Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

MULTI AGENT ROLLOUT FOR BAYESIAN OPTIMIZATION

Shyam Sundar Nambiraja[1], and Giulia Pedrielli[1]

[1]School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, USA

ABSTRACT

Solving black-box global optimization problems efficiently across domains remains challenging especially for large scale optimization problems. Bayesian optimization has obtained important success as a black box optimization technique based on surrogates, but it still suffers when applied to large scale heterogeneous landscapes. Recent approaches have proposed non-myopic approximations and partitioning of the input domain into subregions to prioritize regions that capture important areas of the solution space. We propose a Multi Agent Rollout formulation of Bayesian optimization (MAroBO) that partitions the input domain among finite set of agents for distributed sampling. In addition to selecting candidate samples from their respective sub regions, these agents also influence each other in partitioning the sub regions. Consequently, a portion of the function is optimized by these agents which prioritize candidate samples that don't undermine exploration in favor of a single step greedy exploitation. The efficacy of MAroBO is demonstrated on synthetic test functions.

1 INTRODUCTION

Many real world optimization tasks, including but not limited to, material design (Zhang et al. 2020), hyper parameter tuning (Bansal et al. 2022), robots planning (Calandra et al. 2014), drug discovery (Negoescu et al. 2011), neural architecture search (Kandasamy et al. 2018), often rely on noisy, expensive to evaluate black-box functions that do not have a closed form expression or expose gradient information. Bayesian optimization is among a class of sample efficient frameworks for global optimization that dates back to the work by (Jones et al. 1998), where a surrogate model, a Gaussian process in particular, is sequentially updated by adding samples that maximize an acquisition function (expected improvement) in locations of the input space that have not been evaluated. While successful several challenges have been faced applying Bayesian optimization to realistic problems. In fact, among the challenges that require modifications to the original routine, the following are related to the context of the algorithm presented in this paper: (i) *High dimensional input spaces*, are problematic because the cost of training the surrogate kernel increases non linearly with the dimensions. Most kernels used in Bayesian optimization have a number of hyper parameters that grows linearly with the dimensionality of the input, thus making the likelihood optimization an increasingly difficult task. In fact most kernels exhibit challenges for inputs with $d \geq 20$; (ii) *Large number of Samples* arise with large scale search spaces. This challenge may or may not appear with high dimensional inputs. Also in this case, the surrogate is the bottleneck of the algorithm. This is due to the fact that the complexity of training Gaussian Processes grows cubically with the size of the sample.

Demonstrating the relevance of these challenges, important contributions have been made in the literature for scaling Bayesian Optimization to high dimensions. In particular, approaches that deal with high dimensionality tend to focus on proposing novel kernels that allow to reduce the difficulty of optimizing the hyper parameters by introducing structure.

An example of techniques in this area are *Projection based approaches* that perform the search in lower dimensional manifolds, assuming a low effective dimensional space exists that contains all the behavioral information of the original high dimensional function. Contributions in this category include Random Embedding BO (Wang et al. 2016), Hashing enhanced subspace Bayesian Optimization by (Munteanu

et al. 2019). However, many real world problems exist where low dimensional manifold may not be justified. *Statistical learning based approaches* relax the low-dimensional assumption by relying on structural properties of the black-box function. Kandasamy et al. (2015) proposes an additive structure which assumes the function is a sum of small, disjoint groups of dimensions. Success depends on accurately mimicking inter-dimensional dependencies. *Variable selection approaches* Variable selection assumes not all dimensions are important for optimization. SAASBO (Eriksson and Jankowiak 2021) introduces a novel sparsity inducing SAAS prior which prioritizes dimensions by making the coefficients of unimportant variables near to zero. For the second category, the objective is to reduce the cost of inference, which scales cubically with the number of function evaluations. In (Song et al. 2022), a Monte Carlo tree search (MCTS) is proposed to iteratively partition the variable space, thus reducing the sample sizes, while also deeming subsets of dimensions as unimportant relying on the low-dimensional assumption. *Trust region based approaches* (TuRBO) (Eriksson et al. 2019) uses localized modeling, refining the objective function within localized regions called *Trust Regions*. However, TuRBO's independent learning of trust regions without data sharing may lead to inefficiencies. In fact the trust regions do not form a partition of the input. Similarly, SOAR (Mathesen et al. 2021) uses a surrogate model to generate restart locations for local searches, dynamically allocating function evaluations based on observed performance. This adaptive approach ensures efficient resource utilization while achieving optimal solutions. MAMBO proposed by (Wang et al. 2022) utilizes a sub-sampling technique and subspace embedding to reduce computational effort when dealing with large-scale and high-dimensional problems.

Recently, a relevant branch of the literature in Bayesian optimization, has focused on improving the sample selection by inheriting algorithmic structures from Dynamic Programming. In particular, approaches that use lookahead schemes and rollout have been investigated to improve the sample efficiency (thus, reducing the sample size). Examples are the work in (Lee et al. 2020; Jiang et al. 2020). Also, approaches using Multi-Agent frameworks to reduce the size of the sample sets have been proposed (Ma et al. 2023; Krishnamoorthy and Paulson 2023) that use agents as a means to split the sample set thus reducing the dimensionality of the samples for individual searches.

Our work focuses on scaling BO by improving the quality of selection and increasing the efficiency of the approach when large solution spaces are considered. Similar to MAMBO and the work in (Song et al. 2022) we partition the input space to reduce the sample sizes, and like SOAR and TuRBO we perform local searches and update local models to regions of interest. However, such regions are generated rigorously using a Multi-agent scheme where agents are adaptively coordinated, similarly to the more recent literature in lookahead and multi agent approaches.

The remainder of this paper is organized as follows: Section 2 gives a brief overview of the related work in the literature, Section 3 introduces the MAroBO algorithm in detail, Section 4 provides the experimental results of MAroBO on synthetic test functions. Finally, Section 5 concludes the paper.

