Multi-objective hyperparameter optimization with performance uncertainty

Non Peer-reviewed author version

DOI: 10.1007/978-3-031-22039-5_4
Handle: http://hdl.handle.net/1942/38775

Multi-objective hyperparameter optimization with performance uncertainty*

Alejandro Morales-Hernández[1,2,3][0000−0003−0053−4902], Inneke Van Nieuwenhuyse[1,2,3][0000−0003−2759−3726], and Gonzalo Nápoles[4][0000−0003−1936−3701]

[1] Core Lab VCCM, Flanders Make, Limburg, Belgium
[2] Research Group Logistics, Hasselt University, Agoralaan Gebouw D, Diepenbeek, 3590, Limburg, Belgium
[3] Data Science Institute, Hasselt University, Agoralaan Gebouw D , Diepenbeek, 3590, Limburg, Belgium
{alejandro.moraleshernandez, inneke.vannieuwenhuyse}@uhasselt.be
[4] Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands
g.r.napoles@uvt.nl

Abstract. The performance of any Machine Learning algorithm is impacted by the choice of its hyperparameters. As training and evaluating a ML algorithm is usually expensive, the hyperparameter optimization (HPO) method needs to be computationally efficient to be useful in practice. Most of the existing approaches on multi-objective HPO use evolutionary strategies and metamodel-based optimization. However, few methods account for uncertainty in the performance measurements. This paper presents results on multi-objective HPO with uncertainty on the performance evaluations of the ML algorithms. We combine the sampling strategy of Tree-structured Parzen Estimators (TPE) with the metamodel obtained after training a Gaussian Process Regression (GPR) with heterogeneous noise. Experimental results on three analytical test functions and three ML problems show the improvement in the hypervolume obtained, when compared with HPO using stand-alone multi-objective TPE and GPR.

Keywords: hyperparameter optimization · multi-objective optimization · Bayesian optimization · uncertainty

1 Introduction

In Machine Learning (ML), an hyperparameter is a parameter that needs to be specified before training the algorithm: it influences the learning process, but it is not optimized as part of the training algorithm. The time needed to train a ML algorithm with a given hyperparameter configuration on a given dataset may already be substantial, particularly for moderate to large datasets,

* *Correspondence to*: Alejandro Morales-Hernández

so the HPO algorithm should be as efficient as possible in detecting the optimal hyperparameter setting.

Many of the current algorithms in the literature focus on optimizing a single (often error-based) objective [2,14,11]. In practical applications, however, it is often required to consider the trade-off between two or more objectives, such as the error-based performance of a model and its resource consumption [8], or objectives relating to different types of error-based performance measures [6]. The goal in multi-objective HPO is to obtain the *Pareto-optimal* solutions, i.e., those hyperparameter values for which none of the performance measures can be improved without negatively affecting any other.

In the literature, most HPO approaches take a deterministic perspective using the mean value of the performance observed in subsets of data (cross validation protocol). However, depending on the chosen sets, the outcome may differ: a single HP configuration may thus yield different results for each performance objective, implying that the objective contains uncertainty (hereafter referred to as *noise*). We conjecture that a HPO approach that considers this uncertainty will outperform alternative approaches that assume the relationships to be deterministic. Stochastic algorithms (such as [3,5]) can potentially be useful for problems with heterogeneous noise (the noise level varies from one setting to another). To the best of our knowledge, such approaches have not yet been studied in the context of HPO optimization. The main contributions of our approach include:

- Multi-objective optimization using a Gaussian Process Regression (GPR) surrogate that explicitly accounts for the heterogeneous noise observed in the performance of the ML algorithm.
- The selection of infill points according to the sampling strategy of multi-objective TPE (MOTPE), and the maximization of an infill criterion. This method allows sequential selection of hyperparameter configurations that are likely to be non-dominated, and that yield the largest expected improvement in the Pareto front.

