subregion and only exploits inside that subregion. BOinG considers the exploration and exploitation trade-off at both global and local levels. Hence it can quickly adjust to the local model while avoiding falling into local optimum too early.

**Algorithm 1** Two-stage Bayesian Optimization with BOinG

1: **Input:** a black-box function $f$; search space $\mathcal{X}$; predictive models $\hat{f}_g$ and $\hat{f}_l$ that fit global and local observation distributions respectively; acquisition functions $\alpha_g$ (for global models) and $\alpha_l$ (for local models); evaluation budget $T$
2: **Output:** global minimizer of $f$: $\boldsymbol{x} \in \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$
3: **Initialization:** Initialize data $\mathbf{X}^{(0)} = \{\langle \boldsymbol{x}^{(n)}, f(\boldsymbol{x}^{(n)})\rangle\}_{n=1}^{N_{init}}$ that contains both initial configurations and their evaluations
4: **for** $t = 1, 2, \cdots T$ **do**
5: fit global model $\hat{f}_g^{(t)}$ on $\mathbf{X}^{(t-1)}$
6: select a candidate point $\boldsymbol{x}_g$ with global acquisition function $\alpha_g$: $\boldsymbol{x}_g \in \operatorname{argmax}_{\boldsymbol{x}\in\mathcal{X}} \alpha_g(\boldsymbol{x}; \mathbf{X}^{(t-1)}, \hat{f}_g^{(t)})$
7: extract a subregion $\mathcal{X}_{sub} \subseteq \mathcal{X}$ based on global candidate $\boldsymbol{x}_g$ and model $\hat{f}_g^{(t)}$
8: fit local model $\hat{f}_l$ with the points inside the subregion $\mathbf{X_i}^{(t-1)} \in \mathcal{X}_{sub}$ together with the points outside the subregion $\mathbf{X_o}^{(t-1)} = \mathbf{X}^{(t-1)} \setminus \mathbf{X_i}^{(t-1)}$
9: determine final sampling point based on local acquisition function $\alpha_l$: $\boldsymbol{x}^{(t)} \in \operatorname{argmax}_{\boldsymbol{x}\in\mathcal{X}_{sub}} \alpha_l(\boldsymbol{x}; \mathbf{X_i}^{(t-1)}, \mathbf{X_o}^{(t-1)}, \hat{f}_l^{(t)})$
10: query $y^{(t)} := f(\boldsymbol{x}^{(t)})$
11: update data: $\mathbf{X}^{(t)} \leftarrow \mathbf{X}^{(t-1)} \cup \{\langle \boldsymbol{x}^{(t)}, f(\boldsymbol{x}^{(t)})\rangle\}$
12: **end for**
13: **Return:** $\hat{\boldsymbol{x}} \in \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$

## 3. A General Approach of Two-Stage Bayesian Optimization

Algorithm 1 outlines the general idea of BOinG. In each iteration, we first train a global surrogate model $\hat{f}_g$ (Line 5) on all observations $\mathbf{X}^{(t-1)}$ to estimate the possible regions that are worth being explored. Using an acquisition function $\alpha_g$ on $\hat{f}_g$, we select a promising configuration (Line 6) that will guide the extraction of a subregion (Line 7; see Section 3.1 for details). Then a new local surrogate model $\hat{f}_l$ is fitted to the observations inside the selected subregion $\mathbf{X_i}^{(t-1)}$ (see Section 3.2 for details). Based on $\hat{f}_l$, the maximum of a local acquisition function $\alpha_l$ (Line 9) decides the next configuration to be evaluated on the real cost function $f$ (Line 10).

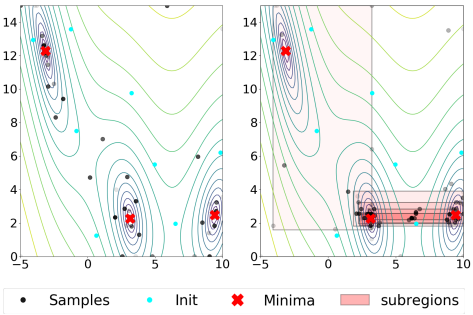


Figure 2: **Left**: The points suggested by a GP that is trained on the entire search space of a Branin function. **Right**: The subregions and points suggested by BOinG over time. Transparency visualizes subregions extracted in different phases.