2 LITERATURE REVIEW

The proposed MAroBO shares aspects with three key classes of algorithms that we discuss herein to highlight the contributions that our methodology brings compared with these approaches. In particular, we discuss (i) partitioning based methods, (ii) lookahead based methods, and (iii) multi-agent approaches.

2.1 Partitioning Based Approaches

Exact optimization methods, like branch and bound, are widely used but assume constant correlation functions within Gaussian processes across the input space. Partitioning the input space into regions with unique correlation functions proves beneficial when data behavior varies across different parts of the input space. LA-MCTS (Limited Area Monte Carlo Tree Search) by (Wang et al. 2020) dynamically explores promising regions using Monte Carlo Tree Search, continuously refining learned boundaries. In contrast, PART-X (Pedrielli et al. 2023) adaptively branches and samples within the input space using local Gaussian

process estimations. LA-MCTS establishes nonlinear boundaries, while PART-X employs axis-aligned branches. While these approaches increase the efficiency of BO when scaled to large solution spaces, the quality of selection is not addressed by these approaches.

2.2 Lookahead Approaches

While most acquisition functions are myopic, considering only the next function evaluation, non-myopic ones consider the impact of multiple future evaluations. (Lee et al. 2020) proposes non-myopic acquisition functions through rollout, which involves simulating multiple steps of BO. These rollout acquisition functions are defined as high-dimensional integrals, making them computationally expensive to compute and optimize. The authors propose methods to significantly reduce the computational burden of computing rollout acquisition functions. (Jiang et al. 2020) proposes a joint Optimization of Decision Variables. Instead of solving nested optimization problems within a multi-step scenario tree, the paper proposes jointly optimizing all decision variables in the full tree simultaneously, termed as a *one-shot* approach. However, these methods often rely on assumptions about the maximizers of second-stage value functions and may involve non-adaptive or computationally expensive procedures. The aforementioned approaches improve the quality of selection but are not computationally efficient. Moreover, propagation of model mis-specification errors undermine the benefits of longer look ahead horizons.

2.3 Multi Agent Approaches

Multi-agent approaches distribute the workload, allowing parallelization and better exploration-exploitation trade-offs. Entropy Search (ES) (Ma et al. 2023) is notable for selecting query points maximizing mutual information about the optimum of the black-box function. However, ES faces computational complexity challenges, addressed by recent gradient-based multi-agent ES algorithms, although they still require heavy computations. A framework by (Krishnamoorthy and Paulson 2023) adds a penalty term to BO acquisition functions to account for agent coupling, solved in parallel using alternating directions method of multipliers (ADMM). Our work refers to the a Multi-Agent Rollout approach as proposed in (Bertsekas 2019) and adapts it to the idea of partitioning and to the problem of BO sampling. The inherent coordinating nature of the agents in this approach enables the agents to act in a coordinated manner while taking independent decisions.

2.4 Contributions

We present an approach that integrates Partition-based methods with a multi-agent methodology. In particular, we propose to integrate a class of Approximate Dynamic Programming technique known as Multi-agent Rollout (Bertsekas 2019) within the Bayesian optimization routine.

The inherent collaborative nature of the agents allows the approach to effectively screen the input space and quickly focus on relevant regions. By assigning regions within the partition to agents, the input space is sequentially split based on the non-myopic actions taken by the agents.

The result of this work is a Multi-agent formulation of Rollout for Bayesian optimization (MAroBO). The key contributions of this paper are as follows:

- Given the high-dimensional nature of many real-world functions, a multi-agent algorithm that focuses on specific regions of the unknown function by partitioning the search space and distributing them among the agents is proposed. These agents provide point estimates from their assigned *active* sub-regions, facilitating better performance compared to vanilla Bayesian optimization.
- The inherent collaborative nature of Multi-agent Rollout allows us to direct the agents towards the optimum. At the same time the explorative nature of BO makes our approach robust to multi modality.

3 MULTI AGENT ROLLOUT FOR BAYESIAN OPTIMIZATION

We introduce the multi-agent rollout (MARO) method, with the notation required by its use in the BO context, thus leading to the MAroBO algorithm in section 3.2. In MARO, an action or control consists of or can be decomposed into multiple components, with a separable control constraint structure, and each component at each stage is chosen by a separate decision entity referred to as "agent" (Bertsekas 2021). In MARO, the action from the original decision problem $\mathbf{x}$ is decomposed across a number of agents. As a result, the evolution of the algorithm together with the cost function are now defined over the cross-product of the agent-level actions. When the problem structure lends itself to the definition of agents, MARO can lead to more efficient solutions compared to traditional Rollout that, as mentioned in the literature review, has already been tested in Bayesian optimization contexts (Bertsekas 2021; Bertsekas 2022a; Bertsekas 2022b).

3.1 Multi Agent Bayesian Optimization Formulation

In the MARO context, the optimization problem we are trying to solve can be formulated as:

$$\min_{\mathbf{x}\in X}\left[\min_{i}\left(f(x_i)\right)\right] \tag{1}$$

Where X is the input space, and the decision variable is encoded as $\mathbf{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$, where the component x_i is an agent feasible solution, and $x_i \in X_i : \cup_i X_i = X, \cap_i X_i = \emptyset$. MAroBO returns as solution the location associated with the agent that achieves the lowest cost. More specifically, we define a decision vector at algorithm iterate k that can be decomposed according to a separable input structure, i.e., $\mathbf{x}_k = \left(\mathbf{x}_k^1, \ldots, \mathbf{x}_k^m\right)$, where $\mathbf{x}_k^i \in X_k^l, l = 1, \ldots, m$ (Bertsekas 2021). Thus the domain of the original problem can be expressed through the product (Bertsekas 2021):

$$X_k = X_k^1 \times \cdots \times X_k^m.$$

MAroBO not only samples points iteratively but also sequentially defines and evaluates *agent configurations* that induce a solution. For a partition with m subregions, i.e., $\cap_i X_k^i = \emptyset, \cup_i X_k^i = X$, each agent is assigned a subregion X_k^i, a data set D_k^i within that subregion, and a surrogate model Y_k^i trained on D_k^i. These elements define the agent configuration, c_k^i, at iteration k, and a configuration c_k is the collection of all agent configurations. At each iteration k, the i-th agent can modify the configuration based on information from agents $j = 1, \ldots, i-1$ and their predictions on the remainder of the agents configurations $j = i+1, \ldots, m$.

