Artificial Intelligence Review (2023) 56:8043–8093
https://doi.org/10.1007/s10462-022-10359-2

A survey on multi-objective hyperparameter optimization algorithms for machine learning

Alejandro Morales-Hernández[1,2] · **Inneke Van Nieuwenhuyse**[1,2] · **Sebastian Rojas Gonzalez**[1,2,3]

Published online: 24 December 2022

Abstract
Hyperparameter optimization (HPO) is a necessary step to ensure the best possible performance of Machine Learning (ML) algorithms. Several methods have been developed to perform HPO; most of these are focused on optimizing one performance measure (usually an error-based measure), and the literature on such single-objective HPO problems is vast. Recently, though, algorithms have appeared that focus on optimizing multiple conflicting objectives simultaneously. This article presents a systematic survey of the literature published between 2014 and 2020 on multi-objective HPO algorithms, distinguishing between metaheuristic-based algorithms, metamodel-based algorithms and approaches using a mixture of both. We also discuss the quality metrics used to compare multi-objective HPO procedures and present future research directions.

Keywords Hyperparameter optimization · Multi-objective optimization · Metamodel · Meta-heuristic · Machine learning

1 Introduction

Nowadays, Artificial Intelligence (AI) is omnipresent in everyday life. Current technological advances allow us to analyze huge amounts of data to generate knowledge that is used in many different ways, e.g. for automatic user recommendations (Cai et al. 2020), image recognition (Phillips et al. 2005; Andreopoulos and Tsotsos 2013), and supporting healthcare-related tasks (Jiang et al. 2017). In general, AI can be seen as a computer technology

✉ Alejandro Morales-Hernández
alejandro.moraleshernandez@uhasselt.be

Inneke Van Nieuwenhuyse
inneke.vannieuwenhuyse@uhasselt.be

Sebastian Rojas Gonzalez
sebastian.rojasgonzalez@uhasselt.be

[1] Faculty of Sciences, Hasselt University, Hasselt, Belgium

[2] VCCM Core Lab and Data Science Institute, Hasselt University, Hasselt, Belgium

[3] Surrogate Modeling Lab, Ghent University, Ghent, Belgium

capable of carrying out functions that traditionally required human intelligence (Ertel 2018). Although learning is a key element in many areas of artificial intelligence, the very concept of learning is mainly studied in the Machine Learning (ML) subfield. According to Mitchell (1997), "a computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**". ML algorithms and their parameters must be intelligently configured to make the most of the data. Those parameters that need to be specified *before* training the algorithm are usually referred to as *hyperparameters*: they influence the learning process, but they are not optimized as part of the training algorithm.

The impact of these hyperparameters on algorithm performance should not be underestimated (Kim et al. 2017; Kong et al. 2017; Singh et al. 2020; Cooney et al. 2020); yet, their optimization (hereafter referred to as *hyperparameter optimization* or HPO) is a challenging task, as traditional optimization methods are often not applicable (Luo 2016). Indeed, classic convex optimization methods such as gradient descent tend to be ill-suited for HPO, as the measure to optimize is usually a non-convex and non-differentiable function (Stamoulis et al. 2018; Parsa et al. 2019). Furthermore, the hyperparameters to optimize may be discrete, categorical and/or continuous (typical hyperparameters for an Artificial Neural Network (ANN), for instance, are the number of layers, the number of neurons per layer, the type of optimizer, and the learning rate). The search space can also contain *conditional hyperparameters*; e.g., the hyperparameters in a support vector machine algorithm depend on the type of kernel used. Finally, the time needed to train a machine learning model with a given hyperparameter configuration on a *given* dataset may already be substantial, particularly for moderate to large datasets; as a common HPO algorithm requires multiple such training cycles, the algorithm itself needs to be computationally efficient to be useful in practice.

HPO should not be confused with the more general topic of *automatic algorithm configuration (AC)*, which is much broader in scope (see López-Ibáñez et al. 2016; Hutter et al. 2009 for examples on this topic). In AC, in general, the aim is to find a well-performing parameter configuration for an arbitrary algorithm on a given, finite set of problem instances. In HPO, we typically search for a well-performing hyperparameter configuration on a *single* data set, for a specific task (classification, image recognition, or other). The scope of AC is also broader than that of HPO, in the sense that the target algorithm does not necessarily carry out a learning process for the task under study; e.g., it also comprises the optimization of solvers and/or metaheuristics.

HPO has gained increasing attention in recent years, probably spurred by the popularity of deep learning algorithms, which have demanding characteristics (e.g., the need for large amounts of data and time to train the models, high model complexity, and a diverse mix of hyperparameter types). Previously, analysts tended to use simple methods to look for the "best" hyperparameter settings. The most basic of these is grid search (Montgomery 2017): the user creates a set of possible values for each hyperparameter, and the search evaluates the Cartesian product of these sets. Although this strategy is easy to implement and easy to understand, its performance is influenced by the number of hyperparameters to optimize, and the (number of) values chosen on the grid. Random search (Bergstra and Bengio 2012) provides an alternative to grid search, and tends to be popular when some of the hyperparameters are more important than others; e.g, learning rate and momentum are critical to guarantee a faster convergence of neural networks (Guo et al. 2020). More advanced optimization methods have also been put forward, such as meta-learning methods (Bui and Yi 2020), neural architecture search (NAS) methods (Jing et al. 2020), multi-fidelity algorithms (such as Freeze-thaw Bayesian optimization (Swersky et al. 2014), Successive

halving algorithm (Karnin et al. 2013), Hyperband (Li et al. 2017), Bayesian Optimization Hyperband (Falkner et al. 2018), and Multi-task Bayesian optimization (Swersky et al. 2013)), population-based optimization algorithms (such as Population-based training (PBT) (Jaderberg et al. 2017) and Population-based Bandits (PB2) (Parker-Holder et al. 2020)), and reinforcement learning algorithms (such as HypRL (Jomaa et al. 2019)) and the model-based Reinforcement Learning algorithm (Wu et al. 2020)).

