Bayesian Optimization approaches for massively multi-modal problems

Ibai Roman[1], Alexander Mendiburu[1], Roberto Santana[1], and Jose A. Lozano[1,2]

[1]University of the Basque Country (UPV/EHU), San Sebastian, Spain, {ibai.roman, alexander.mendiburu, roberto.santana, ja.lozano}@ehu.eus
[2]Basque Center for Applied Mathematics (BCAM), Bilbao, Spain

Abstract

The optimization of massively multi-modal functions is a challenging task, particularly for problems where the search space can lead the optimization process to local optima. While evolutionary algorithms have been extensively investigated for these optimization problems, Bayesian Optimization algorithms have not been explored to the same extent. In this paper, we study the behavior of Bayesian Optimization as part of a hybrid approach for solving several massively multi-modal functions. We use well-known benchmarks and metrics to evaluate how different variants of Bayesian Optimization deal with multi-modality.

Keywords— Bayesian Optimization, Multi-modal Optimization, Gaussian Processes

1 Introduction

Massively multi-modal functions are characterized by having many optimal solutions, where some of them are local and some others are global. There may be numerous or no local optima, and only one global optimum or many global optima. Global optimization of this kind of functions is a difficult task since the optimization algorithm may get trapped in local optima, and due to their complexity, a high number of evaluations is expected. In addition, apart from finding the global optimum, a good covering of all the best solutions is required in many real-world problems [24].

This kind of functions have been extensively studied in Evolutionary Algorithms (EAs) [15]. Traditional methods to obtain a good optima covering in EAs include niching strategies [4, 15, 19]. In standard Genetic Algorithms (GAs), where the crossover operator produces an exploitation oriented behavior, niching methods try to reduce this effect by allowing further exploration.

On the other hand, model-based optimization algorithms take advantage of identifying the landscape of the problem to improve the search. They use all the information gathered from sampling the objective function by means of a Surrogate Model (SM). This type of algorithms have been successfully applied to low-budget optimization tasks [8, 26]. One of the most successful model-based optimization methods is Bayesian Optimization (BO) [1, 11]. BO is a sequential optimization algorithm, where the sampling strategy is based on a probability distribution over all the possible objective functions. This probability distribution acts as a SM, and it is updated every time a solution is evaluated using the actual objective function.

Some of the work done in BO is closely related to the niching strategies proposed for EAs. There are some variants of BO algorithms that use sampling strategies that propose a batch of solutions to perform several evaluations at the same time [3]. Among them, the *Bayesian Optimization via Local Penalization* algorithm proposed in [5] avoids the concentration of samples, similar to sharing [4] and clearing [12] niching strategies investigated in EA literature. On the other hand, the clustering BO approaches [6, 23], where the data is divided into clusters to reduce the complexity of the models, could benefit the discovery of new optima, resembling the clustering [25] techniques proposed for EA. Beyond the mentioned EC and BO approaches, methods that divide the search space have also been proposed in other fields [7, 9].

In this paper, we face massively multi-modal problems bridging the work done in EA and BO literatures. We present preliminary results in this direction, proposing three algorithms that take BO beyond low-budget optimization: 1) Sequential BO with modeling of local optima. 2) Adaptive BO with clustering. 3) BO with batch sampling strategies. Furthermore, we evaluate their behavior using the benchmark presented in [10], and compare our results with those achieved by state-of-the-art algorithms in Evolutionary Computation (EC).

Our research aims to address a number of general questions that are significant for massively multi-modal problems. Among these questions are: Is it possible to get a good covering of multiple optima using a low-budget of function evaluations? Are model-based approaches, specifically BO, an efficient way to obtain a good coverage? Can we take advantage of the work done in EC to design more efficient BO techniques to improve the coverage?

The remainder of the paper is structured as follows: In Section 2, a brief introduction to BO and batch methods is presented. A reduced number of works significantly related to our approach are revised in Section 3. In Section 4, the different BO variants proposed in this paper are introduced. Section 5 presents the experimental benchmark and the numerical results of the experiments. Finally, Section 6 presents the conclusions of the paper.

2 Brief introduction to Bayesian Optimization

BO [11] is a state-of-the-art global optimization technique suitable for low-budget optimization problems. In BO, the samples of the objective function

are sequentially taken based on a sampling strategy. This next sample will be selected by optimizing an acquisition function ($u(\cdot)$) that provides a measure of the utility of each solution. A probability distribution over all the possible objective functions is used to determine this acquisition function, acting as a SM. As the analytic form of the objective function is generally unknown, BO treats the objective function like a random function, and places a prior belief about the space of possible objective functions. Every time the objective function is evaluated, this prior belief is updated with the likelihood of having those observations, generating a posterior distribution over functions.

Algorithm 1 illustrates the whole process of the BO algorithm [1].