The remainder of this article is organized as follows. Section 2 discusses the basics of GPR and MOTPE. Section 3 presents the algorithm. Section 4 describes the experimental setting designed to evaluate the proposed algorithm, and Section 5 shows the results. Finally, Section 6 summarizes the findings and highlights some future research directions.

2 GPR and TPE: Basics

Gaussian Process Regression (GPR) (also referred to as *kriging*, [16]) is commonly used to model an unknown target function. The function value prediction at an unsampled point $\mathbf{x}^{(*)}$ is obtained through the conditional probability $P(f(\mathbf{x}^{(*)})|\mathbf{X},\mathbf{Y})$ that represents how likely the response $f(\mathbf{x}^{(*)})$ is, given that we observed the target function at n input locations $\mathbf{x}^{(i)}, i = 1, \ldots, n$ (contained in matrix $\mathbf{X}$), yielding function values $\mathbf{y}^{(i)}, i = 1, \ldots, n$ (contained in matrix $\mathbf{Y}$)

that may or may not be affected by noise. Ankenman et al. [1] provides a GPR model (referred to as *stochastic kriging*) that takes into account the heterogeneous noise observed in the data, and models the observed response value in the r-th replication at design point $\mathbf{x}^{(i)}$ as:

$$f_r(\mathbf{x}^{(i)}) = m(\mathbf{x}^{(i)}) + M(\mathbf{x}^{(i)}) + \epsilon_r(\mathbf{x}^{(i)}) \tag{1}$$

where $m(\mathbf{x})$ represents the mean of the process, $M(\mathbf{x})$ is a realization of a Gaussian random field with mean zero (also referred to as the *extrinsic uncertainty* [1]), and $\epsilon_r(\mathbf{x}^{(i)})$ is the *intrinsic uncertainty* observed in replication r. Popular choices for $m(\mathbf{x})$ are $m(\mathbf{x}) = \sum_h \beta_h f_h(\mathbf{x})$ (where the $f_h(\mathbf{x})$ are known linear or nonlinear functions of $\mathbf{x}$, and the β_h are unknown coefficients to be estimated), $m(\mathbf{x}) = \beta_0$ (an unknown constant to be estimated), or $m(\mathbf{x}) = 0$. $M(\mathbf{x})$ can be seen as a function, randomly sampled from a space of functions that, by assumption, exhibit spatial correlation according to a covariance function (also referred to as *kernel*).

Whereas GPR models the probability distribution of $f(\mathbf{x})$ given a set of observed points ($P(f(\mathbf{x})|\mathbf{X}, \mathbf{Y})$), TPE tries to model the probability of sampling a point that is directly associated to the set of observed responses ($P(\mathbf{x}|\mathbf{X}, \mathbf{Y})$) [2]. TPE defines $P(\mathbf{x}|\mathbf{X}, \mathbf{Y})$ using two densities:

$$P(\mathbf{x}|\mathbf{X}, \mathbf{Y}) = \begin{cases} l(x) \ if & f(x) < y^*, \mathbf{x} \in \mathbf{X} \\ g(x) & o.w \end{cases} \tag{2}$$

where $l(x)$ is the density estimated using the points $\mathbf{x}^{(i)}$ for which $f(\mathbf{x}^{(i)}) < y^*$, and $g(x)$ is the density estimated using the remaining points. The value y^* is a user-defined quantile γ (splitting parameter of Algorithm 1 in [13]) of the observed $f(\mathbf{x})$ values, so that $P(f(\mathbf{x}) < y^*) = \gamma$. Here, we can see l as the density of the hyperparameter configurations that may have the best response. A multi-objective implementation of TPE (MOTPE) was proposed by [13]; this multi-objective version splits the known observations according to their nondomination rank. Contrary to GPR, neither TPE nor MOTPE provide an estimator of the response at unobserved hyperparameter configurations.