Definition 1 (Agent Configuration) An agent configuration is a defined by the tuple (X_k^i, D_k^i, Y_k^i) where X_k^i represents the region assigned to the i^{th} agent, D_k^i denotes the dataset utilized by the i^{th} agent, Y_k^i indicates the surrogate model utilized by the i^{th} agent. Each component of the tuple defines a specific aspect of the i^{th} agent's setup within the overall configuration.

Definition 2 (Configuration) A configuration c_k is defined as the collection of m agents configurations represented as $\left(X_k^i, D_k^i, Y_k^i\right)_{i=1}^m$.

Given a configuration, the MAroBO incumbent solution is:

$$\hat{x}_k^* = \arg\min_i \left(\min_{x^i \in D_k^i} f\left(x^i\right) \right). \tag{2}$$

We distinguish two types of agent. Agents $i = 1, \ldots, m-1$ are *active agents* and they can at each iteration actively change the subregion of interest and sample new locations, agent m is an *inactive agent*. The

inactive agent is responsible to collect all the subregions that the $m-1$ active agents disregard as a result of their update. The inactive agent can hence have associated multiple subregions and while it maintains the locations and the surrogate model no new sampling is performed by this agent. Subregions contained by the inactive agents can receive new locations if an active agent decides to jump into an inactive region.

At iteration k, we define the state of MAroBO S_k as the set of agents and the corresponding tuple $S_k = \left(X_k^i, D_k^i, Y_k^i\right)_{i=1}^m$, considering this state, the algorithm generates number N^C of alternative configurations, i.e., different possible agent assignments for a given state S_k. To do so, we sequentially solve the following equations starting at agent $i = 1$:

$$\tilde{c}_k^i \in \arg\min_{C^i} F^a(\tilde{c}_k^1, \ldots, c_k^i, \ldots, \hat{c}_k^m)\ i = 1, \ldots, m. \tag{3}$$

Where C^i represents the set of feasible configurations for the i-th agent, and F^a is a cost function that the agent uses to sample the configuration component, which is nothing but the minimum encountered function value. In fact, each agent can change its assigned subregion X_k^i, and, given the subregion, it will sample novel locations following an acquisition function implied by a surrogate model built with the locations already assigned to X_k^i that have been sampled from previous iterations. The additional points are added to those currently present within the generated X_k^i. The sample points are then added to the data set D_k^i, and, upon evaluation of the locations, each agent can update a surrogate model Y_k^i. Once updated, different possible agent assignments from the new state yield N^c configurations, which is user defined. Once the configurations are generated, the rollout method is called to evaluate the competing configurations and move MAroBO to the *best* configuration.

At iteration k, the rollout algorithm, is used to evaluate the candidate configurations, namely $c_k^g, g = 1, \ldots, N^c$. To do so it applies a base-heuristic to simulate the evolution of each configuration for an horizon of length h, namely, $(\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_{k-1}, \tilde{\mathbf{c}}_k, \hat{\mathbf{c}}_{k+1}, \ldots, \hat{\mathbf{c}}_{k+h})$.

The configuration that minimizes the approximate Q-factor, here referred to as F^{RO}, is selected. The MAroBO implementation for the function F^{RO} will be detailed in section 3.2.2.

$$\tilde{\mathbf{c}}_k \in \arg\min_{\mathbf{c}_k^g} F^{RO}\left(\tilde{\mathbf{c}}_1^g, \ldots, \tilde{\mathbf{c}}_{k-1}^g, \mathbf{c}_k^g, \hat{\mathbf{c}}_{k+1}^g, \ldots, \hat{\mathbf{c}}_{k+h}^g\right). \tag{4}$$

It then repeats with $(\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_{k-1})$ replaced by $(\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_k)$. After K iterations the rollout algorithm produces the complete sequence of configurations

$$\tilde{\mathbf{c}} = (\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_N)$$

which is called the *rollout solution*. The fundamental result underlying the rollout algorithm is that under certain assumptions, we have *cost improvement*, i.e.,

$$F(\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_N) \leq F(\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), \tag{5}$$

where $(\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N)$ is the configuration produced by the base heuristic and $\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_N$ is the configuration produced by the rollout policy.

MAroBO full step example (Figures 1-2). Figures 1-2 show an example of configuration update. Specifically, Figure 1 shows that at iteration $k = 1$, there are 4 subregions (active agents), each with an associated dataset $D_1^i, i = 1, \ldots, 4$ (dots). Each agent builds its own surrogate $Y_1^i, i = 1, \ldots, 4$ using the respective datasets. Figure 2 shows the configuration obtained after a full MAroBO iteration, for $k = 2$, where we see two subregions in gray that are now associated with the inactive agent. We observe that the iterate has produced a new set of subregions and, as a result of the sampling decision made by the agents, updated data sets $D_2^i, i = 1, \ldots, 4$ and surrogates $Y_2^i, i = 1, \ldots, 4$. The detailed process for the generation of a configuration will be detailed in Section 3.2.2 (Step 1).

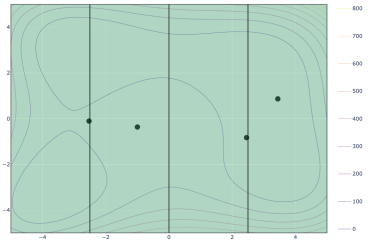


Figure 1: MAroBO iteration $k = 1$, number of active agents $m = 4$, the inactive agent is associated with the empty set.