So far, these more advanced approaches have largely focused on *single-objective* HPO problems. *Multi-objective* optimization is particularly relevant in HPO, as different conflicting objectives may be important for the analyst (e.g., the error-based performance of the target ML algorithm, inference time, model size, energy consumption, etc.). Multi-objective HPO should not be confused with multi-task learning (MTL). In multi-objective HPO, we seek to optimize the hyperparameter configuration for a specific task, on a single data set, in view of marrying multiple conflicting objectives. MTL, by contrast, seeks to optimize the HP configuration for multiple tasks, potentially using multiple datasets; while the performance metrics for the individual tasks can be seen as multiple simultaneous objectives, they are not necessarily in conflict.

Our work aims to provide an overview of the state-of-the-art in the field of *multi-objective hyperparameter optimization* for machine learning algorithms, highlighting the approaches currently used in the literature, the typical performance measures used as objectives, and discussing remaining challenges in the field. To the best of our knowledge, our work presents the first comprehensive review of these multi-objective HPO approaches. Previous reviews (Hutter et al. 2015; Luo 2016; Yang and Shami 2020; Feurer and Hutter 2019; Talbi 2021) mainly discuss single-objective HPO approaches, often focusing on particular contexts (such as biomedical data analysis), specific target algorithms (such as Deep Neural Networks) or specific approaches (Sequential Model-based Bayesian Optimization, multi-fidelity approaches). While two of the most recent surveys (Feurer and Hutter 2019; Talbi 2021) mention multi-objective HPO on the sidelines, they only list some examples or common strategies relevant to this topic, without discussing the actual approaches.

The remainder of this article is organized as follows. Section 2 discusses the methodology used in the literature search. Section 3 formalizes the concepts of single- and multi-objective hyperparameter optimization and discusses the most commonly used performance measures in HPO algorithms. Section 4 categorizes the existing methods for multi-objective hyperparameter optimization. Section 5 discusses the pros and cons of the algorithms. Finally, Sect. 6 summarizes the findings, highlighting potential improvements and avenues for further research.

2 Methodology

Given the remarkable surge in publications on HPO since 2014, we focused on research published between 2014 and 2020. Figure 1 shows an overview of the search and selection process.

We first performed a WoS (Web of Science) search, using the search terms shown in Table 1. Although the main focus is on multi-objective HPO, we also consider the occurrence of the phrase "single objective" in the abstract (AB), as it is common to transform multiple objectives into a single objective by means of a scalarization function. As the use of surrogates is common in single-objective HPO for deep learning networks (e.g., Wistuba et al. 2018; Sjöberg 2019; Victoria and Maragatham 2021), we also searched for

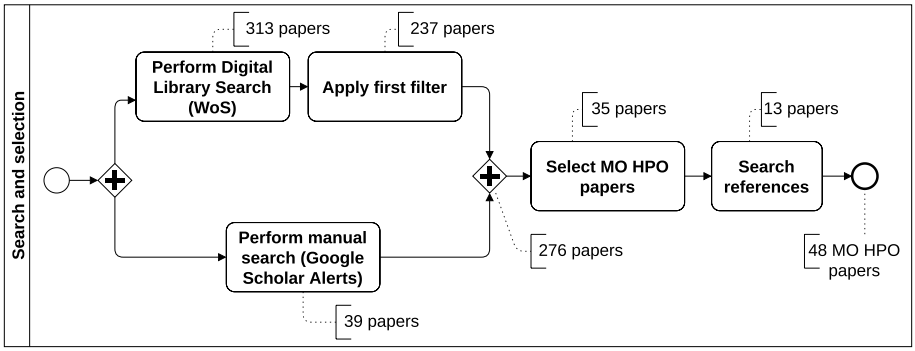
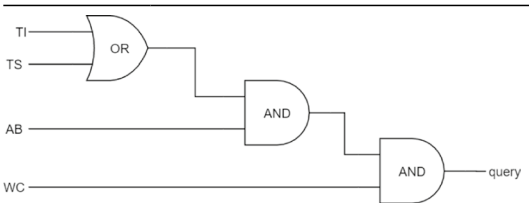


Fig. 1 Overview of search and selection process

Table 1 Search term details



TI	Hyperparameter optimization, hyperparameter tuning, parameter optimization, Hyperparameter, parameter tuning
TS	Hyperparameter optimization, hyperparameter tuning
AB	Neural networks, deep learning, constraint, overfitting, multiobjective, Multi objective, multi-objective, many-objective, many objective, single objective, Surrogate, metamodel, gaussian process, kriging
WC	Computer Science, Artificial Intelligence Computer Science, Information Systems Computer Science, Theory & Methods Computer Science, Interdisciplinary Applications

TI Title, *TS* Topic, *AB* Abstract, *WC* Web of Science Categories

articles mentioning the terms "surrogate", "metamodel", "deep learning", "neural networks", "Gaussian process", and "kriging" in the abstract. The choice of hyperparameters is also related to overfitting (Feurer and Hutter 2019). Finally, we also include the term "constraint", as the required performance targets (e.g., maximum memory consumption, training time Stamoulis et al. 2018; Hu et al. 2019) may be presented as constraints in (multi-objective) HPO. We limited our search to publications (including conference proceedings, articles, book chapters, and meeting abstracts) in computer science-related categories (WC).

We subsequently completed the set of papers through (1) scanning suggestions of papers on Google Scholar alerts, and (2) a reference search. We limited the latter to electronic collections only, and solely considered journals/conference proceedings/workshop

proceedings that were indexed on WoS (for the WoS journals, we included accepted preprints of forthcoming articles).