3 Proposed algorithm

The algorithm (Figure 1) starts by evaluating an initial set of hyperparameter vectors through a Latin hypercube sample; simulation replications are used to estimate the objective values at these points. We then perform two processes in parallel. On the one hand, we use the augmented Tchebycheff scalarization function [10] (with a random combination of weights) to transform the multiple objectives into a single objective using these training data. Throughout this article, we will assume that the individual objectives need to be minimized; hence, the resulting scalarized objective function also needs to be minimized. We then train a (single) stochastic GPR metamodel on these scalarized objective function outcomes; the replication outcomes are used to compute the variance of this scalarized objective.

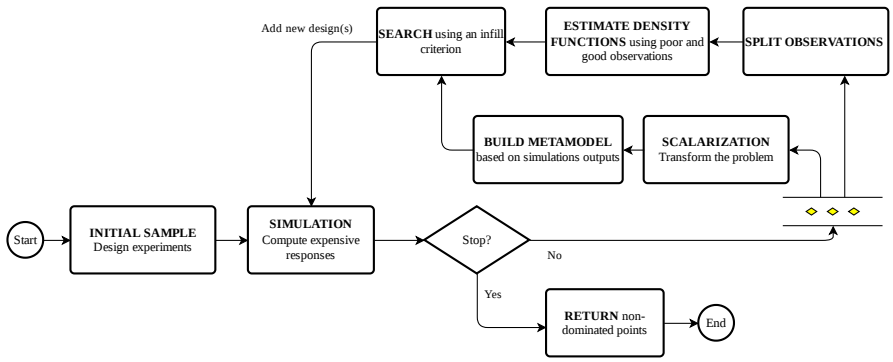


Fig. 1: Proposed multi-objective HPO using GPR with heterogeneous noise and TPE to sample the search space

At the same time, we perform the splitting process used by [13] to divide the hyperparameter vectors into two subsets (those yielding "good" and "poor" observations) to estimate the densities $l(x)$ and $g(x)$ for each separate input dimension (Eq. 2). To that end, our approach uses a greedy selection according to the nondomination rank of the observations, and controlled by the parameter γ [5]. The strategy thus preferably selects the HP configurations with highest nondomination rank to enter in the "good" subset.

Using the densities $l(x)$, we randomly select a candidate set of n_c configurations for each input dimension. These individual values are sorted according to their log-likelihood ratio $log\frac{l(x)}{g(x)}$, such that the higher this score, the larger the probability that the input value is sampled under $l(\mathbf{x}_i)$ (and/or the lower the probability under $g(\mathbf{x}_i)$). Instead of selecting the single configuration with highest score on each dimension (as in [2,13]), we compute the aggregated score $AS(\mathbf{x}) = \sum_{i=1}^{d} log\frac{l(x_i)}{g(x_i)}$ for each configuration, and select the one that maximizes the *Modified Expected Improvement* (MEI) [**?**] of the scalarized objective function in the set of configurations Q with an aggregated score greater than zero (see Eq. 3).

$$\arg\max_{\mathbf{q}\in Q} (\widehat{Z}_{\min} - \widehat{Z}_{\mathbf{q}})\Phi(\frac{\widehat{Z}_{\min} - \widehat{Z}_{\mathbf{q}}}{\widehat{s}_{\mathbf{q}}}) + \widehat{s}_{\mathbf{q}}\phi(\frac{\widehat{Z}_{\min} - \widehat{Z}_{\mathbf{q}}}{\widehat{s}_{\mathbf{q}}}) \;, Q = \{\mathbf{x} \,|\, AS(\mathbf{x}) > 0\} \quad (3)$$

where $\widehat{Z}_{\min}$ is the stochastic kriging prediction at $\mathbf{x}_{\min}$ (i.e. the hyperparameter configuration with the lowest sample mean among the already known configurations), $\phi(\cdot)$ and $\Phi(\cdot)$ are the standard normal density and standard normal distribution function respectively, the $\widehat{Z}_{\mathbf{q}}$ is the stochastic kriging prediction at configuration $\mathbf{q}$, and $\widehat{s}_{\mathbf{q}}$ is the ordinary kriging standard deviation for

[5] Notice that both in [13] and in our algorithm, the parameter γ represents a percentage of the known observations that may be considered as "good".

that configuration [17]. The search using MEI focuses on new points located in promising regions (i.e., with low predicted responses; recall that we assume that the scalarized objective need to be minimized), or in regions with high meta-model uncertainty (i.e., where little is known yet about the objective function). Consequently, the sampling behavior automatically trades off exploration and exploitation of the configuration search space.