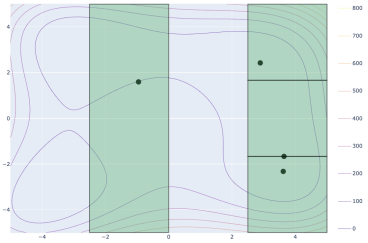


Figure 2: BO iteration $k = 2$, number of active agents $m = 4$, the inactive agent includes the subregions "abandoned" by the active agents.

3.2 Multi Agent Bayesian Optimization Algorithm Overview

In this section, we provide the algorithmic details associated with our MAroBO. We repeat this procedure for K iterations, i.e., until the available function evaluation budget is exhausted. Upon termination MAroBO returns the *rollout solution* as explained in Section 3.1.

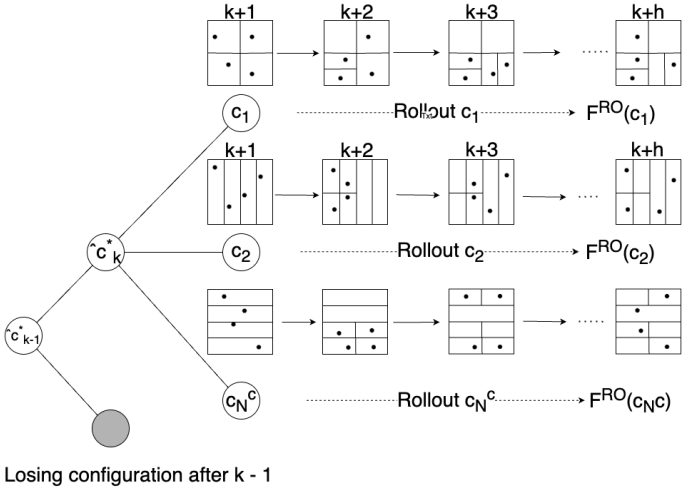


Figure 3: MAroBO progression.

Figure 3 provides an overview of the key components of the algorithm that we further detail below. MAroBO evolves using two key functions (Figure 3): (i) the configuration generator responsible for producing the tuple $c_k = \left(X_k^i, D_k^i, Y_k^i\right)_{i=1}^m$; and (ii) the configuration rollout responsible for the simulation (h-step) and evaluation of the generated configurations at each iteration. While several surrogates and acquisition functions can be used by these two functions, MAroBO adopts a Gaussian process to produce the needed predictions for the function value in each of the subregions (see Section 3.2.1).

3.2.1 Gaussian Process regression

Given n input locations in p-dimensional space $\mathscr{X} \subseteq \mathbb{R}^p$, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, with associated output values $\mathbf{y} = \{f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n)\}$, a Gaussian process model $\mathscr{M}(\mathbf{x}) = \mu + Z(\mathbf{x})$ can be constructed with mean μ, and $Z(\mathbf{x}) \sim \mathscr{GP}(0, \tau^2\mathbf{R})$, where $\mathbf{R}$ is the $n \times n$ variance-covariance matrix whose (i, j)-th entry is the correlation terms built from the distance between pairs of input locations, namely $R_{ij} = r(\mathbf{x}_i, \mathbf{x}_j) = Corr(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j =$

$1,\ldots,n$. In general, the correlation function is parameterized by a vector of hyper parameter θ that requires estimation. The process mean and variance, μ and τ^2 estimates are

$$\hat{\mu} = (\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1})^{-1}\mathbf{1}^T\mathbf{R}^{-1}\mathbf{y} \tag{6}$$

and

$$\hat{\tau}^2 = \frac{1}{n}(\mathbf{y}-\mathbf{1}\hat{\mu})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\hat{\mu}) \tag{7}$$

respectively (Santner et al. 2003), obtained maximizing the log likelihood:

$$L(\mathbf{y}|\mathbf{x},\theta) = -\frac{1}{2}(\mathbf{y}-\mathbf{1}\mu)(\tau^2\mathbf{R})^{-1}(\mathbf{y}-\mathbf{1}\mu) - \frac{1}{2}\log|\tau^2\mathbf{R}| - \frac{n}{2}\log(2\pi). \tag{8}$$

Substituting eqn. (6) and (7) into eqn. (8) we get the concentralized log likelihood formula which only depends on the hyper parameters θ. We then achieve $\mathbf{R}$ through optimization of the concentralized likelihood. The best linear unbiased predictor is given by (Santner et al. 2003):

$$\hat{y}(\mathbf{x}) = \hat{\mu} + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\hat{\mu}),$$

where $\mathbf{r}(\mathbf{x})$ is the $n\times 1$ vector of known correlations of a prediction sample with each sampled location $\mathbf{x}\in\mathscr{X}$:

$$r_i(\mathbf{x}) = Corr(Z(\mathbf{x}),Z(\mathbf{x}_i)), \quad \mathbf{x}_i\in\mathscr{X}$$

and $\mathbf{1}$ the n-vector of ones.

3.2.2 MAroBO Algorithm Steps

Below, we provide the MAroBO algorithmic details.

Step 0. Initialization: Provide the feasible region $\mathbf{X}$, the number of active agents $m-1$, the total budget K; Define the number of configurations per iteration N^c; The rollout simulation horizon h; the inactive agent is assigned the empty set, i.e., $X_0^m \leftarrow \emptyset$. Initialize the active agents considering: (i) An agent is defined by the tuple $\left(X_k^i, D_k^i, Y_k^i\right)$; (ii) Each active agent a_j is assigned to only one sub region X_k^i and each sub region X_k^i contains only one agent a_j.

