Bayesian Optimization for Parameter Tuning in Evolutionary Algorithms

Ibai Roman*, Josu Ceberio†, Alexander Mendiburu* and Jose A. Lozano*
*Faculty of Computer Science, 20018 Donostia, Spain
† School of Engineering, 48013 Bilbao, Spain
email: {ibai.roman, josu.ceberio, alexander.mendiburu, ja.lozano}@ehu.eus
University of the Basque Country UPV/EHU

***Abstract*—Advances in evolutionary computation have demonstrated that Evolutionary Algorithms (EAs) proposed in this area are a solid alternative for solving combinatorial and continuous optimization problems. Despite their success in innumerable real-world scenarios, EAs depend on a set of input parameters that characterize their performance and need to be adjusted. In fact, identifying and setting the most appropriate parameters for an EA is a complex task, which, in some cases, can be as difficult as the optimization problem at hand.**

Recently, parameter tuning has attracted the interest of the research community, designing and proposing techniques that (1) help the algorithm to perform to its best, and (2), indirectly, make fairer comparisons of different methods. In this manuscript, we propose a novel offline parameter tuning algorithm based on Bayesian Optimization, a sequential design strategy for global optimization. In order to illustrate the validity of the proposed method, we considered as a case of study the Hybrid Kernel EDA, an EA that is characterized by 6 parameters. We ran the algorithm with the parameters tuned by means of Bayesian Optimization, and compared the results with those obtained by setting the parameters by hand (using some prior knowledge). Experiments were carried out on a benchmark of 60 instances of the permutation flowshop scheduling problem. Experimental results show that, in general, Hybrid Kernel EDA obtains better results when using the parameters tuned by means of Bayesian Optimization.

I. Introduction

In the last decades, the advances given in the field of optimization have postulated Evolutionary Algorithms (EAs) [1] as a solid alternative for solving combinatorial and continuous optimization problems. Flowshop scheduling [2], nuclear reactor fuel management [3], robotics [4], load balancing in communication networks [5] or chemotherapy treatment [6] are some examples of real-world problems where EAs have been successfully applied.

In spite of their success, EAs require a set of input parameters that need to be fine-tuned in order to obtain a good performance. Each parameter, discrete or continuous, influences the overall behavior of the algorithm. Unfortunately, with the exception of some, but little, intuition on a few parameters, there is no rule of thumb to optimize this set, thus, some domain knowledge is needed in order to choose a good set. In addition, a brute-force approach is intractable. For this reason, preliminary experiments are usually conducted with the intention of gaining some intuition on the set of parameters.

The task of finding the parameter set of an algorithm that maximizes its performance can be modeled as a non-linear optimization problem. In this sense, each parameter set $\mathbf{x}$ can be seen as a candidate solution in the search space, and the objective value obtained from running the algorithm with $\mathbf{x}$, $f(\mathbf{x})$, is a quality measure of $\mathbf{x}$. So, the optimization problem can be formally expressed as follows:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \tag{1}$$

where d is the number of parameters to optimize, and the goal is to minimize the objective function.

Usually, the objective function f is a black-box function, and its analytic form is unknown [7]. Since 2006, parameter tuning has been a recurrent topic. In this sense, different strategies, such as experimental design techniques [8], evolutionary algorithms [9], regression models [10] or sequential parameter optimization [11], have been proposed. In this line, López-Ibáñez et al. [12] presented the `irace` package, software that tries to find the most appropriate algorithm settings for a given set of instances. In the last few years, this software has been successfully applied in many research works [13]–[15].

A common drawback of the methods cited above lies in the considerable number of evaluations needed to adjust parameters. Particularly, if performing the evaluation of a parameter configuration is expensive, i.e. running an EDA, then the previous methods are not affordable.

Motivated by the characteristics of the presented scenario, in this manuscript, we introduce an offline parameter tuning algorithm based on Bayesian Optimization (BO) [23]–[25]. BO is a global optimization method designed to deal with expensive-to-evaluate black-box objective functions. This method has been quite popular in the tuning of parameters of machine learning models [25].

In order to illustrate the validity of the proposed method, we considered a hybrid version of the Kernel EDA [16], called HKEDA, which is characterized by 6 parameters. Particularly, we evaluated the performance of HKEDA with the parameters tuned by means of BO, and compared the results with those obtained by setting the parameters by hand (after some preliminary experiments). An experimental benchmark of 60 instances of the permutation flowshop scheduling problem (PFSP) [2] was selected. According to the results, in general, Hybrid Kernel EDA obtains better results when using the parameters tuned by means of Bayesian Optimization. Finally, in order to observe the behavior of HKEDA with respect to the state-of-the-art approaches for the PFSP, we compare it with

978-1-5090-0623-6/16/$31.00

the algorithms studied in [2], verifying that its performance is really competitive.