The papers obtained through the WoS and manual search were manually filtered based on the title and abstract, to ensure they were related to the topic of discussion. We filtered out irrelevant papers, such as those that focus on the optimization of industrial processes (Chen et al. 2014), meta-learning (Vanschoren 2019), optimization of internal parameters (Wawrzyński 2017), and papers related to AutoML systems that are not focused on hyperparameter optimization (such as model selection algorithms (van Rijn et al. 2015; Silva et al. 2016) or pure feature selection methods (Hegde and Mundada 2020)). Neural Architecture Search (NAS) is usually considered a distinct category with its own methods and techniques for optimizing the structure of a neural network; hence, articles on NAS were only considered when the problem was addressed as an HPO problem. Articles focusing on more specific aspects of NAS (such as Negrinho et al. 2019) are beyond the scope of this research.

A full read of the articles, combined with a reference search, resulted in a final selection of 48 relevant articles. Most of these articles (about 60%) were published in conferences or workshops, though there has been an increase in scientific journal articles in 2020 (see Fig. 2); these were mainly published in Q1/Q2 journals belonging to the Computer Science field.

3 HPO: concepts and performance measures

Section 3.1 provides an overview of the basic concepts related to HPO, while Sect. 3.2 discusses the main performance measures (objectives) used in such optimization. Finally, Sect. 3.3 discusses the quality metrics used for comparing the performance of multi-objective HPO algorithms.

3.1 HPO: concepts and terminology

In mathematics and computer science, an algorithm is a finite sequence of well-defined instructions that, when fed with a set of initial inputs, eventually produces an output. Figure 3 shows that in HPO, the optimization algorithm forms an "outer" shell of optimization instructions; the "inner" optimization refers to the training and cross-validation of the target ML algorithm (e.g., ANN, SVM, etc.). This inner optimization trains the target algorithm to perform the task it should perform (e.g., predicting house prices from a data set,

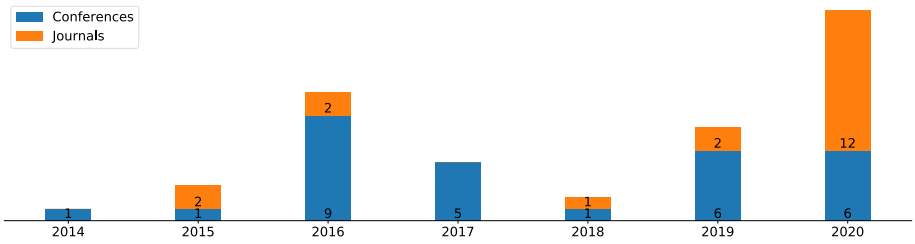


Fig. 2 Number of articles that address multi-objective HPO, according to the publication source (2014–2020)

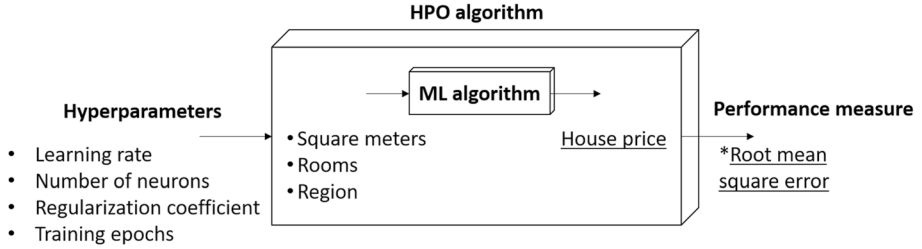


Fig. 3 Example of the interplay between the HPO algorithm and the target ML algorithm (in this case, an ANN for predicting house prices)

using a set of features). In turn, the HPO algorithm takes the hyperparameters of the target ML algorithm as input and produces a number of performance measures as output (e.g., RMSE, energy consumption, etc.). The aim of the HPO algorithm is to optimize the set of hyperparameters, in view of obtaining the best possible outcomes for the performance measures considered.

More formally, the single-objective HPO problem can be formalized as follows. Consider a target ML algorithm $\mathcal{A}$ with N hyperparameters, such that the n-th hyperparameter has a domain denoted by Λ_n. The overall *hyperparameter configuration space* is denoted as $\Lambda = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_N$. A vector of hyperparameters is denoted by $\boldsymbol{\lambda} \in \Lambda$, and an algorithm $\mathcal{A}$ with its hyperparameters set to $\boldsymbol{\lambda}$ is denoted by $\mathcal{A}_{\boldsymbol{\lambda}}$. In the case of HPO, the available data are split into a training set, a validation set, and a test set. The *learning* process of the algorithm takes place on the training set ($\mathcal{D}_{train}$) and is validated on the validation set ($\mathcal{D}_{valid}$). We can then formalize the *single-objective* HPO problem as (Feurer and Hutter 2019):

$$\min_{\boldsymbol{\lambda} \in \Lambda} V(\mathcal{L} \mid \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

where $V(\mathcal{L} \mid \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ is a validation protocol that uses a loss function $\mathcal{L}$ to estimate the performance of a model $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and validated on $\mathcal{D}_{valid}$. Popular choices for the validation protocol $V(\cdot)$ are the holdout and cross-validation process (see Bischl et al. 2012 for an overview of validation protocols). Without loss of generality, we assume in the remainder of this article that the loss function should be minimized.

The previous definition can be readily extended to multi-objective optimization (see Li and Yao 2019). Consider a multi-objective hyperparameter optimization problem with N hyperparameters and a set $\mathbf{L}$ containing m performance measures (objective functions). These can reflect the error-based performance of the algorithm, but also other metrics such as algorithm complexity (as detailed later in Sect. 3.2). The multi-objective HPO problem can then be formalized as follows (assuming that all performance measures should be minimized):

$$\min_{\boldsymbol{\lambda} \in \Lambda} V(\mathbf{L} \mid \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Typically, there is a trade-off among the different objectives: for instance, between the performance of a model and training time (increasing the accuracy of a model often requires larger amounts of data and, hence, a higher training time; see e.g., Rajagopal et al. 2020), or between different error-based measures (e.g., between confusion matrix-based measures (Tharwat 2020) of a binary classification problem; see Horn and Bischl 2016). Considering

these trade-offs is often crucial: e.g., in medical diagnostics (de Toro et al. 2002), the simultaneous consideration of objectives such as sensitivity and specificity is essential to determine if the machine learning model can be used in practice. The goal in multi-objective HPO is to obtain the *Pareto-optimal* solutions, i.e., those solutions for which none of the objectives can be improved without negatively affecting any other objective. In the decision space, the set of optimal solutions is referred to as the *Pareto set*; in objective space, it yields the *Pareto front* (or Pareto frontier). The Pareto-optimal solutions are also referred to as the *non-dominated* solutions (Emmerich and Deutz 2018). Ideally, these solutions should be *diverse* (i.e., spread across the different areas of the Pareto front), while approximating this front as well as possible (i.e., showing *convergence* to the Pareto front).