Once a new hyperparameter configuration has been selected as infill point, the ML algorithm is trained on this configuration, yielding (again) noisy estimates of the performance measures. Following this infill strategy, we choose that configuration for which we expect the biggest improvement in the scalarized objective function, among the configurations that are likely to be non-dominated.

4 Numerical simulations

In this section, we evaluate the performance of the proposed algorithm for solving multi-objective optimization problems (GP_MOTPE), comparing the results with those that would be obtained by using GP modelling and MOTPE individually. In a first experiment, we analyze the performance on three well-known bi-objective problems (ZDT1, WFG4 and DTLZ7 with input dimension $d = 5$; see [7]), to which we add artificial heterogeneous noise (as in [5]). More specifically, we obtain noisy observations $\widetilde{f}_p^j(\mathbf{X}_i) = f_j(\mathbf{X}_i) + \epsilon_p(\mathbf{X}_i), p = \{1, \ldots, r\}, j = \{1, \ldots, m\}$, with $\epsilon_p(\mathbf{X}_i) \sim \mathcal{N}(0, \tau_j(\mathbf{X}_i))$. The standard deviation of the noise ($\tau_j(\mathbf{X})$) varies for each objective between $0.01 \times \Omega^j$ and $0.5 \times \Omega^j$, where Ω^j is the range of objective j. In between these limits, $\tau_j(\mathbf{X})$ decreases linearly with the objective value: $\tau_j(\mathbf{X}) = a_j(f_j(\mathbf{X}) + b_j), \forall j \in \{1, \ldots, m\}$, where a and b are the linear coefficients obtained from the noise range [9].

Table 1: Details of the ML datasets

Dataset	ID	Inst. (Feat.)	Dataset	ID	Inst. (Feat.)
Balance-scale	997	625 (4)	Delta_ailerons	803	7129 (5)
Optdigits	980	5620 (64)	Heart-statlog	53	270 (13)
Stock	841	950 (9)	Chscase_vine2	814	468 (2)
Pollen	871	6848 (5)	Ilpd	41945	583 (10)
Sylvine	41146	5124 (20)	Bodyfat	778	252 (14)
Wind	847	6574 (14)	Strikes	770	625 (6)

In a second experiment, we test the algorithm on a number of OpenML datasets, shown in Table 1. We optimize five hyperparameters for a simple (one hidden layer) Multi-Layer Perceptron (MLP), two for a support vector machine (SVM), and five for a Decision Tree (DT) (see Appendix A). In each experiment, the goal is to find the HPO configurations that minimize classification error while simultaneously maximizing recall. In all experiments, we used 20% of the initial dataset as test set, and the remainder for HPO. We apply stratified *k-fold cross-validation* ($k = 10$) to evaluate each hyperparameter configuration.

We used a fixed, small number of iterations (100) as a stopping criterion in all algorithms; this keeps optimization time low, and resembles real-world

optimization settings where limited resources (e.g., time) may exist. Table 2 summarizes the rest of the parameters used in the experiments.