To be applicable to more complex problems with more reasonable evaluation budgets $T$, the global model needs to scale well to many observations, e.g., an RF is suitable (Hutter et al., 2011). Since the number of points inside the subregion is rather small and nearly constant to some degree, we can afford

the cost of accurate but expensive models (e.g., a
GP) even if the number of the total evaluated points increases by a high degree.

Figure 2 (right) illustrates how the subregion and the suggested points vary during the optimization. As the optimization progresses, the subregion extracted by the global model gradually shrinks toward a potential optimum. BOinG mostly focuses on the regions that look promising—here the lower part of the figure—and invests in more evaluations inside these regions to focus on the exploration of the landscape. Please note that here BOinG hardly explores the region that is shown to be sub-optimal (the upper and right part of the image). It only takes a few evaluations in the beginning to indicate the unfitness of that region. As a comparison, the GP being trained on the entire space invests many evaluations on sub-optimal regions; even worse, the GP suggests lots of points on the boundary; see Figure 2 (left). BOinG hence focuses more on the regions that seem to be more promising. Thus, there is a good chance for our method to find a better solution within a limited budget.

### 3.1. Subregion Extraction with RF

We build the global model on all previous observations and then select the most promising region to be exploited in the next stage. We propose to leverage the scalable and hierarchical structure of a tree-based model, such as an RF, s.t. we can split the global search space into subregions that contain sufficient data points to fit a local model.

First, we do a single BO iteration to determine a global candidate $\boldsymbol{x}_g$ based on a global model $\hat{f}_g$ and acquisition function $\alpha_g$. Then starting from the root node of each tree and a subregion $\mathcal{X}_{sub} = \mathcal{X}$ , we iterate toward the leaf node that contains $\boldsymbol{x}_g$ and get a split of the current node. Each split will shrink $\mathcal{X}_{sub}$ and exclude several points from $\mathbf{X_i}$ (the set of points in the subregion). We repeatedly continue the split until it contains at least $n_{min}$ points. We stop further exploring one node if that split makes the number of the points in the subregion smaller than $n_{min}$. Finally, we stop shrinking the subregion if we cannot step further in any of the trees without keeping the number of the points in the subregion greater than $n_{min}$. For the corresponding algorithm, we refer to the appendix.

One common issue arising from local BO is the size of the subregion. This region should not be too large, otherwise, we would revert to a global optimization again. On the other hand, if the region is too small, the global optimum might be excluded from the subregion and we might only find a local minimum. Furthermore, the size of the subregion needs to be adapted to the number of the previous observations. A local optimizer should ideally contain all the previously evaluated points in the beginning and then gradually shrink to regions that are more likely to contain the global optimum. BOinG achieves this by its user-defined hyperparameter $n_{min}$. If the $n_{min}$ points in the subregion are far from each other, i.e., BOinG still explores a lot and is not certain about a promising region, the selected subregion will be fairly large. In contrast, if the $n_{min}$ points in the subregion are near to each other, i.e., BOinG has already found a promising region, the selected subregion will be fairly small.

### 3.2. Fit Sub-Region with Gaussian Process

The global model suggests a subregion that is worth being further explored. We could then utilize a GP on the points inside the subregion as an accurate model to describe the local

data distribution. However, this could lead to proposed points on the boundaries of the subregion (Oh et al., 2018), since it does not capture the overall trend of the data. So, a compromise between fitting a GP on all observations and on only the observations inside the subregion is required. To this end, we propose a local GP with an augmentation of the global trend that fits all points within the subregion and in addition, efficiently approximates the global data distribution.