The remainder of the paper is organized as follows: in the next section a brief introduction to Bayesian Optimization and Gaussian Processes is presented. Then, in Section III the case of study is described: Hybrid Kernel EDA. Experimental design and results are explained afterwards in Section IV. Finally, conclusions and ideas for future work are presented in Section V.

II. Bayesian Optimization and Gaussian Processes

Bayesian Optimization (BO) [23]–[25] is a global optimization method designed to deal with expensive-to-evaluate black-box objective functions. Although its roots date back to the 70s [26], the last ten years have seen a tremendous development of the technique, both in terms of theoretical achievements and also applications to real-world problems. These developments have been mainly motivated by its use in the field of machine learning where, for instance, the tuning of parameters of a predictive model can be seen as a complex optimization problem.

BO evolves by sampling a point[1] to evaluate at each step of the search. The key idea of BO is the use of a surrogate model of the function to optimize, which is updated with the information provided by each evaluated point. This surrogate model is used to choose the next point to evaluate by means of what is called an acquisition function. This acquisition function explicitly balances exploration versus exploitation in the choice of the next point.

The name Bayesian has to do with the fact that, in most of the cases, the surrogate model is given by a stochastic process assuming a probability distribution over the set of objective functions. Initially, an a priori stochastic process is assumed and, each time a point is evaluated, a posteriori process is obtained following Bayesian statistics. The most recurrent stochastic process used with BO and the one chosen in this work is the Gaussian process. Proof of convergence of this method has been provided in the case of objective functions with specific characteristics [27].

A psuedo-code for a general BO algorithm is given in Algorithm 1 (as presented in [24]). The next section explains in more detail the components of BO.

Algorithm 1: **BO algorithm**

```
/* Sample a random point */
y_1 = f(x_1)
/* Initialize the data set and the surrogate model */
D_{1:1} = {(x_1, y_1)}
SM ← D_{1:1}
t = 2
do {
    /* Select the next point to evaluate by optimizing the acquisition function */
    x_t = arg max_{x∈A} u(x|SM)
    /* Evaluate the objective function */
    y_t = f(x_t)
    /* Update the data set and the surrogate model */
    D_{1:t} = D_{1:t-1} ∪ {(x_t, y_t)}
    SM ← D_{1:t}
    t = t + 1
} while the stop criterion is not met
```

A. Gaussian Process

A Gaussian Process (GP) is a stochastic process that considers a distribution over functions. Basically, it can be interpreted as a Gaussian distribution in an infinite number of dimensions. Furthermore, the marginal distribution of any finite collection of dimensions follows a multivariate Gaussian distribution. A GP is defined by two parameters: a mean function $m(\mathbf{x})$ and a kernel $k(\mathbf{x}, \mathbf{x}')$. The kernel function provides information about the correlation between the objective function values, $f(\mathbf{x})$ and $f(\mathbf{x}')$ of two points **x** and **x'**. In this paper, we considered the Matern 5/2 kernel:

$$k_{M52}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \left(1 + \sqrt{5} r(\mathbf{x}, \mathbf{x}') + \frac{5}{3} r^2(\mathbf{x}, \mathbf{x}')\right) + \exp\left(-\sqrt{5} r(\mathbf{x}, \mathbf{x}')\right) + \theta_n \quad (2)$$

where

$$r^2(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{n_d} \left(\frac{x_d - x'_d}{\theta_d}\right)^2$$

being θ_d the length-scale parameter for each dimension d, expressing the relevance of each dimension in the output, and θ_0 and θ_n the amplitude and noise parameters respectively.

GPs have been previously used in the field of evolutionary algorithms. For instance, in [28] a GP is used to model the Pareto set in a multi-objective problem, and in [29] it is used as a surrogate model in an evolutionary algorithm for medium-scale computationally expensive problems.

B. Acquisition function

The acquisition function is in charge of giving a utility value to each point of the search space by making use of the surrogate model. It explicitly balances exploration and exploitation.