Table 2: Summary of the parameters for the experiments

Setting	**Problem**	**GP**	**MOTPE**	**GP_MOTPE**
Initial design	Analytical fcts	LHS: $11d-1$		
	HPO	Random sampling: $11d-1$		
Replications	Analytical fcts	50		
	HPO	10		
Acquisition function		MEI	EI_{TPE}	MEI
Acquisition function optimization		**PSO***	Maximization on a candidate set	
Number of candidates to sample		-	$n_c = 1000$, $\gamma = 0.3$	
Kernel		Gaussian	-	Gaussian

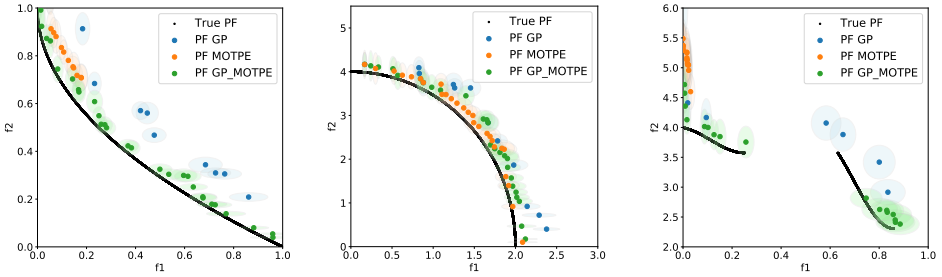
* PSO algorithm (Pyswarm library): swarm size = 300, max iterations = 1800, cognitive parameter=0.5, social parameter=0.3, and inertia=0.9

5 Results

Figure 3 shows the evolution of the hypervolume indicator during the optimization of the analytical test functions. The combined algorithm GP_MOTPE yields a big improvement over both GP and MOTPE algorithms for the ZDT1 and DTLZ7 functions, reaching a superior hypervolume already after a small number of iterations. Results also show that for ZDT1 and DTLZ7, the standard deviation on the final hypervolume obtained by GP and GP_MOTPE is small, which indicates that a Pareto front of similar quality is obtained regardless of the initial design. MOTPE, by contrast, shows higher uncertainty in the hypervolume results at the end of the optimization. For the concave Pareto front of WFG4, MOTPE provides the best results, while GP_MOTPE still outperforms GP.

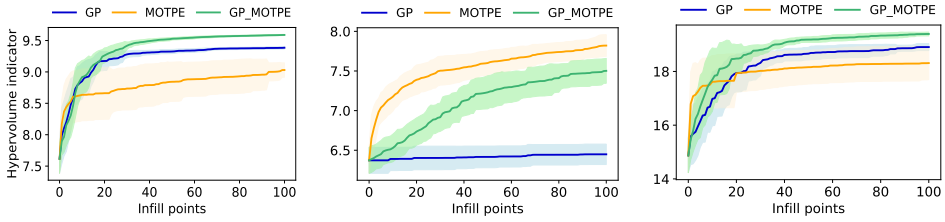
Table 3 shows the average rank of the optimization algorithms according to the hypervolume indicator. The experiments did not highlight significant differences between GP_MOTPE, GP and MOTPE ($p\_value = 0.565 > 0.05$ for the non-parametric Friedman test where H_0 states that the mean hypervolume of the solutions is equal). However, GP_MOTPE has the lowest average rank in the validation set, indicating that on average, the Pareto front obtained with our algorithm tends to outperform those found by GP and MOTPE individually, yielding a larger hypervolume.

Once the Pareto-optimal set of HP configurations has been obtained on the validation set, the ML algorithm (trained with those configurations) is evaluated on the test set. The difference between the hypervolume values obtained from the validation and test set can be used as a measure of reliability: in general, one would prefer HP configurations that generate a similar hypervolume in the test set. Figure 4 shows that the difference between both hypervolume values is almost zero when GP_MOTPE is used, for all ML algorithms. In general,



(a) ZDT1 (b) WFG4 (c) DTLZ7

Fig. 2: Observed Pareto front (PF) obtained at the end of a single macroreplication, for the analytical test functions. The uncertainty of each solution is shown by a shaded ellipse, and reflects the $mean \pm std$ of the simulation replications.