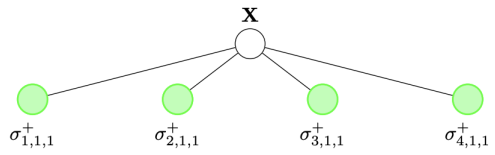
Step 1. Configuration Generation: A configuration is defined as a tuple $\left(X_k^i, D_k^i, Y_k^i\right)_{i=1}^m$. Hence, a configuration update starts from the first agent and updates the tuple by: (1) letting the agent choose whether to stay in its present region or jump into any of the successive agents region; (2) if the agent jumps it will branch the target region according to a partitioning scheme; (3) the agent samples a location in the target region thus updating the data set and the surrogate. We encode such update step for the i-th agent as:

$$X_g^i \leftarrow \quad \mathbb{P}^a\left(\left\{X_k^i, D_k^i, Y_k^i\right\}_{i=1}^m\right) \tag{9}$$

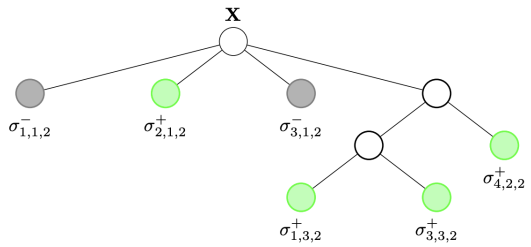
The i-th agent generates the related configuration component fixing the tuples associated to agents $1,\ldots,i-1$, and may change X_k^i: (i) *Jump to a subsequent agent*: agent i has a positive probability to jump into any agent region $l = i+1,\ldots,m-1$ or jumping into the *inactive* region (m); (ii) once a jump has been determined, the agent can probabilistically branch the target subregion. To manage the partitioning of the space, we adopt a tree encoding (see Figure 4). More specifically, the probability to jump to a candidate region σ_{eft}^+ from the agent-associated subregion σ_{eft}^+ is proportional to the maximum Expected Improvement (Jones et al. 1998) calculated in each region as:

$$EI_\cdot^* = \max_{\mathbf{x}\in\sigma_\cdot^\gamma}\mathrm{EI}(\mathbf{x}) = \max\left(\left[f^* - \hat{Y}_\cdot^\gamma(\mathbf{x})\right]\Phi\left(\frac{f^* - \hat{Y}_\cdot^\gamma(\mathbf{x})}{\hat{s}_\cdot^\gamma(\mathbf{x})}\right) + \hat{s}_\cdot^\gamma(\mathbf{x})\,\phi\left(\frac{f^* - \hat{Y}_\cdot^\gamma(\mathbf{x})}{\hat{s}_\cdot^\gamma(\mathbf{x})}\right), 0\right).$$

$$p_\cdot \propto \frac{EI_\cdot^*}{\sum_i EI_i^*}, \tag{10}$$



(a) BO iteration k = 1, number of active agents $m-1=4$.



(b) BO iteration k = 2, number of active agents $m-1=4$, the inactive agent has two subregions.

Figure 4: An example of input space partitioning at iteration $k=1$ and $k=2$ into sub regions σ^{γ}_{eft}. The indices e, f refer to the leaf e located at the f-th level of the tree in the t^{th} iteration. Each leaf is a sub region that is either depicted in green or $\gamma=+$ to denote *active* agents, *inactive* regions depicted in grey or $\gamma=-$ are all assigned to the inactive agent, i.e., X^m_k can be a disconnected set.

where f^* is the best function value sampled so far across the entire input space, and $p_\cdot \propto \frac{EI^*_\cdot}{\sum_i EI^*_i}$ is the jump probability to subregion $(\cdot)$. If an agent jumps to a target region, then it will branch it by choosing uniform at random a dimension and a cutting point. Upon branching the agent makes a sampling decision based on the region-Expected improvement, i.e., the agents adds a location to the selected region according to (Jones et al. 1998):

$$\mathbf{x}^i_g \in \arg\max_{\mathbf{x}\in\sigma^{\gamma}_{eft}} \mathrm{EI}(\mathbf{x}) = \max\left(\left[f^*_i - \hat{Y}^{\gamma}_{eft}(\mathbf{x})\right]\Phi\left(\frac{f^*_i - \hat{Y}^{\gamma}_{eft}(\mathbf{x})}{\hat{s}^{\gamma}_{eft}(\mathbf{x})}\right) + \hat{s}^{\gamma}_{eft}(\mathbf{x})\,\phi\left(\frac{f^*_i - \hat{Y}^{\gamma}_{eft}(\mathbf{x})}{\hat{s}^{\gamma}_{eft}(\mathbf{x})}\right), 0\right). \quad (11)$$

where f^*_i is the best function value sampled so far in subregion σ^{γ}_{eft}. Once the location has been sampled, the true function is evaluated and the following updates are performed by the agent to complete the configuration generation:

$$D^i_g \leftarrow D^i_g \cup \mathbf{x}^i_g : D^i_g = \left\{x \in \cup_l D^l_k : x \in X^i_g\right\}, \quad (12)$$

$$Y^i_g \leftarrow GP|X^i_g, D^i_g. \quad (13)$$

Once all agents have performed the generation step, and N^c configurations have been generated, the rollout is needed to evaluate the configuration. The algorithm proceeds to Step 2.

Step 2. Configurations Rollout: The agent configurations are rolled out for an horizon of length h. As previously mentioned, this means that MAroBO, for each configuration resulting from Step 1, simulates Step 1-Step 2 of the algorithm using a base heuristic in place of the exact evaluations. A graphic representation of this process is depicted in Figure 3 along with the progression of each of the N^c configurations.