Algorithm 1 BO algorithm

procedure BAYESIAN-OPTIMIZATION
  $y_1 = f(\mathbf{x}_1)$ ▷ Sample and evaluate a random point
  $D_{1:1} = \{(\mathbf{x}_1, y_1)\}$ ▷ Initialize the dataset
  $SM \leftarrow D_{1:1}$ ▷ Initialize the SM
  t = 2
  repeat
    $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathbb{R}^d} \; u(\mathbf{x}|SM)$ ▷ Select the next point to evaluate
    $y_t = f(\mathbf{x}_t)$ ▷ Evaluate the objective function
    $D_{1:t} = D_{1:t-1} \cup \{(\mathbf{x}_t, y_t)\}$ ▷ Update the dataset
    $SM \leftarrow D_{1:t}$ ▷ Update the SM
    $t = t + 1$
  until the stopping criterion is met
end procedure

The optimization process begins by sampling a point randomly, and evaluating it using the objective function. Typically, the SM is initialized at this point. Then, until the stopping criterion is met, the next point to evaluate is selected and evaluated, augmenting the dataset and updating the SM. At each step, except the first one, the point that maximizes the acquisition function is selected to be sampled.

2.1 Gaussian Processes

SMs are a key element in Algorithm 1, as the acquisition function will rely on them to select the next point. One of the most popular choices in BO is to use a GP as SM. A GP is a stochastic process, defined by a collection of random variables, any finite number of which has a multivariate Gaussian distribution [17]. What makes GPs interesting for BO is that the posterior distribution of a GP given some observations of the objective function is also a GP.

GPs can be completely defined by a mean function ($m(\mathbf{x})$) and a covariance function, which depends on a kernel ($k(\mathbf{x}, \mathbf{x}')$). Given that, the GP can be expressed as follows:

$$f(x) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1}$$

Usually, a non-informative mean function is used, such as $m(\mathbf{x}) = 0$. However, in more sophisticated approaches, this function will depend on the data.

The kernel establishes the covariance between the objective function values of two different points. Although many kernels have been proposed in the literature, we use the Matern32 kernel, due to the performance shown in previous studies [18]. This kernel is expressed as follows:

$$\begin{aligned} k_{M32}(\mathbf{x}, \mathbf{x}') &= \theta_0^2 \left(1 + \sqrt{3}\sqrt{r^2_{x,x'}}\right) exp\left(-\sqrt{3}\sqrt{r^2_{x,x'}}\right) + \theta_n \\ \text{where } r^2(\mathbf{x}, \mathbf{x}') &= \sum_{d=1}^{n_d} \left(\frac{x_d - x'_d}{\theta_d}\right)^2 \end{aligned} \tag{2}$$

where θ_d, θ_0 and θ_n are the length-scale, amplitude and noise parameters respectively. While the length-scale parameter expresses the relevance of each dimension d, the amplitude parameter scales all dimensions. The noise parameter allows the GP to adapt the random function when the observations of the objective function are noisy.

2.2 Acquisition function

Once we have a way to approximate the value of the objective function through the SM, the acquisition function is placed to select the next point. It assigns a measure of utility for each point in the search space given the SM. This measure of utility balances the exploration versus exploitation trade-off.

In this paper we use GP-UCB [22] as acquisition function. This upper-confidence based algorithm is a combined strategy that balances the reduction of the uncertainty over the objective function with maximizing the expected reward:

$$GP - UCB(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\beta_t}\ \sigma(\mathbf{x}) \tag{3}$$

where β_t is a constant specified depending on the context.

In the previous equation $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ represent the expected value for the objective function at point $\mathbf{x}$ and its variance. At step $t+1$, these values are given by the following equations:

$$\begin{aligned} \mu(\mathbf{x}) &= m(\mathbf{x}) + k(\mathbf{X}_{1:t}, \mathbf{x}) k(\mathbf{X}_{1:t}, \mathbf{X}_{1:t})^{-1} f(\mathbf{X}_{1:t}) \\ \sigma^2(\mathbf{x}) &= k(\mathbf{x}, \mathbf{x}) - k(\mathbf{X}_{1:t}, \mathbf{x}) k(\mathbf{X}_{1:t}, \mathbf{X}_{1:t})^{-1} k(\mathbf{x}, \mathbf{X}_{1:t}) \end{aligned} \tag{4}$$

where $k(\mathbf{X}_{1:t}, \mathbf{x})$ and $k(\mathbf{x}, \mathbf{X}_{1:t})$ are vectors with the value of the kernel function between $\mathbf{x}$ and all the previously evaluated solutions. $k(\mathbf{X}_{1:t}, \mathbf{X}_{1:t})$ is the matrix of the kernel values between all the pairs of the previously evaluated solutions.

3 Related work

Our research is closely related to previous work in a number of areas. In EAs, research on multi-modal problems have been addressed mainly using niching

methods [19, 21]. However, most of these methods do not explicitly construct a model to guide the search. Among model based methods, some of the BO strategies can resemble the work done in EAs. Although they were not specifically designed to solve multi-modal problems, these BO methods can benefit from using a model.