The literature contains numerous acquisition functions such as GP-UCB [30], probability of improvement [31] or expected improvement [26]. We have chosen expected improvement (u_{EI}), as it has shown a competitive performance in BO [27] and depends only on one parameter, which simplifies its configuration. This acquisition function measures the expectation of improving the best result up to now, and is defined as:

[1]We will use the term point to distinguish the elements of the search space where the BO searches from those of the main combinatorial optimization problem that we plan to solve, which will be called solutions.

$$u_{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}$$

where

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{3}$$

where $\mathbf{x}^+$ is the best point so far. Φ denotes the cumulative distribution function (CDF) and ϕ the probability density function (PDF) of a standard normally distributed variable. ξ is a parameter that can be used to balance the exploration versus exploitation trade-off.

In the previous equation, $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are given by the current GP. Specifically, at the $(t+1)$-th iteration, where t points have been evaluated, $D_{1:t} = \{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)) \ldots, (\mathbf{x}_t, f(\mathbf{x}_t))\}$, those values can be calculated as follows:

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

where $k(\mathbf{X}_{1:t}, \mathbf{x})$ (resp. $k(\mathbf{x}, \mathbf{X}_{1:t})$), represents the vector $(k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \ldots, k(\mathbf{x}_t, \mathbf{x}))$ (resp. $(k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \ldots, k(\mathbf{x}, \mathbf{x}_t))^T$) and $k(\mathbf{X}_{1:t}, \mathbf{X}_{1:t})$ is the matrix whose entries are the values $k(\mathbf{x}_i, \mathbf{x}_j)$ with $i, j = 1, \ldots, t$.

The search for the point that maximizes the acquisition function (step 10 of Algorithm 1) can be performed with many different techniques. In this work we follow the approach based on the use of a Sobol grid as detailed in [25].

III. A Case of Study: the Hybrid Kernel EDA

Estimation of Distribution Algorithms (EDAs) [17] constitute a powerful tool for optimization. Based on machine learning techniques, at each generation, EDAs estimate a joint probability distribution associated with the set of most promising individuals, trying to explicitly express the interrelations between the variables. Sampling the probabilistic model learnt in the previous generation, a new population of solutions is created. The algorithm stops iterating and returns the best solution across the generations when a certain stopping criterion is met.

Ceberio et al. [2], [16], [18] firstly proposed using probability models specifically designed for permutation domains in EDAs. Particularly, the authors introduced distance-based probability models such as the Mallows and Generalized Mallows. These models are analogous over permutations to the Gaussian distribution: they are both exponential unimodal models defined by two parameters, the mean and the variance. Under this model, the probability of every permutation $\pi \in \mathbb{S}_n$ depends on two parameters: a spread parameter θ[2] and the distance to a central permutation π_0.

[2]The Generalized Mallows model is defined with $n-1$ spread parameters.

In this paper, we use an algorithm similar to the Hybrid Generalized Mallows EDA [2] for solving the PFSP. Particularly, this new approach replaces the Generalized Mallows model with Mallows model kernels [16][3]. The resulting algorithm is called Hybrid Kernel EDA (HKEDA). In the following lines, we present its main stages:

- As proposed in [2], the algorithm starts by generating the initial population based on the LR(n/m) heuristic algorithm.
- In the second stage, Kernel EDA is executed with the population created with LR(n/m). At each generation of the EDA, one Mallows kernel is defined for each selected solution, using that solution as the central permutation of the model. As regards the spread parameter θ, a value must be set. It is worth mentioning that this parameter is critical, since it controls the probabilities assigned to each permutation according to its distance to the central permutation (exploration / exploitation ratio). Once the kernels are defined, new solutions are then sampled from each kernel using the methods explained in [19]. In order to prevent HKEDA from getting stuck, the algorithm guarantees that all the solutions in the population are different by checking whether the sampled solution already exists in the population.
  In any case, as seen in the literature, the elitism and truncation may lead the algorithm to lose diversity in the population. Therefore, a restart mechanism is applied when a maximum number of generations is reached without improvement. The restart mechanism generates a new population by applying a shake procedure over the best individual found so far. The shake procedure consists of choosing a percentage of items and moving them to random positions. The first stage terminates and returns the best solution found so far when a maximum number of function evaluations or a given number of restarts are reached.
- In the third stage of HKEDA, a VNS is applied to the (best) solution obtained in the previous stage. The VNS performs as follows: in the first step, a greedy local search procedure is carried out using the *interchange* neighborhood until a local optima is reached. In the next step, the algorithm chooses the best neighbor in the *insert* neighborhood. The algorithm repeats this procedure until a local optimum is found for both neighborhoods. If the obtained solution is better than that found by the VNS so far, the best solution is updated. Next, a shake procedure, similar to that employed in the restart mechanism, is applied. The whole process is repeated until the maximum number of evaluations is reached.