In (general) multi-objective optimization problems, the multiple objectives are often *scalarized* into one single function, such that the problem can be solved as a single-objective problem. Care should be taken, though, when selecting the scalarization approach: e.g., not all approaches allow to detect non-convex parts of the front (see Miettinen and Mäkelä 2002 for further details about scalarization functions). Scalarization methods have also been applied in multi-objective HPO; see Section 4 for further details.

3.2 Multi-objective HPO: typical objectives

Tables 2 and 3 show an overview and concise description of the performance measures occurring in the current literature on multi-objective HPO (Table 2 focuses on error-based measures, while Table 3 summarizes the non-error-based measures). These measures will reappear later in Section 4, when we categorize the different multi-objective HPO algorithms. As evident from Table 2, for regression problems, the error-based metrics are commonly based on the squared errors; for classification problems, they are commonly related to the elements of the confusion matrix [True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN)].

Error-based measures are heavily used in multi-objective HPO, as they ensure a response from the model that is close to reality. Additionally, model complexity objectives are often included [following Occam's razor principle; (Blumer et al. 1987)], along with time-based metrics (e.g., training time on embedded devices) and/or (computational) cost objectives. The complexity of a neural network, for instance, is often estimated using the number of parameters (weights of the connections between neurons) (Liang et al. 2019; Lu et al. 2020; Baldeon and Lai-Yuen 2020; Calisto and Lai-Yuen 2020). The number of features can also be used as a complexity measure: see Sopov and Ivanov (2015), Martinez-de Pison et al. (2017), Binder et al. (2020), Faris et al. (2020), Bouraoui et al. (2018). The more features the training algorithm has to consider, the more expensive it will be. On the other hand, considering fewer features may negatively affect the error-based performance of the algorithm.

Metrics reflecting model size naturally depend on the target ML algorithm to be optimized (e.g., the number of neurons in a single-layer NN (Juang and Hsu 2014), the number of support vectors in a SVM (Bouraoui et al. 2018), the DNN file size (Shinozaki et al. 2020), or the number of models used (Garrido and Hernández 2019) for ensemble algorithms). Alternatively, the number of floating point operations (FLOPs) in a NN can be used (Wang et al. 2019, 2020; Lu et al. 2020; Chin et al. 2020; Loni et al. 2020). This metric is also used to reflect the energy consumption (Han et al. 2015); likewise, the number of parameters in a NN is used as a measure for complexity as well as for model size. Both FLOPs and the number of parameters are sometimes used as memory consumption

Table 2 Error-based measures used in multi-objective HPO algorithms

Type of problem	Performance measure	Description
Classification	Classification error	$\frac{FN+FP}{P+N}$
	Recall/Sensitivity	$\frac{TP}{TP+FN}$
	Precision	$\frac{TP}{TP+FP}$
	Specificity	$\frac{TN}{TN+FP}$
	False positive rate (FPR)	$\frac{FP}{FP+TN}$
	False negative rate (FNR)	$\frac{FN}{FN+TP}$
	Reward of spiking trace	Custom measure defined for Spiking Neural Networks to measure how good was the network categorizing looming and non-looming stimuli at a given time
Regression	Root mean square error (RMSE)	$\sqrt{\frac{\sum (Real-Prediction)^2}{Total\ of\ observations}}$
	Mean square error (MSE)	$\frac{\sum (Real-Prediction)^2}{Total\ of\ observations}$
	Sum square error (SSE)	$\sum (Real - Prediction)^2$
Speech recognition	Word error rate (WER)	Measures how different the recognized word is from the reference word
Image recognition	Segmentation accuracy	Computed as two times the area of overlap between two images divided by the total number of pixels in both images
	Mutual Information	Measures the dependencies between two images, or the amount of information that one image contains about the other

The description given for the classification metrics assumes a two-class problem

Table 3 Non error-based performance measures used in multi-objective HPO algorithms

Type	Performance measure
Complexity	Number of floating points operations (FLOPs) or number of multiply-adds (MAdds) in NNs
	Number of features used to train the ML algorithm
Model size	Number of parameters (weights) in a NN
	Number of neurons in NNs
	Number of support vectors in SVM
	The file size used to save a DNN
	Number of models used in an ensemble
Time	Related to the target ML algorithm (Training and Prediction time, inference time on forward passes of ANN, decoding time), or the optimization
Hardware-based measures	Memory footprint
	Energy consumption
Other	Diversity measures in ensembles.
	Outliers detected by a threshold-based algorithm.

measures (Laskaridis et al. 2020), and can be combined with a time-based measure (Shah and Ghahramani 2016). Time-based measures can be related to the training phase (Tanaka et al. 2016; Rajagopal et al. 2020; Laskaridis et al. 2020; Lu et al. 2020), the prediction phase (Hernández et al. 2016; Abdolsh et al. 2019; Garrido and Hernández 2019), the inference process on forwarding passes in ANNs (Kim et al. 2017), or the whole optimization process (Richter et al. 2016).

The increasing computational cost of Deep Learning models generally translates into higher hardware costs. As a result, optimization using both algorithm performance and hardware cost should be considered, especially for edge devices. Hardware-related costs can be measured in different ways; e.g., through energy consumption (Hernández-Lobato et al. 2016) or memory utilization (Chandra and Lane 2016). In many cases, these measures are estimated as a function of the hyperparameters. For instance, Parsa et al. (2019) present an abstract energy consumption model that depends on the neural network architecture (number of layers, number of outputs of each layer, kernel size, etc).