We denote the points in the subregion and their function values as $\mathbf{X_i}$ and $\mathbf{f_i}$; here $\mathbf{i}$ refers to "inside subregions". Similarly, we abbreviate the points outside the subregion as $\mathbf{X_o}$ and $\mathbf{f_o}$ respectively). Training a GP with $\mathbf{X_i}$ and $\mathbf{f_i}$ implicitly assumes that the prior distribution of $\mathbf{f_i}$ follows $p(\mathbf{f_i}) = \mathcal{N}(\mathbf{0}, \mathbf{K_{i,i}})$; on the contrary, when a GP model is built to fit all the previous evaluated points, $\mathbf{f_o}$ provides a prior for $\mathbf{f_i}$:

$$p(\mathbf{f_i}|\mathbf{f_o}) = \mathcal{N}(\mu_{\mathbf{f_i}|\mathbf{f_o}}, \sigma^2_{\mathbf{f_i}|\mathbf{f_o}}) \tag{3}$$

$$\mu_{\mathbf{f_i}|\mathbf{f_o}} = \mathbf{k}^T_{\mathbf{o,i}}(\mathbf{K_{o,o}} + \sigma^2\mathbf{I})^{-1}\mathbf{y_o} \tag{4}$$

$$\sigma^2_{\mathbf{f_i}|\mathbf{f_o}} = \mathbf{K_{i,i}} - \mathbf{k}^T_{\mathbf{o,i}}(\mathbf{K_{o,o}} + \sigma^2\mathbf{I})^{-1}\mathbf{k_{o,i}} \tag{5}$$

However, computing the posterior distribution with Equation 4 and 5 for $\mathbf{X_i}$ is expensive and inefficient if we aim for higher evaluation budgets. Luckily, we can make use of the fact that the number of $\mathbf{X_o}$ is much greater than the $\mathbf{X_i}$ and most of $\mathbf{X_o}$ are far away from the subregion and thus have little influence on the subregion. Thus, we approximate Equation 3 with a much cheaper proxy: a Sparse GP (Candela and Rasmussen, 2005).

Sparse GPs introduce a set of latent variables $u$ called **inducing points** so that training points $\mathbf{x}$ and test points $\mathbf{x}_*$ are conditionally independent given $u$. We apply the same idea to approximate the posterior in Equation 4 and 5 and compute the posterior as a prior for $p(\mathbf{f_i}|\mathbf{f_o})$. Since the local GP model is augmented with a globally approximated distribution, our model is dubbed Local GP with Global Augmentation (LGPGA).

We illustrate the difference between the Fully Independent Training Conditional (FITC) (Snelson and Ghahramani, 2006) on all observations and our LGPGA with an approximation of the global trend and exact fit of the points inside the subregion in Figure 3. We note that we cannot simply consider LGPGA as a special case of FITC with $\mathbf{f_i}$ being a subset of $\mathbf{u}$: in FITC, all the inducing points receive the information directly from $\mathbf{f_o}$; while in LGPGA, $\mathbf{f_i}$ only acquire the information from $\mathbf{f_o}$ through $\mathbf{u}$ that allows us to reduce the computation complexity even further (for details, we refer to the appendix).

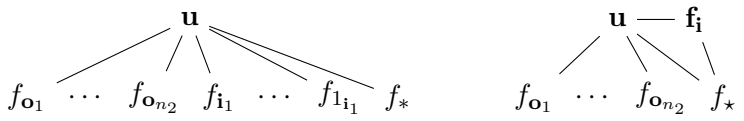


Figure 3: A comparison between FITC (left) and LGPGA (right). In FITC, test latent function values $f_*$ can only obtain the information of $\mathbf{f}$ through the inducing variable $\mathbf{u}$. While in LGPGA, $f_\star$ can directly obtain information from the information from $\mathbf{f_i}$.