The stopping criterion is defined as a maximum number of function evaluations. The second and third stages, each perform, a priori, half of the evaluations. If the criterion on restarts is fulfilled before half of the evaluations have been

[3]Based on the results reported in [16], the Cayley distance has been the metric considered under the Mallows model.

consumed, the remaining evaluations are performed in the third step by the VNS.

The input parameters of HKEDA that need to be set are the following:

- Spread parameter θ.
- Population size.
- Selection size.
- Offspring size.
- Maximum number of generations without improvement.
- Maximum number of restarts.
- % of items to be moved in the restart shake.
- % of items to be moved in the VNS shake.

IV. Experimental design

In order to assess whether the Bayesian Optimization (BO) approach is suitable for offline parameter tuning, we considered as a benchmark the HKEDA and the permutation flowshop scheduling problem (PFSP) [2]. Particularly, we selected 60 instances from the Taillard's instance-set [20] with structures 50×10, 50×20, 100×10, 100×20, 200×10 and 200×20 (10 instances of each structure). We then ran two versions of HKEDA, one with parameters optimized with BO, called HKEDA.bo, and another with parameters set by hand, called HKEDA.fix. It must be noted that the parameters set by hand were those for which HKEDA showed the best results in the previous experimentation.

From the parameters enumerated above, in the following list, the parameters to fine-tune and their ranges are presented (they have been set without performing any previous experimentation):

- Spread parameter θ: [10,15]
- Population size (**p**): [50,1000]
- Maximum number of generations without improvement. (**m**): [100,1000]
- Maximum number of restarts (**r**): [10,100]
- % of items to be moved in the restart shake (**b**): [0.01,0.99]
- % of items to be moved in the VNS shake (**v**): [0.01,0.99]

As regards the selection size, 10% of the individuals are considered by truncation, and the number of sampled offspring equals the size of the population. We fix both by hand as we consider their influence to be lower than that of the previous parameters. Nevertheless, they could also be included in the BO step.

The stopping criterion has been set [2] to a maximum number of function evaluations (see Table I).

In order to get the best-performing parameter set, we run BO for each instance-structure, taking as reference one instance from each set (the first one). Each BO-run tests 30 different parameter configurations and, since EDAs are stochastic algorithms, five HKEDA runs are completed for each configuration, averaging the obtained fitness values. The whole process is repeated 10 times for each reference instance, and the parameter set that obtained the best results is selected. The parameter sets are summarized in Table I.

TABLE I
Best parameter settings for the Hybrid Kernel EDA provided by BO for each reference instance

Instance	θ	**p**	**m**	**r**	**b**	**v**	evaluations
50x10	11.22	700	876	92	0.29	0.07	256208100
50x20	14.83	428	555	56	0.81	0.37	275954150
100x10	12.25	975	251	93	0.68	0.11	266211000
100x20	11.10	479	455	17	0.60	0.09	283040000
200x10	13.38	783	609	90	0.17	0.02	272515500
200x20	14.78	955	896	73	0.17	0.01	287728850

In addition, in order to prove the validity of BO for parameter tuning, Fig 1 shows, at each iteration, the average of 10 repetitions (*dashed*) and the best configuration (*solid*) found so far.

The plots reveal very different scenarios. For the smallest instances, 50×10 and 50×20, the BO shows a large variability in the results. Contrarily, for the largest instances, it is clear that, as the optimizer progresses, configurations with better fitness are found.

A. HKEDA.bo versus HKEDA.fix

In the second step of the experimental study, the performance of the HKEDA.bo and HKEDA.fix is compared. To this end, each *algorithm - instance* pair was run 20 times. HKEDA.bo is configured using the parameters described in Table I. As regards HKEDA.fix, the parameters have been set as follows:

- Spread parameter θ: the value that assigns $P(\sigma_0) = 0.9$ with the Mallows model under the Cayley distance. For $n = 50, 100$ and 200[4], the corresponding values are 9.4, 10.8 and 12.2 respectively.
- Population size (**p**): $10n$.
- Maximum number of generations without improvement. (**m**): 500.
- Maximum number of restarts (**r**): $10n$.
- % of items moved in the restart shake (**b**): 5 items are moved.
- % of items moved in the VNS shake (**v**): 10 items are moved.