Some objectives encountered in the literature do not fall into any of the categories above. In Table 3, they are grouped into the category "Other" (e.g., diversity measures for ensembles (Kuncheva 2014)).

3.3 Quality metrics for comparing multi-objective HPO algorithms

The surveyed literature presents different metrics to judge and/or compare the strengths and weaknesses of multi-objective HPO algorithms. The first set of quality metrics is related to the resulting Pareto front. Here, hypervolume is the most widely used (Horn and Bischl 2016; Hernández et al. 2016; Shah and Ghahramani 2016; Horn et al. 2017; Garrido and Hernández 2019; Lu et al. 2020). It computes the volume of the area enclosed by the Pareto front and a reference point, specified by the user. Binder et al. (2020) compute the *generalization* dominated hypervolume, which is obtained by evaluating the non-dominated solutions of the validation set on the test set data. Other quality metrics based on the Pareto front are the difference in performance between

each solution on the front and the single-objective version of the algorithm (holding the other objectives steady) (Chatelain et al. 2007), the average distance (or Generational Distance) of the front to a reference set (such as the approximated true Pareto front obtained by exhaustive search, see Smithson et al. 2016; or an aggregated front, see Gülcü and Kuş 2021), a coverage measure computed as the percentage of the solutions of an algorithm *A* dominated by the solutions of another algorithm *B* (Juang and Hsu 2014; Li et al. 2004), or metrics based on the shape of the Pareto front (Abdolsh et al. 2019) or its diversity (Juang and Hsu 2014; Li et al. 2004). The latter can be computed using the spacing and the spread of the solutions: spacing evaluates the diversity of the Pareto points along a given front (Gülcü and Kuş 2021), whereas spread evaluates the range of the objective function values (see Zitzler et al. 2000).

Some authors use performance measures that do not relate to the quality of the front obtained; e.g., execution time (Parsa et al. 2019; Richter et al. 2016; Horn et al. 2017), number of performance evaluations (Parsa et al. 2019), CPU utilization in parallel computer architectures (Richter et al. 2016), measures that were not considered as an objective and that are evaluated in the Pareto solutions (usually, confusion matrix-based measures for classification problems; see Salt et al. 2019), or measures that are specific for the HPO algorithm used (e.g., the number of new points suggested per batch is used by Gupta et al. (2018) to evaluate the performance of the search executed during batch Bayesian optimization).

4 Multi-objective HPO algorithms: categorization

In this section, we categorize the literature on multi-objective HPO algorithms based on the way in which the algorithms perform the search for the optimal solutions (i.e., the search methodology). We distinguish the following three categories (Fig. 4):

- Metaheuristic-based optimization algorithms (Sect. 4.1): these algorithms use a metaheuristic to guide the search process, based on the empirically observed input/output observations.
- Metamodel-based optimization algorithms (Sect. 4.2): in these algorithms, a *metamodel* is fit to the empirical input/output observations, and an acquisition function is used to search for the optimal HPO configurations.
- Hybrid algorithms (Sect. 4.3): a metamodel is fit to the input/output observations, and a metaheuristic is used to guide the search for better solutions.

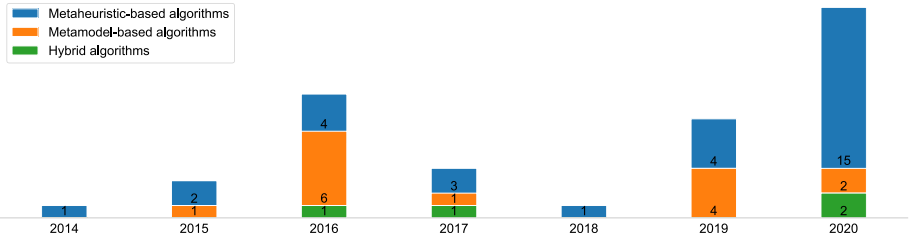


Fig. 4 Multi-objective HPO algorithms: number of articles per category (2014–2020)

4.1 Metaheuristic-based HPO algorithms

Heuristic search attempts to optimize a problem by improving the solution based on a given heuristic function or a cost measure (Russell and Norvig 2010). A heuristic search method does not always guarantee to find the optimal solution but aims to find a good or acceptable solution within a reasonable amount of time and memory usage. Metaheuristics are algorithms that combine heuristics (which are often problem-specific) in a more general framework (Bianchi et al. 2009). Figure 5 summarizes the general procedure of a metaheuristic-based algorithm for multi-objective optimization (MOO). The algorithm generates new solution(s) starting from one or more initial solution(s). Depending on the algorithm, the information available from the search process so far (which can include updates in the sampling distribution used by the metaheuristic, *or* other adjustments such as updates in the velocity vectors in Particle Swarm Optimization, or the pheromone paths in Ant Colony Optimization) can be updated before the next iteration starts, and/or bad solutions can be discarded. The process is repeated until a stop criterion is met.

While some metaheuristics start from a single initial solution (e.g., Tabu Search (Glover 1986)), others (referred to as population-based algorithms) start from a set of solutions (e.g., Ant Colony Optimization (Dorigo and Blum 2005) and Evolutionary Algorithms, e.g. Evolution Strategies and Genetic Algorithms (Mitchell 1998)).