In batch BO sampling strategies, instead of proposing one solution at each iteration, B solutions are simultaneously evaluated. Among these approaches, the *Batch Bayesian Optimization via Local Penalization* approach [5] tries to avoid the surrounding area of the already selected points applying a penalization to the utility of these selected solutions. This property can prevent the algorithm from getting stuck on local optima.

The clustering BO approaches [6, 23] try to the reduce the complexity of the model by dividing the data in several clusters and using several models to create the acquisition function. Our intuition is that BO could take advantage of dividing the data into niches to encourage exploration.

Sharing [4] is one of the first attempts in niching methods. It consists in modifying the fitness value of each individual. When several individuals are occupying a niche, their fitness value is shared. Two individuals belong to the same niche when they are closer than a niche radius. The computational cost and the difficulty to set a good niche radius are the main downsides of this method. Besides, in clearing techniques [12], the neighboring solutions of the k best ones are "cleared", i.e., their fitness value is set to zero. Although it is closely related to sharing, it reduces the computational cost by only measuring the distances between the best solutions and the rest. On the contrary, it requires one extra parameter to adjust.

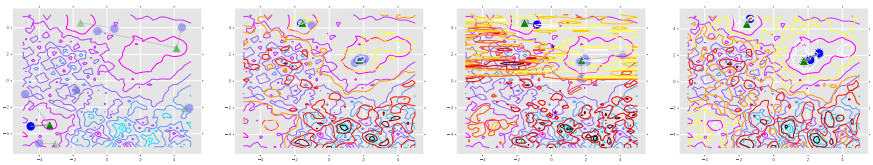
Other niching techniques use a clustering algorithm to divide the population into niches [25]. The fitness of each individual depends on the number of solutions in the niche and the distance to their centroids. This approach also reduces the distance computations required in sharing and encourages the individuals to leave the niche and explore new regions.

In research conducted on multi-modal functions in EAs, some of the state of art results are reported for Niching Evolutionary Algorithm 2 (NEA2) [14] and Niching migratory multi-swarm optimizer (NMMSO) [2]. NEA2 is a combination of clustering and local optimization techniques. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is used to find the local optima while Nearest-better Clustering allows a good covering of the optima. On the other hand, NMMSO uses concurrent swarms. Each swarm exploits its local mode and the algorithm splits or mergers swarms depending on characteristics of the optimization problem. While NEA2 incorporates a covariance matrix model as part of CMA-ES, these two algorithms do not learn a SM in order to predict the fitness value of a solution as BO does.

4 Dealing with multi-modal problems with BO optimization

In the multi-modal optimization problems we address in this work, the number of evaluations that might be needed to cover the different optima is much higher than in low-budget optimization problems. Being BO an optimization technique well suited for low-budget optimization problems, devising efficient ways to learn a GP model with a large dataset is not straightforward. The reason is that the covariance matrix grows according to the number of evaluations. The inverse or the Cholesky decomposition of this matrix is required to compute the posterior distribution, and this computational cost may be prohibitive for massively multi-modal problems if the usual approach were used to learn the model.

In BO literature, some variants that attempt to reduce the computational cost of computing a model when the number of points is high have been proposed. These variants use various mechanisms, such as, "forgetting" older or less informative points, to handle a large number of points. Works that propose sparse GP approaches include the Subset of Data (SoD) approximations [16] and the Subset of Regressors (SoR) [20]. These methods have their own drawbacks, e.g., SoR is a very sophisticated and complex approach, that make them not directly applicable to the type of multi-modal problems we address.



(a) Multi-start (b) Sequential BO (c) Clustering BO (d) Batch BO

Figure 1: BO approaches. The objective function is represented with a contour plot. Each figure describes the behavior of the proposed algorithms in a certain step. The yellow and red surface and contour plot refer to the acquisition function. The current step is represented with a triangle and a circle, where the circle represents the solution selected by the model, and the triangle represents the result of the local search starting from the selected solution. Previous steps are faded. Note that in batch_ls several solutions are selected and they are not faded. Finally, in the last figure, the centroids of the clusters are represented with big purple starts.

Our proposal consists of combining BO with local search techniques. This way we can take full profit of the budget of available evaluations and guarantee that those solutions modeled by the covariance matrix are of better quality. In spite of including the local search, still a large dataset is needed to model the landscape of the objective function. Thus, we propose the following three algorithms that are able to deal with these datasets: The first algorithm is inspired in the SoD approach. The second algorithm avoids the unmanageable

growth of the covariance matrix by splitting the current set of solutions into different clusters. Finally, the third method we propose is similar to the first but it is conceived for situations where the parallel evaluation of several points is desired. Their behavior is illustrated in Figure 1.