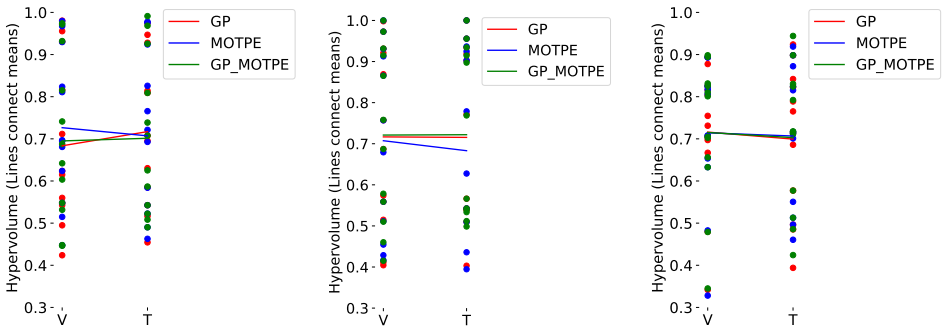
(a) ZDT1 $ref = [1, 10]$ (b) WFG4 $ref = [3, 5]$ (c) DTLZ7 $ref = [1, 23]$

Fig. 3: Hypervolume evolution during the optimization of the analytical test functions. Shaded area represents $mean \pm std$ of 13 macro-replications. Captions contain the reference point used to compute the hypervolume indicator

MOTPE and GP_MOTPE have the smallest (almost identical) mean absolute hypervolume difference (0.0444 and 0.0445 respectively), compared with that of GP (0.051). However, GP_MOTPE has the smallest standard deviation (0.054), followed by MOTPE (0.066) and GP (0.067).

Table 3: Average rank (given by the hypervolume indicator) of each algorithm

	Validation set			**Test set**		
	GP	**MOTPE**	**GP_MOTPE**	**GP**	**MOTPE**	**GP_MOTPE**
Avg. rank	2.125	1.9861	1.8889	2.1528	1.875	1.9722



(a) MLP (b) SVM (c) DT

Fig. 4: Hypervolume generated by the HP configurations found using the validations set (V) and then evaluated with the test set (T)

It is somehow surprising that the combined GP_MOTPE algorithm does not always obtain an improvement over the individual MOTPE and GP algorithms. By combining both approaches, we ensure that we select configurations that (1) have high probability to be nondominated (according to the candidate selection strategy), and (2) has the highest MEI value for the scalarized objective. In the individual GP algorithm, (1) is neglected, which increases the probability of sampling a non-Pareto optimal point, especially at the start of the algorithm. In the original MOTPE algorithm, (2) is neglected, which may cause the algorithm to focus too much on exploitation, which increases the probability of ending up in a local optimum. We suspect that the MOTPE approach for selecting candidate points may actually be too restrictive: it will favor candidate points close to already sampled locations, inherently limiting the exploration opportunities the algorithm still has when optimizing MEI.

6 Concluding remarks

In this paper, we proposed a new algorithm (GP_MOTPE) for multi-objective HPO of ML algorithms. This algorithm combines the predictor information (both predictor and predictor variance) obtained from a GPR model with heterogenous noise, and the sampling strategy performed by Multi-objective Tree-structured Parzen Estimators (MOTPE). In this way, the algorithm should select new points that are likely to be non-dominated, and that are expected to cause the maximum improvement in the scalarized objective function.

The experiments conducted report that our approach performed relatively well for the analytical test functions of study. It appears to outperform the pure GP algorithm in all analytical instances; yet, it does not always outperform the original MOTPE algorithm. Further research will focus on why this is the case, which may yield further improvements in the algorithm. In the HPO experiments, GP_MOTPE shows the best average rank w.r.t. the hypervolume computed on the validation set. In addition, it showed promising reliability properties (small changes in hypervolume when the ML algorithm is evaluated on the test set). Based upon these first results, we believe that the combination of GP and TPE is promising enough to warrant further research. The observation that it outperforms the pure GP algorithm (which used PSO to maximize the infill criterion) is useful in its own right, as the optimization of infill criteria is known to be challenging. Using MOTPE, a candidate set can be generated that can be evaluated efficiently, and which (from these first results) appears to yield superior results.