The joint distribution of LGPGA is then computed in the following way (with FITC to approximate the global trend):

$$\begin{bmatrix} \mathbf{f_o} \\ \mathbf{f_i} \\ \mathbf{f_\star} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{Q_{o,o}} - \mathrm{diag}[\mathbf{Q_{o,o}} - \mathbf{K_{o,o}}] & \mathbf{Q_{o,i}} & \mathbf{Q_{o,\star}} \\ \mathbf{Q_{i,o}} & \mathbf{K_{i,i}} & \mathbf{K_{i,\star}} \\ \mathbf{Q_{\star,o}} & \mathbf{K_{\star,i}} & \mathbf{K_{\star,\star}} \end{bmatrix}\right) \tag{6}$$

where $\mathbf{Q_{a,b}}$ is defined as $\mathbf{K_{a,u}K_{u,u}^{-1}K_{u,b}}$ (Candela and Rasmussen, 2005).

The posterior $q(\mathbf{f}_\star|\mathbf{f_o}, \mathbf{f_i})$ can be directly computed from Eq. 6 analytically. However, analogous to the hierarchy structure of BOinG, here we compute its posterior in a 2-step manner. First notice that since $\mathbf{f_i}$ and $\mathbf{f}_\star$ have the same format in Eq. 6, we can regard them as one single test input, in which case the whole distribution is equivalent to a standard sparse GP. Hence we can easily compute the joint posterior distribution of $\mathbf{f_i}$ and $\mathbf{f}_\star$ (here we denote them as $\mathbf{f_{i'}}$):

$$q(\mathbf{f_{i'}}|\mathbf{f_o}) = \mathcal{N}(\mu_{\mathbf{i'}}, \sigma^2_{\mathbf{i'}}) \tag{7}$$

$$\mu_{\mathbf{i'}} = \mathbf{K_{i',u}\Sigma K_{u,i}\Lambda^{-1}y_o} \tag{8}$$

$$\sigma^2_{\mathbf{i'}} = \mathbf{K_{i',i'}} - \mathbf{Q_{i',i'}} + \mathbf{K_{i',u}\Sigma K_{u,i'}} \tag{9}$$

$$\mathbf{\Sigma} = (\mathbf{K_{u,u}} + \mathbf{K_{u,o}\Lambda^{-1}K_{g,o}})^{-1} \tag{10}$$

$$\mathbf{\Lambda} = \mathrm{diag}[\mathbf{K_{o,o}} - \mathbf{Q_{o,o}} + \sigma^2_{noise}\mathbf{I}] \tag{11}$$

Now we build another GP model with mean and variance prioritize by Eq. 8 and Eq. 9 respectively. Finally, the posterior of $\mathbf{f}_\star$ is computed by Eq. 1 and Eq. 2. In a nutshell, the first stage aims at approximating the global data distribution $\mathbf{f_o}$ with sparse GPs and provides a prior that warm starts the full GP in the second stage. For detailed information on how an LGPGA is trained and guides the search in subspace, we refer to the appendix.

### 3.3. BOinG+: BOinG with Local TuRBO for Huge Budgets

Since RFs can have poor uncertainty estimates, SMAC (Lindauer et al., 2022) adds additional exploration by randomly sampling a new point with a certain probability and thus can provably converge to the true optimum in the limit (Hutter et al., 2011). In BOinG, we can also combine SMAC's approach with the more explorative strategy of TuRBO: if we cannot make further progress with BOinG, we gradually increase the probability of switching to TuRBO instead of sticking to the subregion proposed by RF. To avoid unnecessary exploration, similar to Wang et al. (2020), we only do a TuRBO search inside a subregion, i.e. we randomly sample several points from $\mathcal{X}$ and extract the subregions $\mathcal{X}_{sub}$ around them with the method proposed in Section 3.1. Since we are interested in stronger exploration, we then select the subregions with the largest volume and thus with the smallest density of observed points. Then we start a new TuRBO run within this subregion whose initial points are the existing points in this subregion. Similarly, if we cannot make further progress with TuRBO, we increase the probability of switching back to BOinG. For the details, we refer to the appendix.