To study the performance of both approaches, the average relative percentage deviation (ARPD) values to the best known results are measured. The ARPD is calculated as

$$ARPD = \frac{\frac{1}{20}\sum_{i=1}^{20} Algorithm_i}{Best} - 1 \qquad (5)$$

where $Algorithm_i$ denotes the solution found by the algorithm in the ith repetition, and $Best$ stands for the state-of-the-art results reported in [2]. The results are summarized in Table II. The experiment shows that HKEDA.bo outperforms HKEDA.fix in 48 instances out of 60 as regards the ARPD. In order to assess whether the differences seen in Table II are significant, we applied the Mahn-Whitney test on each instance (Wilcoxon non-paired), and the p-values were adjusted with

[4] n denotes the size of the problem.

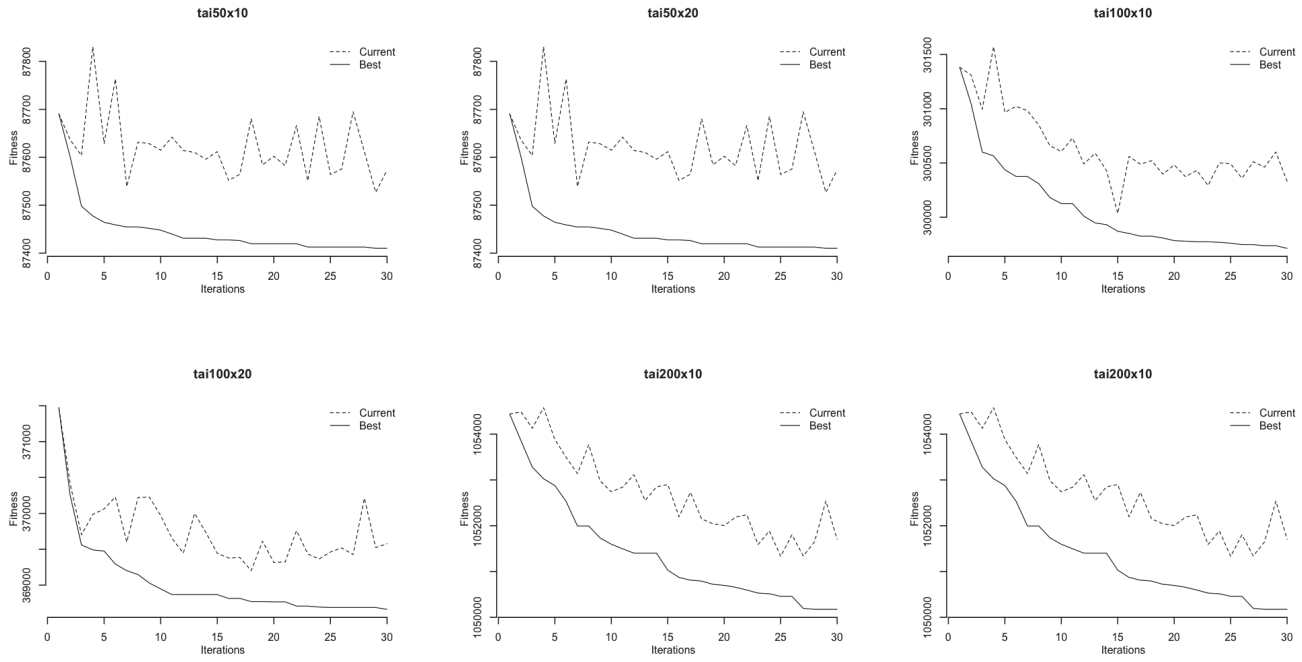

Fig. 1. The fitness of the configurations evaluated by the Bayesian Optimizer for the reference instance of types 50×10, 50×20, 100×10, 100×20, 200×10 and 200×20. Averages of 10 repetitions of the current (*dashed*) and best (*solid*) solutions obtained at each iteration have been calculated.

the Finner's method[5]. A level of significance $\alpha = 0.05$ was set. The results are presented in column "Sig." in Table II. We used the symbols '+' and "=" to denote that HKEDA.*bo* is statistically better than HKEDA.fix, or that no significant differences were found.

The statistical test concludes that HKEDA.*bo* is statistically better that HKEDA.fix in 28 out of 60 instances, while, for the rest of the instances, no significant differences were found. It is especially meaningful to see that HKEDA.*bo* performs the best in the largest instances.