Acknowledgements

This research was supported by the Flanders Artificial Intelligence Research Program (FLAIR).

Appendix 1. Setup of hyperparameters in the HPO experiments

HP	Description	Type	Range
Multilayer Perceptron (MLP)			
max_iter	Iterations to optimize weights	Int.	$[1, 1000]$
neurons	Number of neurons in the hidden layer	Int.	$[5, 1000]$
lr_init	Initial learning rate	Int.	$[1, 6]$
b1	First exponential decay rate	Real	$[10^{-7}, 1]$
b2	Second exponential decay rate	Real	$[10^{-7}, 1]$
Support Vector Machine (SVM)			
C	Regularization parameter	Real	$[0.1, 2]$
kernel	Kernel type to be used in the algorithm	Cat.	[linear, poly, rbf, sigmoid]
Decision Tree (DT)			

Continued on next page

HP	Description	Type	Range
max_depth	Maximum depth of the tree. If 0, then *None* is used	Int.	$[0, 20]$
mss	Minimum number of samples required to split an internal node	Real	$[0, 0.99]$
msl	Minimum number of samples required to be at a leaf node	Int.	$[1, 10]$
max_f	Features in the best split	Cat.	[auto, sqrt, log2]
criterion	Measure the quality of a split	Cat.	[gini, entropy]

References

1. Ankenman, B., Nelson, B.L., Staum, J.: Stochastic kriging for simulation metamodeling. Operations Research **58**(2), 371–382 (2010). https://doi.org/10.1109/WSC.2008.4736089
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in neural information processing systems **24** (2011)
3. Binois, M., Huang, J., Gramacy, R.B., Ludkovski, M.: Replication or exploration? sequential design for stochastic simulation experiments. Technometrics **61**(1), 7–23 (2019). https://doi.org/10.1080/00401706.2018.1469433
4. Bischl, B., Mersmann, O., Trautmann, H., Weihs, C.: Resampling methods for meta-model validation with recommendations for evolutionary computation. Evolutionary Computation **20**(2), 249–275 (2012). https://doi.org/10.1162/EVCO_a_00069
5. Gonzalez, S.R., Jalali, H., Van Nieuwenhuyse, I.: A multiobjective stochastic simulation optimization algorithm. European Journal of Operational Research **284**(1), 212–226 (2020). https://doi.org/10.1016/j.ejor.2019.12.014
6. Horn, D., Bischl, B.: Multi-objective parameter configuration of machine learning algorithms using model-based optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8. IEEE (2016). https://doi.org/10.1109/SSCI.2016.7850221
7. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10**(5), 477–506 (2006)
8. Igel, C.: Multi-objective model selection for support vector machines. In: International conference on evolutionary multi-criterion optimization. pp. 534–546. Springer (2005). https://doi.org/10.1007/978-3-540-31880-4_37
9. Jalali, H., Van Nieuwenhuyse, I., Picheny, V.: Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. European Journal of Operational Research **261**(1), 279–301 (2017)
10. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**(1), 50–66 (2006). https://doi.org/10.1109/TEVC.2005.851274
11. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research **18**(1), 6765–6816 (2017)

12. Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Marben, J., Müller, P., Hutter, F.: Boah: A tool suite for multi-fidelity bayesian optimization & analysis of hyperparameters. arXiv:1908.06756 [cs.LG]
13. Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M.: Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 533–541 (2020)
14. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
15. Viana, F.A.: Things you wanted to know about the latin hypercube design and were afraid to ask. In: 10th World Congress on Structural and Multidisciplinary Optimization. vol. 19. sn (2013)
16. Williams, C.K., Rasmussen, C.E.: Gaussian processes for machine learning, vol. 2. MIT press Cambridge, MA (2006)
17. Zhan, D., Xing, H.: Expected improvement for expensive optimization: a review. Journal of Global Optimization **78**(3), 507–544 (2020). https://doi.org/10.1007/s10898-020-00923-x