All in all, BOinG (and BOinG+) is introduced to alleviate the weakness of an RF model: RF is a poor extrapolator and might fail to estimate the data distribution correctly in the regions with lower density of evaluated points (Shahriari et al., 2016). Thus we build another GP model in the vicinity of the point suggested by RF with lower evaluated budgets. Because of the poor extrapolation abilities, we extend BOinG by random search with a model-guided optimizer (TuRBO) for larger budgets where we can afford and potentially need better exploration.

## 4. Experiments

**Experimental Setup** As a baseline, we compare BOinG with RF and a GP that is trained on all the previous evaluations. Since TuRBO[1] and LA-MCTS[2] are based on a similar idea of leveraging subregions, we compare BOinG against these, using their original implementations. LA-MCTS provides two ways of doing local optimization: BO and TuRBO; we present both approaches in our experiments, dubbed `LAMCTS-BO` and `LAMCTS-TuRBO`. TuRBO allows for batch BO; however, for a fair comparison, we only evaluate TuRBO-1 and set its batch size to 1. Finally, we consider TPE (Bergstra et al., 2011) as another baseline.

SMAC implemented LogEI to model heavy-tailed cost distributions (Lindauer et al., 2022), our global optimizer and the RF baseline follow the same implementation for HPO problems. However, our local optimizer and the GP baseline will optimize EI instead in each iteration as EI is considered one of the most popular choices of acquisition functions. Further implementation details can be found in the appendix.[3] We empirically set the number of included points of a subregion based on an RF as $n_{min} := 5 \cdot d$, where $d$ is the number of dimensions of the target function.

### 4.1. Synthetic Function

We assessed the algorithms on the following functions: Branin (2D), Ackley (10D) and Levy (10D). The input domain follows the suggestions by Surjanovic and Bingham (2013). All the optimization processes are repeated 30×. These synthetic functions usually do not fit the requirement of "heavy-tailed cost distributions". We simply apply the EI acquisition function instead of logEI on these functions at the global level (the first stage). (However, this setting only works for the experiments in the synthetic functions. For the other experiments in this paper, logEI is still considered as the acquisition function in the first stages)

Figure 4 shows that–as one might expect–GP-BO is a very strong approach for the Branin and Levy function, while on the Ackley function, the GP is not able to capture the high-dimensional, complex landscape well, and is outperformed by other approaches. Among all the compared optimizers, BOinG is the only one showing a robust performance on higher-dimensional problems, being always at least on par with the best optimizer.

### 4.2. Hyperparameter Optimization of ML Algorithms

Although synthetic functions are a nice sanity check, we designed BOinG with HPO in mind and thus evaluated it on several HPO tasks.

We utilize BOinG to optimize the following hyperparameter benchmarks introduced by Falkner et al. (2018), provided by HPObench[4] (Eggensperger et al., 2021): (i) Tuning hyperparameter of a neural network surrogate model (ParamNet) for the Adult dataset (Adult) with eight hyperparameters; (ii) Tuning seven hyperparameters of proximal policy optimization (PPO) (Schulman et al., 2017) to learn the cartpole task. For the details, we refer to the appendix.

1. https://github.com/uber-research/TuRBO
2. https://github.com/facebookresearch/LaMCTS/tree/master/LA-MCTS
3. To ensure reproducibility, our code is publicly available at https://github.com/automl/SMAC3
4. https://github.com/automl/HPOBench/