B. State-of-the-art algorithms

Finally, in order to obtain a bigger picture, we would like to observe the performance of HKEDA.*bo* compared to the state-of-the-art algorithms for PFSP. In this sense, we ran 20 repetitions of Hybrid Generalized Mallows EDA (HGMEDA) [2] and Asynchronous Genetic Algorithm (AGA) [22] and calculated the corresponding ARPD values as detailed in Eq. 5. The obtained results (see Fig. 2) reveal that, for instances with $n = 50$, AGA is the most competitive algorithm. For instances with $n = 100$, AGA, HGMEDA and HKEDA.*bo* are similarly competitive. On the contrary, for the largest instances, $n = 200$, HGMEDA is clearly the algorithm that performs the best, followed by HKEDA.*bo*.

[5]The statistical tests in this work have been carried out with **scmamp** package for R [21], following the guidelines included in the documentation of the package

V. Conclusion & Future Work

In this manuscript, we proposed an offline parameter tuning method based on Bayesian Optimization. As a case of study, the Hybrid Kernel EDA and the permutation flowshop scheduling problem were considered as a benchmark. Particularly, BO has been used to fine-tune six parameters of HKEDA. According to the results, HKEDA initialized with the parameters proposed by BO obtains better results compared to the same algorithm using parameters fixed by hand. In summary, BO appears to be a useful tool for parameter tuning. In addition, it could also be used to put different algorithms under equal conditions (best configurations) so that fair comparisons can be conducted.

In this work, we obtained an idea of the suitability of BO for parameter tuning in EAs. Nonetheless, as future work, we plan to analyze the competitiveness of BO for tuning the parameters on a representative set of metaheuristic algorithms (GA, VNS, ES,...), and compare it to other state-of-the-art proposals such as `irace` [12]. Alternatively, we aim to analyze other configurations that BO may adopt. For instance, different acquisition functions, kernel functions or optimizers for the surrogate model could be evaluated.

Acknowledgments

This work has been partially supported by the Research Groups 2013-2018 (IT-609-13) programs (Basque Government), TIN2013-41272P (Ministry of Science and Technol-

TABLE II
ARPD RESULTS OBTAINED WITH THE TWO PARAMETER CONFIGURATION OPTIONS, HKEDA.*bo* AND HKEDA.fix ON THE 60 INSTANCES. THE RESULTS IN BOLD DENOTE THE APPROACH WITH THE LOWEST ARPD.

Instance	ID	Best Known	HKEDA.*bo*	HKEDA.fix	Sig.	Instance	ID	Best Known	HKEDA.*bo*	HKEDA.fix	Sig.
50x10	1	87207	0.005	**0.004**	=	100x20	1	367267	0.007	**0.007**	=
	2	82820	**0.007**	0.007	=		2	374032	**0.005**	0.009	+
	3	79987	0.005	**0.004**	=		3	371417	0.005	**0.005**	=
	4	86581	0.004	**0.003**	=		4	373822	0.008	**0.007**	=
	5	86377	0.005	**0.004**	=		5	370697	**0.004**	0.006	+
	6	86637	0.004	**0.004**	=		6	372768	**0.005**	0.008	+
	7	88750	0.006	**0.005**	=		7	374483	**0.007**	0.010	=
	8	86824	**0.003**	0.004	=		8	385456	**0.005**	0.006	=
	9	85526	**0.005**	0.006	=		9	375858	**0.005**	0.007	+
	10	88077	0.006	**0.005**	=		10	379899	**0.007**	0.008	=
50x20	1	125831	**0.004**	0.004	=	200x10	1	1047662	**0.003**	0.004	+
	2	119259	**0.003**	0.003	=		2	1036042	**0.006**	0.008	+
	3	116459	**0.004**	0.005	+		3	1047571	**0.003**	0.006	+
	4	120712	**0.004**	0.007	=		4	1031794	**0.003**	0.007	+
	5	118184	0.005	**0.004**	=		5	1036770	**0.002**	0.006	+
	6	120703	**0.003**	0.005	=		6	1006650	**0.004**	0.006	+
	7	122962	**0.004**	0.004	=		7	1053390	**0.005**	0.009	+
	8	122489	**0.003**	0.004	=		8	1046246	**0.003**	0.006	+
	9	121872	**0.003**	0.005	+		9	1025145	**0.002**	0.006	+
	10	124064	**0.004**	0.006	+		10	1031176	**0.004**	0.008	+
100x10	1	299301	**0.003**	0.004	+	200x20	1	1226879	**0.005**	0.007	+
	2	274593	0.008	**0.008**	=		2	1241811	**0.005**	0.010	+
	3	288630	**0.006**	0.008	=		3	1266153	**0.006**	0.007	=
	4	302105	**0.004**	0.006	=		4	1237053	**0.009**	0.011	+
	5	285340	**0.005**	0.006	=		5	1223551	**0.006**	0.010	=
	6	270693	**0.003**	0.007	+		6	1225254	**0.008**	0.009	=
	7	280457	**0.003**	0.004	+		7	1241847	**0.006**	0.007	+
	8	291665	**0.004**	0.006	=		8	1240820	**0.009**	0.012	+
	9	302624	**0.005**	0.007	=		9	1229066	**0.008**	0.010	+
	10	292230	**0.003**	0.005	+		10	1247156	**0.007**	0.009	+