For ease of reference, Table 4 gives an overview of the metaheuristic-based algorithms currently used in multi-objective HPO, while Table 5 gives an overview of the experimental comparisons reported in these papers. Clearly, the most popular metaheuristic-based algorithm for multi-objective HPO is the Non-dominated Sorting Genetic Algorithm II (NSGA-II; Deb et al. 2002). This is not surprising, as genetic algorithms have shown to perform quite well in single-objective HPO settings: see, e.g., Deighan et al. (2021), who showed that they cannot only obtain CNN configurations from scratch but can also refine state-of-the-art CNNs. NSGA-II builds on the original NSGA algorithm (Srinivas and Deb 1994); yet, it is computationally less expensive (a temporal complexity of $O(MN^2)$ versus $O(MN^3)$ for the original algorithm, where M is the number of objectives and N is the population size). Another important difference is the preservation of the best solutions, through an elitist selection according to the fitness and spread of solutions. Ekbal and Saha (2015) applied NSGA-II to jointly optimize hyperparameters and features, and demonstrated the superiority of the resulting models over others (trained with default hyperparameters, and using all the features included in

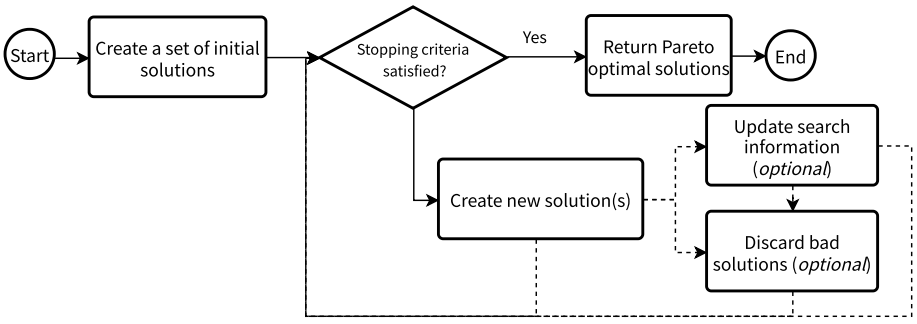


Fig. 5 General procedure in metaheuristic-based MOO algorithms

Table 4 Overview of Metaheuristic-based HPO algorithms

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measures						Application field
				Error	Complexity	Model size	Time	Hardware-based	Other	
NSGA-II	Ekbal and Saha (2015)	**N:** 1, **D:** 1, **C:** -	CRF	Recall Precision	–	–	–	–	–	Mention recognition in texts
		N: 2, **D:** 2, **C:** 1	SVM							
	Magda et al. (2017)	**N:** -, **D:** 2, **C:** -	Komi threshold algorithm	Mean error	–	–	–	–	Number of outliers	Muscle onset detection
	Mostafa et al. (2020)	**N:** 3, **D:** 2, **C:** -	CNN	Accuracy Recall Specificity	–	–	–	–	–	Apnea detection
	Sopov and Ivanov (2015)	**N:** -, **D:** 2, **C:** -	MLP	Classification rate	–	Number of neurons	–	–	–	Emotion recognition
	Binder et al. (2020)	**N:** -, **D:** 2, **C:** -	SVM	Generalization error	Fraction of selected features	–	–	–	–	OpenML benchmark
		N: 5, **D:** 4, **C:** -	XGBoost							
		N: -, **D:** 2, **C:** 1	kkNN							
	Shinozaki et al. (2020)	**N:** 7, **D:** 3, **C:** 1	Spoken Language Systems	Word error rate	–	DNN file size	–	–	–	Speech recognition
	Kim et al. (2017)	**N:** 2, **D:** 1, **C:** -	LeNet	Accuracy	–	–	Inference time	–	–	Image recognition

Table 4 (continued)

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measures						Application field
				Error	Complexity	Model size	Time	Hardware-based	Other	
	Bouraoui et al. (2018)	**N:** 4, **D:** 1, **C:** -	SVM	Accuracy	Number of features	Number of SV	–	–	–	UCI datasets
	Nabil et al. (2019)	**N:** 2, **D:** 2, **C:** 4	GRU-based RNN	Accuracy FPR	–	–	–	–	–	Electricity Theft Detection
	Loni et al. (2020)	**N:** -, **D:** 3, **C:** 3	CNN	Accuracy	Number of parameters		–	–	–	Image recognition
GA (scalarized objectives)	Deighan et al. (2021)	**N:** 4, **D:** 6, **C:** -	CNN	Accuracy	Number of parameters		–	–	–	Gravitational wave classification
MOEA/D (scalarized objectives)	Calisto and Lai-Yuen (2020)	**N:** 1, **D:** 3, **C:** 3	AdaEn-net	Segmentation accuracy	Number of parameters		–	–	–	Image segmentation
	Baldeon and Lai-Yuen (2020)	**N:** 2, **D:** 1, **C:** 2	AdaResU-net	Segmentation accuracy	Number of parameters		–	–	–	Image segmentation
	Zhang et al. (2016)	**N:** 2, **D:** 1, **C:** -	DBN ensembles	Accuracy	–	–	–	–	Ensemble diversity	Remaining useful life prediction
ENS-MOEA/D	Zhang et al. (2020)	**N:** 1, **D:** 2, **C:** -	VMD	MSE Mutual Information	–	–	–	–	–	Wind speed prediction

Table 4 (continued)

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measures						Application field
				Error	Complexity	Model size	Time	Hardware-based	Other	
CMA-ES for MOO	Tanaka et al. (2016)	**N:**19, **D:** 6, **C:** 2	NNLM	Word error rate	–	–	Training time	–	–	Speech recognition
	Shinozaki et al. (2020)	**N:** 7, **D:** 3, **C:** 1	Spoken Language Systems	Word error rate	–	DNN file size	–	–	–	Speech recognition
	Qin et al. (2017)	**N:** 6, **D:** 4, **C:** -	NMT System	BLEU score	–	–	Validation time	–	–	Machine translation
	Ekbal and Saha (2016)	**N:** -, **D:** 2, **C:** -	CRF	Recall Precision	–	–	–	–	–	Named entity recognition
		N: -, **D:** 1, **C:** -	SVM							
		N: -, **D:** 1, **C:** -	MBL							
OMOPSO	Wang et al. (2019, 2020)	**N:** -, **D:** 5, **C:** -	Densenet-121	Accuracy	FLOPs	–	–	–	–	Image classification
	Rajagopal et al. (2020)	**N:** 3, **D:** 3, **C:** -	CNN	Accuracy	FLOPs	–	–	–	–	Scene classification
PSO (scalarized objectives)	Faris et al. (2020)	**N:** 1, **D:** 2, **C:** -	SVM	Error	Feature selection rate	–	–	–	–	Medical specialties classification
CoDeepNeat	Liang et al. (2019)	**N:** 2, **D:** 2, **C:** 3	CNN	Error	Number of parameters		–	–	–	Medical image classification