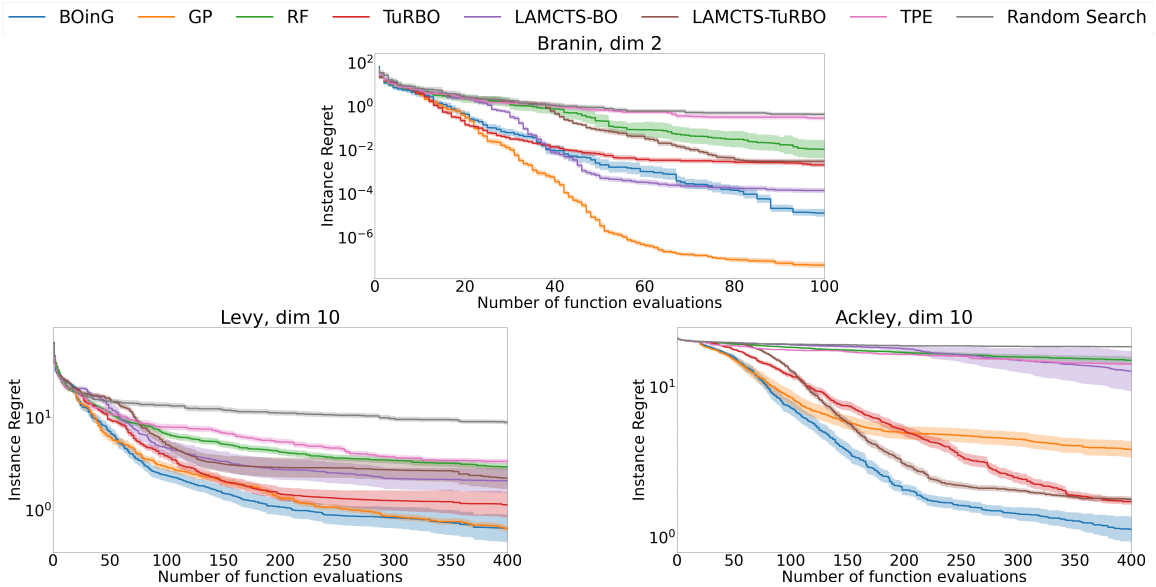


Figure 4: Optimization on different synthetic functions, the solid lines are the mean of the best observed values with semi-transparent areas to indicate standard error.

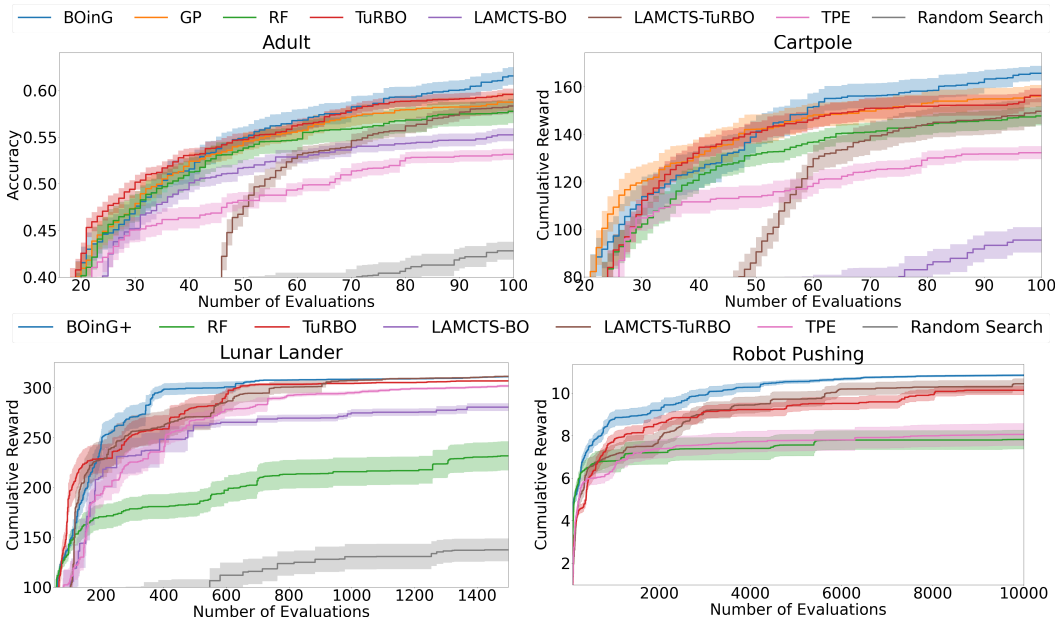


Figure 5: Results on ParamNet (Top Left), CartPole (Top Right), Lunar (Bottom Left) and Robot (Bottom Right). We show the mean performance and the standard error. Larger is better.