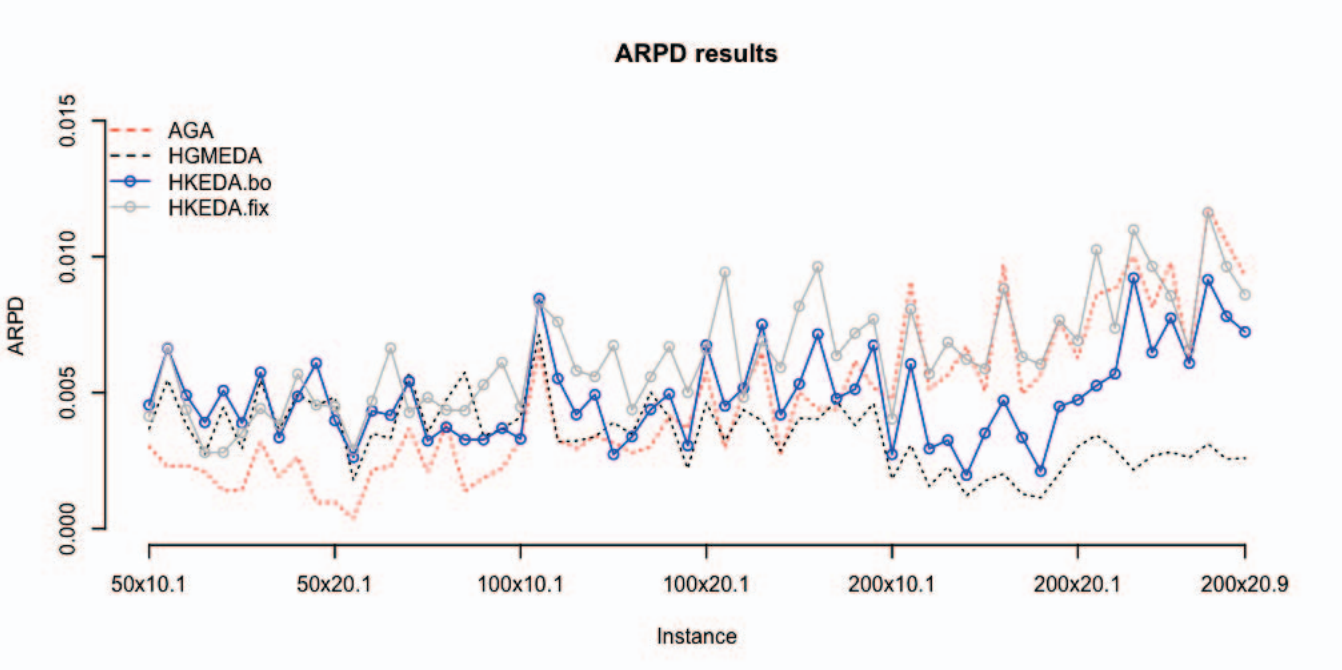

Fig. 2. ARPD results of AGA, HGMEDA, HKEDA.*bo* and HKEDA.fix on 60 PFSP instances. Each value was obtained averaging results of 20 repetitions.

ogy). Ibai Roman holds a predoctoral grant from the Basque Government.

REFERENCES

[1] D. E. Goldberg and R. L. Jr., "Alleles Loci and the Traveling Salesman Problem," in *Proceedings of an International Conference on Genetic*

Algorithms and Their Applications, vol. 154. Lawrence Erlbaum, Hillsdale, NJ, July 1985, pp. 154–159.

[2] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A Distance-based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 286 – 300, April 2014.

[3] S. Jiang, A. Ziver, J. Carter, C. Pain, A. Goddard, S. Franklin, and H. Phillips, "Estimation of Distribution Algorithms for nuclear reactor fuel management optimisation," *Annals of Nuclear Energy*, vol. 33, no. 11-12, pp. 1039–1057, 2006.