Table 4 (continued)

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measures						Application field
				Error	Complexity	Model size	Time	Hardware-based	Other	
SPEA-II (scalarized objectives)	Loni et al. (2019)	**N:** -, **D:** 2, **C:** 4	CNN	Accuracy	Number of parameters		–	–	–	Image classification
MADE	Pathak et al. (2020)	**N:** 1, **D:** 5, **C:** 4	Bidirectional LSTM	Recall Specificity	–	–	–	–	–	Classification
MODE (scalarized objectives)	Singh et al. (2020)	**N:** 2, **D:** 5, **C:** 3	CNN	Recall Specificity	–	–	–	–	–	Classification
MO-RACACO	Juang and Hsu (2014)	**N:** -, **D:** 1, **C:** -	Fuzzy Neural Networks	RMSE	–	Number of neurons	–	–	–	Regression
MOSA	Gülcü and Kuş (2021)	**N:** -, **D:** 10, **C:** 4	CNN	Accuracy	FLOPs	–	–	–	–	Image classification
Nelder-Mead (scalarized objectives)	Albelwi and Mah (2016)	**N:** -, **D:** 7, **C:** -	CNN	Accuracy Mutual information	–	–	–	–	–	Image classification

N, *D*, *C* refer to the number of numeric, discrete, and categorical hyperparameters respectively. The use of scalarization is indicated in the first column (when relevant)

Table 5 Experimental comparisons reported in the literature on metaheuristic-based HPO algorithms

HPO Algorithm	Ref.	Compared against	Quality metrics
NSGA-II	Magda et al. (2017)	Manual selection	Mean error (objective)
	Sopov and Ivanov (2015)	SPEA (Zitzler and Thiele 1999), VEGA (Schaffer 1985), SelfCOMO-GA (Sopov and Ivanov 2015)	Classification rate and number of neurons (objectives)
	Binder et al. (2020)	ParEGO (Knowles 2006)	Generalization error (objective) and hypervolume
	Shinozaki et al. (2020)	CMA-ES (Hansen et al. 2003)	WER and DNN file size (both objectives)
	Bouraoui et al. (2018)	Grid search	Accuracy (objective)
GA (scalarized objectives)	Deighan et al. (2021)	GA variants	Scalarized fitness function
MOEA/D (scalarized objectives)	Calisto and Lai-Yuen (2020)	Manual selection	Segmentation accuracy and number of parameters (both objectives)
	Baldeon and Lai-Yuen (2020)	GP-EI (Snoek et al. 2012)	Performance measures *not used as objectives* (Sensitivity, Mean surface distance, etc)
CMA-ES for MOO	Shinozaki et al. (2020)	NSGA-II (Deb et al. 2002)	WER and DNN file size (both objectives)
MO-RACACO	Juang and Hsu (2014)	MO-EA (Juang 2002), MO-ACOr (Socha and Dorigo 2008)	Coverage metric and diversity in Pareto front

a dataset). Binder et al. (2020) observed analogous results optimizing a SVM, kkNN, and XGBoost. Yet, according to the generalization-dominated hypervolume, NSGA-II performed slightly worse than ParEGO, a Bayesian optimization-based approach (see Knowles 2006 for further details). Binder et al. (2020) thus suggest to prefer NSGA-II over ParEGO only when model evaluations are cheap and marginal degradation of performance is acceptable.

Contrary to NSGA-II, the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) (Zhang and Li 2007) uses *scalarization* to solve the multi-objective HPO problem. Both MOEA/D and NSGA-II have shown to improve the accuracy of the resulting model compared with manual hyperparameter selection (Magda et al. 2017; Calisto and Lai-Yuen 2020). In Baldeon and Lai-Yuen (2020), MOEA/D is compared with a Bayesian Optimization approach (using Gaussian Process Regression with Expected Improvement as acquisition function), for tuning an adaptive convolutional neural network (AdaResU-Net) used for medical image segmentation. The use of MOEA/D resulted in a reduction in the number of parameters to train; the comparison is not really reliable, though, as the Bayesian approach was used in a single-objective optimizer, focusing only on segmentation accuracy and not on model size. The ENS-MOEA/D algorithm proposed by Zhao et al. (2012) presents a further improvement to the original MOEA/D algorithm, by adaptively adjusting the neighborhood size (as large neighborhood sizes favor more global search, while smaller sizes lead to more local search). Zhang et al. (2020) apply this method to optimize the hyperparameters of a Variational Model Decomposition (VMD) procedure, used to pre-process time series for forecasting wind speeds. The authors prove that this yields better forecasts, yet they did not perform any comparison against other HPO procedures.

The Covariance matrix adaptation-evolutionary strategy (CMA-ES) (Hansen et al. 2003) is a population-based metaheuristic that differs from Genetic Algorithms in the use of a fixed-length real-valued vector as a gene (instead of the typical vector of binary components), and a multivariate Gaussian distribution to generate new solutions. Multi-objective CMA-ES can be formulated considering the dominance of solutions on the Pareto Frontier, to redefine the ranking function used to determine the best solution found so far (now a Pareto front) (Tanaka et al. 2016; Qin et al. 2017; Shinozaki et al. 2020). Shinozaki et al. (2020) optimize DNN-based Spoken Language Systems using this approach; the resulting networks had lower word error rates and were smaller than the networks designed by NSGA-II. Additionally, multi-objective CMA-ES generated smaller networks than the one obtained with single-objective CMA-ES (using the error-based measure as an objective to optimize). In our opinion, though, this last comparison does not make much sense, since network size did not appear as an objective in the single-objective setting.