4.1 Sequential BO with modeling of local optima

In this approach, the "forgetting" SoD approach is combined with a local search algorithm (see Algorithm 2). Two major changes have been applied compared to Algorithm 1. First of all, a local search algorithm is used to exploit each solution proposed by the BO. Then, the SM is updated with the point suggested by the BO and the best solution achieved by the local search, provided that the local search is able to improve the solution proposed by the BO. The second major change is that the size of the dataset has been limited. When the dataset outsizes the maximum number of points (N), the oldest point is discarded, i.e. the most recently proposed points are those used for modeling. This approach might be the simplest solution when dealing with sparse GP models, but also is the computationally cheapest.

Algorithm 2 Sequential BO with modeling of local optima

1: **procedure** BO-LS(N)
2: $y_1 = f(\mathbf{x}_1)$
3: $D_{1:1} = \{(\mathbf{x}_1, y_1)\}$
4: $SM \leftarrow D_{1:1}$
5: t = 2
6: **repeat**
7: $\mathbf{x}_{bo} = \arg\max_{\mathbf{x}\in\mathbb{R}^d} \; u(\mathbf{x}|SM)$
8: $y_{bo} = f(\mathbf{x}_{bo})$
9: $(\mathbf{x}_{ls}, y_{ls}) = ls(f, \mathbf{x}_{bo})$ ▷ Run the local search
10: $D_{1:t+1} = D_{1:t-1} \cup \{(\mathbf{x}_{bo}, y_{bo})\} \cup \{(\mathbf{x}_{ls}, y_{ls})\}$
11: $SM \leftarrow D_{t-N-1:t+1}$ ▷ "Forget" old data
12: $t = t + 2$
13: **until** the stopping criterion is met
14: **end procedure**

4.2 Adaptive BO with clustering

Our second approach is based on the previous one, but it aims to take more advantage of the knowledge acquired during the optimization process. Inspired in clustering niching techniques and the work done in [6] and [23], this approach divides the dataset into several subsets using clustering techniques. This way, to obtain the conditional distribution of the GP given some solution, only the closest solutions will be taken into account, reducing the size of the covariance matrix. It can be seen as a single SM that is composed by several GPs, each one with an exclusive subset of the dataset.

This algorithm differs from Algorithm 2 in the way the SM is updated and evaluated. Initially, only one subset of the dataset is used, and the posterior of the GP is calculated in the original manner. Every time a data point is sampled, it is added to the closest cluster. As can be seen in Algorithm 3, this updating process consists on updating the centroid and the GPs. When the maximum size of a cluster is reached, the dataset is split into two subsets using a clustering technique, and their centroids are calculated. Two new GPs are generated and updated, one for each subset, and the old cluster is removed. Note that this sparse GP technique can be incrementally applied in the BO process, as only one GP is updated at each step.

Algorithm 3 Update *ClusterSM*

procedure UPDATE-CLUSTERS($\mathbf{x}_t, y_t, \{\mathbf{c}\}, \{D\}, \{SM\}$)
  $\mathbf{c}_{closest}, D_{closest}, SM_{closest} = closest(\mathbf{x}_1, \{\mathbf{c}\})$
  $D_{closest} = D_{closest} \cup \{(\mathbf{x}_t, y_t)\}$
  if $size(D_{closest}) < N$ **then**
    $\mathbf{c}_{closest} = \text{centroid}(D_{closest})$
    $SM_{closest} \leftarrow D_{closest}$
  else
    $\{D_k\}_{k=1}^{K} = split\_dataset(D_{closest}, K)$
    for $k = 1$ to K **do**
      $\mathbf{c}_k = \text{centroid}(D_k)$
      $SM_k \leftarrow D_k$
    end for
    $\{\mathbf{c}\} = \{\mathbf{c}\} + \{\mathbf{c}_k\}_{k=1}^{K} - \{\mathbf{c}_{closest}\}$
    $\{D\} = \{D\} + \{D_k\}_{k=1}^{K} - \{D_{closest}\}$
    $\{SM\} = \{SM\} + \{SM_k\}_{k=1}^{K} - \{SM_{closest}\}$
  end if
end procedure

To evaluate the posterior distribution of this system given a certain solution, the distance to each centroid must be calculated. The SM associated to the closest data-subset is used to predict the outcome of the fitness.

4.3 BO with batch sampling strategies

In order to expand the scope of the analysis of BO techniques for multi-modal problems, we also propose a modified version of Batch BO via Local Penalization [5]. This technique iteratively selects a batch of solutions by optimizing an acquisition function that penalizes the previous selections in the batch. Similar to the work that has been done in EAs to avoid the exploitation oriented nature of the selection operator, this penalization step might be suitable for multi-modal optimization problems, when parallel evaluations can be implemented.

The algorithm works as follows: First, a solution is selected according to a traditional acquisition function (we use UCB in this paper as suggested by the

results of the original work). Then, according to Algorithm 4, a batch of solutions is obtained iteratively, selecting a solution and penalizing the acquisition function according to this selection.