At each step of the rollout, the configurations are sequentially updated (i.e., region partitions updated and sample points selected) based on the base heuristic, i.e., the Expected Improvement (Jones et al. 1998) (eqn. (11)). In fact, the base heuristic simulates the evolution of the agents configurations for the rollout horizon according to the following:

$$X^i_{k+r} \leftarrow \mathbb{P}^{RO}\left(\left\{X^i_{k+r-1}, D^i_{k+r-1}, Y^i_{k+r-1}\right\}^m_{i=1}\right), \quad r=1,\ldots,h \quad (14)$$

$$D^i_{k+r} \leftarrow \left\{x : x \in X^i_{k+r}, x \in \cup_l D^l_{k+r-1}\right\} \cup x^i_{k+r-1} : x^i_{k+r-1} \in \arg\max_{x\in X^i_{k+r}\setminus D^i_{k+r}} EI(x), \quad r=1,\ldots,h \quad (15)$$

$$Y^i_{k+r} \leftarrow GP|X^i_{k+r}, D^i_{k+r}, \quad r=1,\ldots,h \quad (16)$$

In eqn. (14), $\mathbb{P}^{RO}$ represents our partitioning scheme. In our implementation, the i-th agent receives as input the subregions from the other agents and follows the two mechanisms we discussed in Step 1 i.e.,

the probability to jump to any agent region $l = i+1,\ldots,m$ and the probability to jump to *inactive* region $(m+1)$ given by eqn. (10).

Example of tree update simulation step. An instance of the partitioning scheme is depicted in Figure 4. Agent $i=1$ assigned to subregion $\sigma_{1,1,2}$, following $\mathbb{P}^{RO}$, it jumps to subregion $\sigma^{+}_{4,1,2}$ thus splitting the subregion into $\sigma_{1,2,2}$ and $\sigma^{+}_{4,2,2}$ with agent and $i=4$. Next, agent $i=2$, stays in its assigned subregion $\sigma_{2,1,2}$. And agent 3 jumps to subregion $\sigma_{1,2,2}$ to split it into $\sigma^{+}_{1,3,2}$ and $\sigma^{+}_{3,3,2}$. Finally, agent 4 stays in its subregion $\sigma^{+}_{4,2,2}$. Consequently, the inactive agent a' *collects* the subregions $\sigma^{-}_{1,1,2}$ and $\sigma^{-}_{3,1,2}$. During each jump, the agent level updates to the dataset and GP follow eqn. (15)-(16). To increase the stability of the approach N^{RO} independent replications of the rollout are performed for each configuration and the approximate Q-factor (F^{RO} in eqn. (4)) is updated as follows:

$$f^{\text{RO}}_{g,\ell} = \min_{i=1,\ldots,m}\left(q^{g}_{1,\ell}\left(\widehat{S}^{1}_{k+h,\ell}\right), q^{g}_{2,\ell}\left(\widehat{S}^{2}_{k+h,\ell}\right),\ldots, q^{g}_{m,\ell}\left(\widehat{S}^{m}_{k+h,\ell}\right)\right)\ g=1,\ldots,N^{c},\ell=1,\ldots,N^{\text{RO}}$$

$$F^{\text{RO}}_{g} = \frac{1}{N^{\text{RO}}}\sum_{\ell} f^{\text{RO}}_{g,\ell}. \tag{17}$$

where $\left(\widehat{S}^{i}_{k+h}\right)$ represents the state of the i-th agent at the end of the rollout horizon and the cost component q^{g}_{i} is nothing but the minimum predicted function value by the i-th agent. Based on the approximate Q-factor, we can choose the configuration c_{k+1} as follows:

$$c_{k+1} \leftarrow \arg\min_{g} F^{RO}_{g}, \tag{18}$$

$$\hat{x}^{*}_{k+1} \leftarrow \arg\min_{x\in\cup_i D^{i}_{K}} \hat{Y}(x). \tag{19}$$

This renders the other configurations in iteration k as *losing* configuration depicted in gray in Figure 3. From the configuration c_{k+1} , the sampling decision $\hat{x}^{*}_{k+1}$ is obtained from all the agents. The locations sampled are a result of predicted values $\hat{Y}(x)$ as opposed to actual function evaluations.

Update the iteration index, $k \leftarrow k+1$.

Step 3. Stopping Condition if $k == N$, STOP, return the best so far $\hat{x}^{*}_{K}\arg\min_{x\in\cup_i D^{i}_{k+1}} f(x)$. Else Go to Step 1.

4 PRELIMINARY ANALYSIS OF MAROBO

We present MAroBO results on synthetic test functions, focusing on the impact of user-defined parameters: rollout horizon, number of agents, and number of configurations on the performance of the algorithm. We evaluate the impact of multi-agent nature coupled with rollout approach across increasing dimensions to assess the rate of performance deterioration. Results are compared against Bayesian optimization.

4.1 Experimental Settings

We consider the difference between the best observed function value and the true optimum value $|\bar{f}^{K} - f^{*}|$ averaged across 25 macro-replications. The mean along with the standard error of the metric are recorded.

We considered the following synthetic test functions:

- Shubert 2-d function has feasible region $[-10,10]^{d}$, has many local minima and one global minima at $(-7.0835, 4.8580)$;
- Rastrigin 10-d function has a feasible region $[-2.5,3]^{d}$, with several local minima and one global minimum at $\mathbf{0}$;

- Schwefel 4-d and 10-d function has a feasible region $[-500, 500]^d$, with several local minima and one global minimum at $\mathbf{0}$;
- Langermann 6-d and 10-d function has a feasible region $[0, 10]^d$ with many local minima and one global minimum with a function value (-5.16)

All the experiments were performed with an initial budget $n_0 = 10d$ where d is the dimension of the test function. These samples were generated using Latin Hypercube Sampling. Across the experiments we modify the number of configurations N^c, the rollout horizon h and the number of agents m. In the following we show a selected fraction of the experiments. All the experiments were executed on Sol - High Performance Computing infrastructure at ASU with 64-core node and 2GB of RAM per core.

4.2 Performance Analysis

In the following, we verify the hypothesis concerning the performance of MAroBO. In fact, we want to verify the following: (H1) Increasing the number of configurations improves the performance of MAroBO; (H2) The number of agents and the rollout horizon length have a positive impact on the performance of MAroBO; (H3) MAroBO is more robust to higher dimensions than BO.