[4] C. Blum and R. Groß, "Swarm intelligence in optimization and robotics," in *Springer Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Springer Berlin Heidelberg, 2015.

[5] G. Di Caro and M. Dorigo, "Antnet: Distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research*, vol. 9, no. 1, pp. 317–365, Dec. 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622797.1622806

[6] A. E. I. Brownlee, M. Pelikan, J. A. W. McCall, and A. Petrovski, "An application of a multivariate estimation of distribution algorithm to cancer chemotherapy," in *Proceedings of the 10th Genetic and Evolutionary Computation Conference*, ser. GECCO '08, C. Ryan and M. Keijzer, Eds., 2008, pp. 463–464.

[7] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[8] B. Adenso-Diaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," *Oper. Res.*, vol. 54, no. 1, pp. 99–114, Jan. 2006.

[9] C. Ansótegui, M. Sellmann, and K. Tierney, "A gender-based genetic algorithm for the automatic configuration of algorithms," in *Principles and Practice of Constraint Programming - CP 2009*, ser. Lecture Notes in Computer Science, I. P. Gent, Ed. Springer Berlin Heidelberg, 2009, vol. 5732, pp. 142–157.

[10] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, ser. LION'05. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 507–523.

[11] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss, "The sequential parameter optimization toolbox," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 337–360.

[12] M. Lopez-Ibañez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package: Iteratec racing for automatic algorithm configuration," IRIDIA, Tech. Rep. TR/IRIDIA/2011-004, February 2011.

[13] T. Liao, T. Stützle, M. A. M. de Oca, and M. Dorigo, "A unified ant colony optimization algorithm for continuous optimization," *European Journal of Operational Research*, vol. 234, no. 3, pp. 597–609, 2014.

[14] Z. Ren, H. Jiang, J. Xuan, and Z. Luo, "Hyper-heuristics with low level parameter adaptation," *Evolutionary computation*, vol. 20, no. 2, pp. 189–227, 2012.

[15] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari, "Automode: A novel approach to the automatic design of control software for robot swarms," *Swarm Intelligence*, vol. 8, no. 2, pp. 89–112, 2014.

[16] J. Ceberio, A. Mendiburu, and J. A. Lozano, "Kernels of mallows models for solving permutation-based problems," in *In Proceedings of 2015 Genetic and Evolutionary Computation Conference (GECCO-2015)*, Madrid, Spain, July 2015, pp. 505–512.

[17] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[18] J. Ceberio, A. Mendiburu, and J. A. Lozano, "Introducing the Mallows Model on Estimation of Distribution Algorithms," in *Proceedings of International Conference on Neural Information Processing (ICONIP)*, ser. Lecture Notes in Computer Science, B.-L. Lu, L. Zhang, and J. T. Kwok, Eds. Springer, 2011, pp. 461–470.

[19] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "Extending distance-based ranking models in estimation of distribution algorithms," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*. IEEE, July 2014.

[20] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, January 1993.

[21] B. Calvo and G. Santafe, *scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems*, 2015, r package version 2.0.

[22] X. Xu, Z. Xu, and X. Gu, "An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization," *Expert Systems with Applications*, vol. 38, no. 7, pp. 7970 – 7979, 2011.

[23] J. Mockus, "The Bayesian Approach to Local Optimization," in *Bayesian Approach to Global Optimization*, ser. Mathematics and Its Applications. Springer Netherlands, 1989, no. 37, pp. 125–156. [Online]. Available: http://link.springer.com/chapter/10.1007/978-94-009-0909-0_7

[24] E. Brochu, V. M. Cora, and N. d. Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," Department of Computer Science, University of British Columbia, Tech. Rep. TR-2009-23, Nov. 2009.

[25] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959. [Online]. Available: http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf

[26] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," in *Towards Global Optimization*. Elsevier, 1978, vol. 2, pp. 117–129.

[27] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, May 2012.

[28] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using gaussian process-based inverse modeling," *Evolutionary Computation, IEEE Transactions on*, vol. 19, no. 6, pp. 838–856, Dec 2015.

[29] B. Liu, Q. Zhang, and G. Gielen, "A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 2, pp. 180–192, April 2014.

[30] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "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*, 2010, pp. 1015–1022. [Online]. Available: http://www.icml2010.org/papers/422.pdf

[31] H. J. Kushner, "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise," *J. Basic Eng*, vol. 86, no. 1, pp. 97–106, Mar. 1964. [Online]. Available: http://dx.doi.org/10.1115/1.3653121