Figure 5 (top) shows the result of different optimizers on both target algorithms. BOinG achieves substantially better final performance on both benchmarks. Since BOinG shares the same model as a full GP model at the beginning of the optimization phases ($5 \cdot d$), it

achieves nearly the same performance as a full GP in the first few iterations and is able to identify the most promising region after $\approx 60$ function evaluations.

Additionally, as BOinG introduces a local model to reduce the complexity and BOinG+ solves the poor-extrapolator issue by RF, we could now study how BOinG scales to problems with larger budgets. We optimize 12 hyperparameters of a heuristic controller for a lunar lander implemented in the OpenAI gym (Brockman et al., 2016) and 14 hyperparameters of a controller for the robot pushing problem (Wang et al., 2018).[5] Both tasks are applied by Eriksson et al. (2019) and we set the same search space as TuRBO did.[6] The results are shown in the lower part of Figure 5.

Although TuRBO and LA-MCTS-TuRBO are able to gain some advantage in the beginning, BOinG+ is able to catch up and even outperform TuRBO and LA-MCTS-TuRBO after 200 function evaluations on Lunar Lander. On the robot benchmark, BOinG+, TuRBO and LAMCTS-TuRBO clearly outperform all the other methods, with a small advantage for BOinG+. Overall, the results show the strong performance of BOinG even in the large-budget setting.

## 5. Discussion and Outlook

In this paper, we proposed BOinG, a hierarchical approach that combines the best of a random forest on a global optimization level and a Gaussian Process on a local level. The underlying idea is that BOinG can better focus on promising subregions by having better local models and pinpointing the optimum with fewer function evaluations. Empirical experiments show that BOinG is a very robust approach and is able to outperform vanilla BO and other local BO approaches on HPO problems.

Even though BOinG is a robust approach, i.e. combining the best of GP-BO and RF-BO, GP-BO is often still the most efficient approach in low-dimensional spaces. Furthermore, in higher-dimensional spaces, BO frameworks tend to suggest points near the boundary (Oh et al., 2018) of the configuration space. Such points might not give the RF enough insights about how to extract a subregion, i.e., the subregion might still be too close to the boundary. Last but not least, BOinG's combination of two surrogate models and two acquisition functions might be brittle in some applications; however, in our experiments, BOinG turned out to be surprisingly robust across different tasks.

Although we focused on RFs and GPs as upper and lower-level surrogate models, BOinG can also be seen as a model-agnostic approach, which could adopt other surrogate models, such as TPE. In future work, we plan to study whether we can benefit from a data density estimation approaches such as TPE or non-axis-aligned splits (Wang et al., 2020) for better guidance on the upper level. Another promising approach would be to combine Thompson sampling for BO (Kandasamy et al., 2018) while maintaining multiple subregions (Eriksson et al., 2019) to perform batched BOinG efficiently.

5. This task requires 10000 evaluations that might be too expensive for 'LAMCTS-BO' so we omit 'LAMCTS-BO' here
6. https://github.com/uber-research/TuRBO

## References

J. Assael, Z. Wang, and N. Freitas. Heteroscedastic treed bayesian optimisation. *arXiv:1410.7172 [cs.LG]*, 2014. URL http://arxiv.org/abs/1410.7172.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 13: 281–305, 2012.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Proc. of NeurIPS'11*, pages 2546–2554, 2011.

J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. of ICML'13*, pages 115–123, 2013.

L. Breimann. Random forests. *MLJ*, 45:5–32, 2001.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs.LG]*, 2016.

J. Candela and C. Rasmussen. A unifying view of sparse approximate gaussian process regression. *JMLR*, 6:1939–1959, 2005.

K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NeurIPS Workshop on Bayesian Optimization in Theory and Practice (BayesOpt'13)*, 2013.

K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo. In *Proc. of ICLR'21*, 2021.