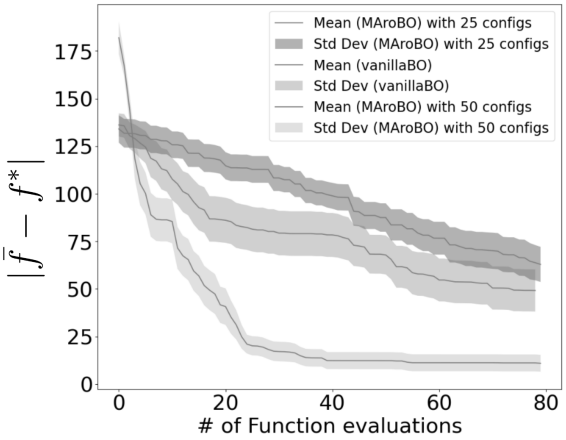


Figure 5: 2-d Shubert with $N^c = 25, N^c = 50$.

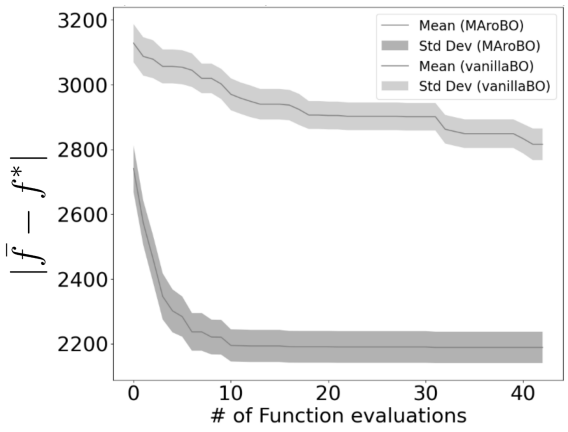


Figure 6: 10-d Schwefel with $m = 4, h = 4$.

Experiment 1: Performance in low dimensional settings for varying configurations. Table 1 shows the results of different 2d functions. We see here the effect of the number of configurations. We expect that the higher the number of configurations evaluated the better the solution. The choice of number of configurations is arbitrary. This is confirmed by the empirical results in all the cases Figure 5 shows how MAroBO Shubert has better performance when the configurations are doubled from 25 to 50.

Table 1: Results for low dimensional experiments. Across the experiments $m = 4$, $h = 4$.

Function	d	N^c	$\lvert\bar{f}^K_{\text{MAroBO}} - f^*\rvert \pm \text{stdErr}$	$\lvert\bar{f}^K_{\text{BO}} - f^*\rvert \pm \text{stdErr}$	K
Shubert	2	25	62.20±6.56	48.50±6.05	78
Shubert	2	50	10.27±4.39	48.50±6.05	78
Langermann	2	25	0.83±0.17	1.62±0.22	78
Langermann	2	50	0.03±0.09	1.62±0.22	78

Experiment 2: Performance for varying number of agents and higher dimensions. MAroBO was run with $N^c = 100$ configurations across all the cases and we notice that it is always superior to BO (Table 2). We can also observe how doubling the number of agents does not seem to have a remarkable effect. We also observed an important variability in the quality of the solutions which may be brought back to the

low number of rollout replications N^{RO}, which we set to 10 across all the experiments. Observing the Rastrigin results, we notice how increasing the rollout horizon has a positive impact, but this finding is not confirmed under the Schwefel which calls for a further experimental fraction and a deeper understanding of the impact of the interaction between the agents and the horizon.

Table 2: Experiments to assess the impact of more agents and longer lookahead horizon. All the functions were evaluated with $N^c = 100$.

Function	d	m	h	$\lvert\bar{f}^K_{\text{MAroBO}} - f^*\rvert \pm \text{stdErr}$	$\lvert\bar{f}^K_{\text{BO}} - f^*\rvert \pm \text{stdErr}$	K
Langermann	6	4	4	4.09±0.03	4.50±0.05	100
Langermann	6	8	4	4.02±0.001	4.50±0.05	100
Rastrigin	6	4	2	2.34±0.24	2.42±0.25	80
Rastrigin	6	4	6	2.04±0.20	2.42±0.25	80
Schwefel	4	8	2	464.15±24.15	518.85±36.22	100
Schwefel	4	8	6	516.13±27.76	518.85±36.22	100

Experiment 3: Effect of increased dimensionality. Table 3 reports the results for the higher dimensional cases. First off we observe the detrimental effect that the increased dimensionality has on the performance of the algorithms. In fact, while MAroBO was not designed for high dimensional inputs, we observed that the algorithm is relatively more explorative than BO by distributing the sampling among agents. Figure 6 shows an exemplar behavior where BO has a very hard time initially to identify improving areas and MAroBO with its agents succeeds in moving towards improved values.

Table 3: Experiments on higher dimensions. Langermann used $N^c = 500$, while Rastrigin and Schwefel were evaluated under $N^C = 600$.

Function	d	m	h	$\lvert\bar{f}^K_{\text{MAroBO}} - f^*\rvert \pm \text{stdErr}$	$\lvert\bar{f}^K_{\text{BO}} - f^*\rvert \pm \text{stdErr}$	K
Langermann	10	4	4	4.95±0.0002	4.99±0.0002	47
Langermann	10	10	4	4.64±0.01	4.99±0.0002	47
Rastrigin	10	4	4	1.31±0.09	3.39±0.31	53
Schwefel	10	4	4	2188.67±48.44	2816.05±42.49	42

5 CONCLUSIONS AND FUTURE DIRECTIONS

We present, for the first time, MAroBO a Multi-Agent Implementation of non-myopic Bayesian optimization. Agents decompose the input space in a partition and sequentially decide whether to enter and partition another subregion or continue investigating their current region. Given an input space, an algorithm for coordination and a heuristic for multi-step simulation, MAroBO allocates agents to different areas, distributing modeling and sampling efforts. The algorithm relies on a collection of Gaussian processes, each defined over a single subregion, estimate the unknown objective function in unsampled areas. Upon communication, the agents simulate their actions without evaluation for a specific horizon, update the partition, and make sampling decisions until the budget is exhausted. Our results show the impact of the number of solutions considered simultaneously by the agents, simulation horizon length, and number of agents. Preliminary results indicate that a version of MAroBO always outperforms BO, granting further analysis. Future work will focus on increasing agent exploration and improving computational efficiency.