Algorithm 4 Batch Selection via Local Penalization

procedure SELECT-BATCH-LP(t, K, SM)
  $\tilde{u}_{t,0}(\mathbf{x}) = u(\mathbf{x}|SM)$
  $\hat{M} = min_i\{y_i\}$
  $\hat{L} = max_{\mathbf{x}\in\mathbb{R}^d}||\mu_\Delta(\mathbf{x})||$ ▷ Approximate L.
  for $k = 0$ **to** K **do**
    $\mathbf{x}_{t,k} = \arg\max_{\mathbf{x}\in\mathbb{R}^d} \tilde{u}_{t,k}(\mathbf{x})$
    $\tilde{u}_{t,k}(\mathbf{x}) = \tilde{u}_{t,0}(\mathbf{x}) \prod_{j=1}^{k} \varphi(\mathbf{x}, \mathbf{x}_{t,j}, \hat{L}, \hat{M})$
  end for
end procedure

To adapt the acquisition function, a penalization ball is applied for each solution already selected for the batch, as can be seen in Equation 5.

$$\varphi(\mathbf{x}, \mathbf{x}_j, \hat{L}, \hat{M}) = \frac{1}{2} erfc(-z) \; where \; z = \frac{1}{\sqrt{2\sigma^2(\mathbf{x}_j)}}(\hat{L}||\mathbf{x}_j - \mathbf{x}|| + \hat{M} - \mu(\mathbf{x}_j)) \quad (5)$$

where $erfc$ is the complementary Gauss error function and L is the Lipschitz constant of the objective function (assuming that it is Lipschitz continuous). As suggested in [5], it can be approximated through the GP.

Finally, as in the first approach, it uses the SoD approach. In addition, we add the local search step to guarantee exploitation, where all the solutions in the batch are used as a starting point for the local search.

5 Experiments

The goal of this experimentation is to validate the BO approaches introduced in the previous section in the context of multi-modal optimization problems. We expect that BO will be able to model multi-modal landscapes and that will help to achieve a good covering of the optima. Moreover, the low-budget orientation of those algorithms may also help to achieve these goal with a limited number of function evaluations. The proposed clustering technique is supposed to have a better ability to model the landscape than the sequential one, and the batch approach may provide more diversity in the solutions due to the local penalization procedure. Although our main goal is investigate the benefits and limitations of the BO approaches, we will compare them to some of the best performing EAs since this is an area where highly multi-modal problems have been extensively investigated.

5.1 A framework for solving low-budget multi-modal optimization problems

To test our proposals, we will use the benchmark proposed in *CEC Niching Methods for Multi-modal Optimization* competition [10], where 20 multi-modal optimization problems were introduced, along with 3 performance metrics. Table 1 describes the benchmark.

Id	Dim.	# GO	Name	Characteristics
F_1	1	2	Five-Uneven-Peak Trap	Simple, deceptive
F_2	1	5	Equal Maxima	Simple
F_3	1	1	Uneven Maxima	Simple
F_4	2	4	Himmelblau	Simple, non-scalable, non-symmetric
F_5	2	2	Six-Hump Camel Back	Simple, non-scalable, non-symmetric
F_6	2, 3	18,81	Shubert	Scalable, # optima increase with d, unevenly dist. grouped optima
F_7	2, 3	36,216	Vincent	Scalable, # optima increase with d, unevenly dist. optima
F_8	2	12	Modified Rastrigin	Scalable, # optima independent from d, symmetric
F_9	2	6	Composition Function 1	Scalable, separable, non-symmetric.
F_{10}	2	8	Composition Function 2	Scalable, separable, non-symmetric.
F_{11}	2, 3, 5, 10	6	Composition Function 3	Scalable, non-separable, non-symmetr.
F_{12}	2, 3, 5, 10	8	Composition Function 4	Scalable, non-separable, non-symmetr.

Table 1: Set of multi-modal functions as originally introduced in [10].

To measure the performance of the optimization algorithms three performance metrics were introduced: Peak Ratio, Success Ratio, and Convergence speed.

- Peak Ratio (P_{ratio}) measures how many optima has been found over multiple runs in average: $P_{ratio} = \frac{\sum_{i=1}^{n_r} n_{g_i}}{n_g * n_r}$ where i is the run index and n_{g_i} is the number of global optima found at the end of process in each run. n_g refers to the number of known global optima, and n_r to the number of runs.
- Success Ratio (S_{ratio}) denotes the percentage of runs where all known global optima were found, out of all runs: $S_{ratio} = \frac{n_{sr}}{n_r}$ where n_{sr} is the number of successful runs.

- Convergence speed (C_{speed}) measures the number of function evaluations required to locate all known global optima, over multiple runs: $C_{speed} = \frac{\sum_{i=1}^{n_r} e_i}{n_r}$ where e_i is the number of evaluations used in each run to find all global optima. If all global optima is not found, the maximum function evaluations allowed is used.