D. Eriksson, M. Pearce, J. Gardner, R. Turner, and M. Poloczek. Scalable global optimization via local bayesian optimization. In *Proc. of NeurIPS'19*, pages 5496–5507, 2019.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proc. of ICML'18*, pages 1437–1446, 2018.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. pages 3–38. Springer, 2019. Available for free at http://automl.org/book.

R. Gramacy, H. Lee, and W. Macready. Parameter space exploration with gaussian process trees. In *Proc. of ICML'04*, pages 45–52, 2004.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proc. of UAI'13*, pages 282–290, Arlington, Virginia, USA, 2013.

F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION'11*, pages 507–523, 2011.

F. Hutter, H. Hoos, and K. Leyton-Brown. Parallel algorithm configuration. In *Proc. of LION'12*, pages 55–70, 2012.

D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black box functions. *JGO*, 13:455–492, 1998.

K. Kandasamy, J. Schneider, and B. Póczos. High Dimensional Bayesian Optimisation and Bandits via Additive Models. In *Proc. of ICML'15*, pages 295–304, 2015.

K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *Proc. of AISTATS'18*, pages 133–142, 2018.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast bayesian hyperparameter optimization on large datasets. In *Electronic Journal of Statistics*, volume 11, pages 4945–4968, 2017.

M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *JMLR*, 2022.

M. McLeod, S. J. Roberts, and M. A. Osborne. Optimization, fast and slow: Optimally switching between local and bayesian optimization. In *Proc. of ICML'18*, 2018.

C. Oh, E. Gavves, and M. Welling. BOCK : Bayesian Optimization with Cylindrical Kernels. In *Proc. of ICML'18*, pages 3865–3874, 2018.

V. Perrone, R. Jenatton, M. Seeger, and C. Archambeau. Scalable hyperparameter transfer learning. In *Proc. of NeurIPS'18*, pages 12751–12761, 2018.

C. Pimenta, A. de Sá, G. Ochoa, and G. Pappa. Fitness landscape analysis of automated machine learning search spaces. In *Proceedings of Evolutionary Computation in Combinatorial Optimization*, pages 114–130. Springer, 2020.

Y. Pushak and H. Hoos. Algorithm configuration landscapes: - more benign than expected? In *Proc. of PPSN'18*. Springer, 2018.

H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: Non-conjugate variational inference in gaussian process models. In *Proc. of AISTATS*, 2018.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347 [cs.LG]*, 2017.

B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Proc. of NeurIPS'06*, volume 18, pages 1257–1264. MIT Press, 2006.

J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proc. of NeurIPS'12*, pages 2960–2968, 2012.

Jiaming Song, Lantao Yu, Willie Neiswanger, and Stefano Ermon. A general recipe for likelihood-free Bayesian optimization. In *Proc. of ICML'22*, 2022.

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved January 28, 2021, from http://www.sfu.ca/~ssurjano, 2013.

C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD'13*, pages 847–855, 2013.

Louis C Tiao, Aaron Klein, Matthias W Seeger, Edwin V. Bonilla, Cedric Archambeau, and Fabio Ramos. Bore: Bayesian optimization by density-ratio estimation. In *Proc. of NeurIPS'21*, 2021.

M. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Proc. of AISTATS'09*, 2009.

R. Turner. The bayes opt benchmark documentation. https://bayesmark.readthedocs.io/en/latest/, 2019.

X. Wan, V. Nguyen, H. Ha, B. Ru, C. Lu, and M. A. Osborne. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. In *Proc. of ICML'21*, 2021.

L. Wang, R. Fonseca, and Y. Tian. Learning search space partition for black-box optimization using monte carlo tree search. In *Proc. of NeurIPS'20*, 2020.

Z. Wang, B. Shakibi, L. Jin, and N. Freitas. Bayesian multi-scale optimistic optimization. In *Proc. of AISTATS'14*, 2014.

Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched Large-scale Bayesian Optimization in High-dimensional Spaces. In *Proc. of AISTATS'18*, pages 745–754, 2018.