REFERENCES

Bansal, A., D. Stoll, M. Janowski, A. Zela and F. Hutter. 2022. "JAHS-Bench-201: A Foundation For Research On Joint Architecture And Hyperparameter Search". In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Bertsekas, D. 2019. "Multiagent rollout algorithms and reinforcement learning". *arXiv preprint arXiv:1910.00120*.

Bertsekas, D. 2021. *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific.

Bertsekas, D. 2022a. *Lessons from AlphaZero for optimal, model predictive, and adaptive control*. Athena Scientific.

Bertsekas, D. 2022b. "Newton's method for reinforcement learning and model predictive control". *Results in Control and Optimization* 7:100121.

Calandra, R., N. Gopalan, A. Seyfarth, J. Peters and M. P. Deisenroth. 2014. "Bayesian gait optimization for bipedal locomotion". In *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers 8*, 274–290. Springer.

Eriksson, D. and M. Jankowiak. 2021. "High-dimensional Bayesian optimization with sparse axis-aligned subspaces". In *Uncertainty in Artificial Intelligence*, 493–503. PMLR.

Eriksson, D., M. Pearce, J. Gardner, R. D. Turner and M. Poloczek. 2019. "Scalable global optimization via local Bayesian optimization". *Advances in neural information processing systems* 32.

Jiang, S., D. Jiang, M. Balandat, B. Karrer, J. Gardner and R. Garnett. 2020. "Efficient nonmyopic bayesian optimization via one-shot multi-step trees". *Advances in Neural Information Processing Systems* 33:18039–18049.

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.

Kandasamy, K., W. Neiswanger, J. Schneider, B. Poczos and E. P. Xing. 2018. "Neural architecture search with bayesian optimisation and optimal transport". *Advances in neural information processing systems* 31.

Kandasamy, K., J. Schneider, and B. Póczos. 2015. "High dimensional Bayesian optimisation and bandits via additive models". In *International conference on machine learning*, 295–304. PMLR.

Krishnamoorthy, D. and J. A. Paulson. 2023. "Multi-agent black-box optimization using a Bayesian approach to alternating direction method of multipliers". *IFAC-PapersOnLine* 56(2):2232–2237.

Lee, E., D. Eriksson, D. Bindel, B. Cheng and M. Mccourt. 2020. "Efficient rollout strategies for Bayesian optimization". In *Conference on Uncertainty in Artificial Intelligence*, 260–269. PMLR.

Ma, H., T. Zhang, Y. Wu, F. P. Calmon and N. Li. 2023. "Gaussian max-value entropy search for multi-agent Bayesian optimization". In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10028–10035. IEEE.

Mathesen, L., G. Pedrielli, S. H. Ng, and Z. B. Zabinsky. 2021, jan. "Stochastic optimization with adaptive restart: a framework for integrated local and global learning". *J. of Global Optimization* 79(1):87–110 https://doi.org/10.1007/s10898-020-00937-5.

Munteanu, A., A. Nayebi, and M. Poloczek. 2019. "A Framework for Bayesian Optimization in Embedded Subspaces". In *Proceedings of the 36th International Conference on Machine Learning, (ICML)*. Accepted for publication. The code is available at https://github.com/aminnayebi/HesBO.

Negoescu, D. M., P. I. Frazier, and W. B. Powell. 2011. "The knowledge-gradient algorithm for sequencing experiments in drug discovery". *INFORMS Journal on Computing* 23(3):346–363.

Pedrielli, G., T. Khandait, Y. Cao, Q. Thibeault, H. Huang, M. Castillo-Effen *et al*. 2023. "Part-x: A family of stochastic algorithms for search-based test generation with probabilistic guarantees". *IEEE Transactions on Automation Science and Engineering*.

Santner, T. J., B. J. Williams, W. I. Notz, and B. J. Williams. 2003. *The design and analysis of computer experiments*, Volume 1. Springer.

Song, L., K. Xue, X. Huang, and C. Qian. 2022. "Monte carlo tree search based variable selection for high dimensional bayesian optimization". *Advances in Neural Information Processing Systems* 35:28488–28501.

Wang, H., E. Zhang, S. H. Ng, and G. Pedrielli. 2022. "A model aggregation approach for high-dimensional large-scale optimization". *arXiv preprint arXiv:2205.07525*.

Wang, L., R. Fonseca, and Y. Tian. 2020. "Learning search space partition for black-box optimization using monte carlo tree search". *Advances in Neural Information Processing Systems* 33:19511–19522.

Wang, Z., F. Hutter, M. Zoghi, D. Matheson and N. De Feitas. 2016. "Bayesian optimization in a billion dimensions via random embeddings". *Journal of Artificial Intelligence Research* 55:361–387.

Zhang, Y., D. W. Apley, and W. Chen. 2020. "Bayesian optimization for materials design with mixed quantitative and qualitative variables". *Scientific reports* 10(1):4924.

AUTHOR BIOGRAPHIES

SHYAM SUNDAR NAMBIRAJA is a Master student in the School of Computing and Augmented Intelligence at Arizona State University. His research interest is primarily in Bayesian optimization. His email address is snambira@asu.edu.

GIULIA PEDRIELLI is Associate Professor in the School of Computing and Augmented Intelligence at Arizona State University. She is interested in the area of stochastics and simulation based optimization, and deals with applications in biomanufacturing, power systems, supply chains, safety critical systems. Her email address is giulia.pedrielli@asu.edu.