To consider that a global optimum is found, the fitness value of the solution should be close to the best possible value, given an accuracy level. Five accuracy levels were considered to measure the performance metrics, as indicated in the benchmark: $\epsilon \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. Apart from that, the distance between the solution and the optimum should be lower than a threshold that depends on the optimization problem.

5.2 Experimental setup

We have conducted the experiments with the BO approaches introduced in Section 4 for all the functions in the benchmark and measured the performance metrics. Considering the stochastic nature of the algorithms, we conducted several runs of each algorithm configuration for each optimization problem. We were only able to perform 10 runs with our BO algorithms, while the EAs shown in the benchmark are averaged across 50 runs. All the results presented in this section have been measured with $\epsilon = 10^{-3}$ as accuracy level.

For all the BO approaches we have used the following configuration. First of all, regarding GPs, the mean function was set to $m(\mathbf{x}) = \theta_\mu$, where θ_μ is the mean of all fitness values obtained so far. On the other hand, for the kernel function, the Matern32 was employed, and for its parameter selection, the likelihood function was optimized every step with a grid search. Moreover, to limit the size of the covariance matrix, N was set to 500. UCB was used as the acquisition function adjusting the β constant as suggested in [22]. Finally, to maximize this acquisition function, a stochastic version of DIRECT [7] was applied. Being DIRECT a deterministic algorithm, repeated solutions may be selected to sample. In consequence, we decided to add a random shift to the starting point, in order to produce stochastic solutions.

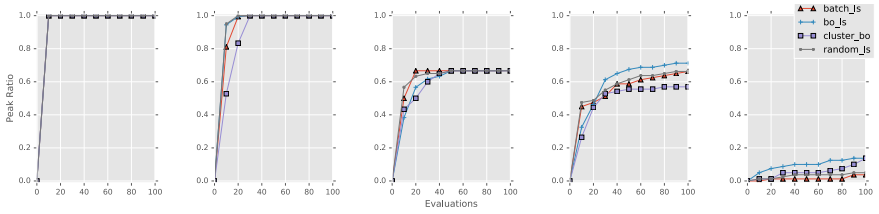
On the other hand, for the batch approach B was set to 10, and in the clustering BO algorithm, k-means was used to create the clusters with $K = 2$.

We used a bounded version of the Powell's conjugate direction method [13] to guide the local search. This algorithm does not need derivatives, which makes it suitable for our optimization problem. When an out-of-bounds solution is required by this algorithm, the worst possible value is returned.

In addition to the three variants of BO investigated, a random multi-start optimization algorithm has been added as a reference. It also uses Powell's method as the local search algorithm, starting from a random solution at each iteration.

5.3 Comparison between the different BO variants

Figure 2 shows the evolution of the peak ratio over function evaluations. In the figure, for illustration proposes, some functions with different number of dimensions are shown. In the lower axis, the number of evaluations are shown in percentage, while the P_{ratio} is illustrated in the vertical axis.



(a) F_2 1D (b) F_6 2D (c) F_{11} 5D (d) F_{12} 3D (e) F_{12} 10D

Figure 2: Evolution of the peak ratio over function evaluations (%) for the proposed algorithms and the random multi-start algorithm in five optimization problems.

As can be seen in Figure 2a, there are some functions in the benchmark were all the optima are found in the first 10% of the evaluations for all BO approaches. In Figure 2b, for example, it takes a little bit longer to achieve the best possible result, but the algorithms are able to solve this problem with a low-budget. On the contrary, in Figure2c, all the approaches find always the same optima but are not able to find more difficult ones in the second part of the optimization.

Figure 2d and Figure 2e correspond to F_{12} function, with 3 and 10 dimensions respectively. Here, it can be seen that the performance of all methods decreases in the higher dimensional problems. In this problem, bo_ls performs better than the other BO approaches. Regarding the C_{speed}, cluster_bo takes more evaluations to find all global optima in F_6 2D.

5.4 Comparison to state-of-the-art EAs for multi-modal problems

Here we compare BO approaches to NEA2 [14] and NMMSO [2] algorithms. Both algorithms won the *CEC Niching Methods for Multi-modal Optimization* competition [10] in 2013 and 2015 respectively.

Table 2 shows the peak-ratio of both algorithms, taken from the CEC 2015 competition, along with the peak-ratios computed from the results of all the algorithms we evaluate in our experimentation. The first five functions were omitted as all approaches were able to obtain the best possible score.

The sequential and the batch BO methods show competitive results in low dimensional functions compared to NEA2 and NMMSO. Our methods improve

	NEA2	NMMSO	random_ls	bo_ls	batch_ls	cluster_bo
F_6 2D	0.958	0.992	**1.000**	**1.000**	**1.000**	**1.000**
F_7 2D	0.918	**1.000**	0.889	0.864	0.892	0.917
F_6 3D	0.240	0.922	0.985	0.973	**0.989**	-
F_7 3D	0.584	**0.978**	0.571	0.782	0.722	-
F_8 2D	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
F_9 2D	0.967	0.990	**1.000**	**1.000**	**1.000**	**1.000**
F_{10} 2D	0.843	0.995	**1.000**	**1.000**	**1.000**	**1.000**
F_{11} 2D	0.960	**0.983**	0.684	0.767	0.684	0.684
F_{11} 3D	**0.810**	0.723	0.667	0.667	0.667	0.667
F_{12} 3D	**0.720**	0.642	0.662	0.713	0.662	0.569
F_{11} 5D	**0.673**	0.660	0.667	0.667	0.667	0.667
F_{12} 5D	**0.695**	0.470	0.412	0.375	0.425	0.338
F_{11} 10D	**0.667**	0.650	0.367	0.333	0.400	0.433
F_{12} 10D	**0.667**	0.457	0.050	0.138	0.037	0.138
F_{12} 20D	**0.360**	0.172	0.000	0.000	0.000	0.000

Table 2: Peak Ratio for the EAs, the random multi-start algorithm and the proposed algorithms for the functions presented in the benchmark. Best result for each function is written in bold. As all the algorithms get the best possible results in the first five functions their results are not shown. $\epsilon = 10^{-3}$ was used as accuracy level. *cluster_bo* was not able to finish the experimentation on time for the F_6 3D and F_7 3D problems.

the results of NEA2 and NMMSO for F_6 2D, F_6 3D, F_9 2D and F_{10} 2D functions. However, for these functions, the random multi-start algorithm achieves similarly good performance. Although the results of the sequential approach are similar to the random multi-start algorithm overall, the batch BO method is able to beat the random algorithm in 5 functions while the opposite occurs only once. The comparison among the BO approaches is favorable to the batch approach, beating the other proposed approaches in F_{12} 5D function, and obtaining the best result in F_6 3D overall. The cluster BO approach is able to get good results in F_7 2D and F_{10} 11D, but most of the comparisons are favorable to the other approaches. In functions with more dimensions, NEA2 clearly outperforms the rest of the algorithms.

5.5 Discussion

Among our BO approaches, bo_ls and batch_ls seem to be the most competitive proposals. They show good results for the lowest dimension (2D) problems, beating in some functions the state-of-the-art algorithms. However, for higher dimensions, particularly in F_{11} and F_{12}, their performance decreases. Having only one hyperparameter for all dimensions instead of many, may cause this dimensionality problem. It would be interesting to study in depth the characteristics of such parameters.

It is somehow surprising the good behavior of the baseline random multi-start approach, being able to identify all the optima in four problems, and outperforming the best algorithms. Being such low dimension, one could think that the function is easy to solve, but this is not the case (looking at NEA2 and NMMSO).

6 Conclusions

In this paper we have investigated the suitability of BO methods for massively multi-modal optimization problems traditionally addressed with EAs. We have proposed three new approaches to deal with these problems. These approaches incorporate ideas and insights of the previous work done in the EC filed. Although BO is usually used in low-budget optimization problems, we were able to achieve competitive results in optimization problems with low dimensions by modeling the search space.

Acknowledgments

This research has been partially supported by the Basque Government (ELKARTEK programs), and Spanish Ministry of Economy and Competitiveness MINECO (project TIN2016-78365-R). Jose A. Lozano is also supported by BERC 2018-2021 (Basque Government), and Severo Ochoa Program SEV-2017-0718 (Spanish Ministry of Economy, Industry and Competitiveness).

References

[1] Brochu, E., Cora, V.M., de Freitas, N.: A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. arXiv:1012.2599 [cs] (Dec 2010), http://arxiv.org/abs/1012.2599, arXiv: 1012.2599

[2] Fieldsend, J.E.: Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 2593–2600. IEEE (2014)

[3] Ginsbourger, D., Riche, R.L., Carraro, L.: A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. Tech. Rep. hal-00260579, CCSD (Mar 2008), https://hal.archives-ouvertes.fr/hal-00260579/document

[4] Goldberg, D.E., J.Richardson: Genetic algorithms with sharing for multimodal function optimisation. In: Proceedings of the of Second International Conference on Genetic Algorithms and Their Applications. pp. 41–49 (1987)

[5] González, J., Dai, Z., Hennig, P., Lawrence, N.D.: Batch Bayesian Optimization via Local Penalization. arXiv:1505.08052 [stat] (2015), http://arxiv.org/abs/1505.08052, arXiv: 1505.08052

[6] Guhaniyogi, R., Li, C., Savitsky, T.D., Srivastava, S.: A Divide-and-Conquer Bayesian Approach to Large-Scale Kriging. arXiv:1712.09767 [stat] (Dec 2017), http://arxiv.org/abs/1712.09767, arXiv: 1712.09767

[7] Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications **79**(1), 157–181 (Oct 1993). https://doi.org/10.1007/BF00941892, http://link.springer.com/article/10.1007/BF00941892

[8] Jones, D.R.: A Taxonomy of Global Optimization Methods Based on Response Surfaces. Journal of Global Optimization **21**(4), 345–383 (Dec 2001). https://doi.org/10.1023/A:1012771025575, http://link.springer.com/article/10.1023/A%3A1012771025575

[9] Kvasov, D.E., Sergeyev, Y.D.: Deterministic approaches for solving practical black-box global optimization problems. Advances in Engineering Software **80**, 58–66 (Feb 2015). https://doi.org/10.1016/j.advengsoft.2014.09.014, http://www.sciencedirect.com/science/article/pii/S096599781400163X

[10] Li, X., Engelbrecht, A., Epitropakis, M.G.: Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization. Tech. Rep., Royal Melbourne Institute of Technology (Mar 2013), http://goanna.cs.rmit.edu.au/xiaodong/cec13-niching/

[11] Mockus, J.: The Bayesian Approach to Local Optimization. In: Bayesian Approach to Global Optimization, pp. 125–156. No. 37 in Mathematics and Its Applications, Springer Netherlands (1989), http://link.springer.com/chapter/10.1007/978-94-009-0909-0_7

[12] Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of IEEE International Conference on Evolutionary Computation. pp. 798–803 (1996). https://doi.org/10.1109/ICEC.1996.542703

[13] Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal **7**(2), 155–162 (Jan 1964). https://doi.org/10.1093/comjnl/7.2.155, http://comjnl.oxfordjournals.org/content/7/2/155

[14] Preuss, M.: Niching the CMA-ES via Nearest-better Clustering. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 1711–1718. GECCO '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1830761.1830793, http://doi.acm.org/10.1145/1830761.1830793

[15] Preuss, M.: Multimodal Optimization by Means of Evolutionary Algorithms. Natural Computing Series, Springer International Publishing (2015), https://www.springer.com/la/book/9783319074061

[16] Quiñonero-Candela, J., Rasmussen, C.E.: A Unifying View of Sparse Approximate Gaussian Process Regression. Journal of Machine Learning Research **6**(Dec), 1939–1959 (2005), http://www.jmlr.org/papers/v6/quinonero-candela05a

[17] Rasmussen, C.E., Williams, C.K.: Gaussian processes for machine learning. MIT Press (2006)

[18] Roman, I., Santana, R., Mendiburu, A., Lozano, J.A.: Dynamic Kernel Selection Criteria for Bayesian Optimization. In: BayesOpt 2014: NIPS Workshop on Bayesian Optimization. Montreal, Canada (2014), http://bayesopt.github.io/papers/paper13.pdf

[19] Sareni, B., Krahenbuhl, L.: Fitness sharing and niching methods revisited. IEEE Transactions on Evolutionary computation **2**(3), 97–106 (1998)

[20] Silverman, B.W.: Some aspects of the spline smoothing approach to non-parametric regression curve fitting. Journal of the Royal Statistical Society. Series B (Methodological) pp. 1–52 (1985)

[21] Singh, G., Deb, K.: Comparison of Multi-modal Optimization Algorithms Based on Evolutionary Algorithms. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. pp. 1305–1312. GECCO '06, ACM, New

York, NY, USA (2006). https://doi.org/10.1145/1143997.1144200, http://doi.acm.org/10.1145/1143997.1144200

[22] Srinivas, N., Krause, A., Kakade, S., Seeger, M.: Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 1015–1022 (2010), http://www.icml2010.org/papers/422.pdf

[23] Wang, H., van Stein, B., Emmerich, M., Bäck, T.: Time Complexity Reduction in Efficient Global Optimization Using Cluster Kriging. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 889–896. GECCO '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071321, http://doi.acm.org/10.1145/3071178.3071321

[24] Wong, K.C., Leung, K.S., Wong, M.H.: Protein Structure Prediction on a Lattice Model via Multimodal Optimization Techniques. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. pp. 155–162. GECCO '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1830483.1830513, http://doi.acm.org/10.1145/1830483.1830513

[25] Yin, X., Germay, N.: Investigations on solving the load flow problem by genetic algorithms. Electric Power Systems Research **22**(3), 151–163 (1991). https://doi.org/10.1016/0378-7796(91)90001-4, http://www.sciencedirect.com/science/article/pii/0378779691900014

[26] Zhigljavsky, A., Žilinskas, A.: Methods Based on Statistical Models of Multimodal Functions. In: Stochastic Global Optimization, pp. 149–244. No. 9 in Springer Optimization and Its Applications, Springer US (2008). https://doi.org/10.1007/978-0-387-74740-8_4, http://link.springer.com/chapter/10.1007/978-0-387-74740-8_4