Analogous to Genetic Algorithms, Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995) works with a population of candidate solutions, known as *particles*. Each particle is characterized by a velocity and a position. The particles search for the optimal solutions by continuously updating their position and velocity. Their movement is influenced not only by their own local best-known position but is also guided toward the best-known position found by other particles in the search space. A multi-objective PSO algorithm (OMOPSO) was developed by Sierra and Coello (2005), using Pareto dominance and crowding distance to filter out the best particles. It employs different mutation operators which act on subsets of the swarm, and applies the ϵ-dominance concept (see Laumanns et al. 2002 for more details) to fix the size of the set of final solutions produced by the algorithm.

Strength Pareto Evolutionary Algorithm II (SPEA-II) (Zitzler et al. 2001) adds several improvements to the original SPEA algorithm presented by Zitzler and Thiele (1999). Loni et al. (2019) used the algorithm to optimize six hyperparameters of a CNN, yielding more accurate and less complex networks than could be obtained with hand-crafted networks, or with NAS algorithms.

Differential Evolution (DE) (Storn and Price 1997) is similar to Genetic Algorithms but differs in the way in which the solutions are coded (using real vectors instead of binary-coded ones) and, consequently, in the way in which the evolutionary operators are applied. Multi-Objective Differential Evolution (MODE) (Babu and Gujarathi 2007) selects the non-dominated solutions to generate new solutions on each iteration. To reduce the computational effort while maintaining accuracy, a memetic adaptive DE method (MADE) was developed by Li et al. (2019). DE depends significantly on its control parameter settings. Therefore, MADE uses a historical memory of successful control parameter settings to guide the selection of future control parameter values (Tanabe and Fukunaga 2013). Additionally, a local search method (e.g., the Nelder-Mead simplex method (NMM) (Li et al. 2019), or chaotic local search (Pathak et al. 2020)) is employed to refine the solutions, and a ranking-based elimination strategy (using non-dominated and crowding distance sorting) is proposed to maintain the most promising solutions.

Ant Colony Optimization (ACO) (Dorigo et al. 1996) is inspired by the behavior of real ants; the basic idea is to model the HPO problem as the search for a minimum cost path in a graph. ACO algorithms can be applied to solve multi-objective problems, and may differ in three respects (Alaya et al. 2007): (1) the way solutions are built, using only one *pheromone structure* for an aggregation of several objectives, or associating a different pheromone structure with each objective (Iredi et al. 2001; Gravel et al. 2002; 2) the way in which solutions are updated (Iredi et al. 2001; Barán and Schaerer 2003) and (3) the incorporation of existing problem-specific knowledge into the transition rule that defines how to create new solutions from existing ones (Gravel et al. 2002; Doerner et al. 2004). The latter is included in a multi-objective version of ACO (MO-RACACO, Hsu and Juang 2013) for Fuzzy Neural Network (FNN) optimization (Juang and Hsu 2014). The results showed that MO-RACACO outperformed other population-based MO algorithms (MO-EA, Juang 2002; and MO-ACOr, Socha and Dorigo 2008) in terms of the coverage measure obtained, yet it did not always obtain the best diversity values.

Simulated annealing (SA) is a probabilistic technique for finding the global optimum of a single-objective problem (Kirkpatrick et al. 1983). Gülcü and Kuş (2021) applied a multi-objective approach (MOSA) to optimize 14 hyperparameters of a CNN. The algorithm selects new solutions based on their relative merit (measured by the dominance relationship) w.r.t. the current solutions.

The Nelder-Mead simplex method (NMM) (Olsson and Nelson 1975) has been applied by Albelwi and Mah (2016) to optimize seven hyperparameters for a CNN. As NMM is a single-objective optimization procedure, the objectives need to be scalarized (the authors used a weighted sum approach). NMM is a local optimization procedure, so it may get stuck in a local minimum. This may be avoided by running the algorithm from different starting points, which increases the probability of reaching the global minimum. Alternatively, modifications to the algorithm have been proposed (as in McKinnon 1998) that allow the algorithm to escape from local minima, yet at the cost of a large number of iterations.

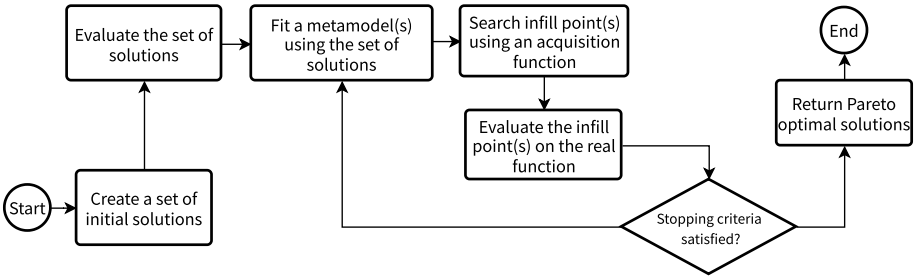


Fig. 6 Generic optimization procedure in metamodel-based MOO algorithms

4.2 Metamodel-based HPO algorithms

Training a machine learning algorithm can be computationally expensive, e.g. due to the target algorithm's own structure (e.g., Deep Learning models), the amount and complexity of the data to process, resource limitations (execution time, memory and energy consumption, etc), and/or the type of training algorithm used. Therefore, different HPO approaches have been developed that employ less expensive models (referred to as metamodels or surrogate models) to emulate the computation of the real performance functions. The resulting algorithms have also been referred to as Efficient Global Optimization (EGO) or Bayesian Optimization (BO) algorithms, and use an *acquisition function* or *infill criterion* to guide the search. Figure 6 summarizes the main steps in such an algorithm.

The optimization starts with a set of initial points (input/output observations) to train the metamodel. Next, the acquisition function is used to select one or more new points (infill points) to be evaluated. The use of this acquisition function is a key element in the search (approaches that combine metamodels with metaheuristic search are referred to as *hybrid* methods, and are discussed in Sect. 4.3). The metamodel is updated with this new information (adding the new I/O observations to the initial set), and the procedure continues until a stopping criterion is met.

For ease of reference, Table 6 gives an overview of the metamodel-based algorithms currently used for multi-objective HPO, while Table 7 gives an overview of the experimental comparisons reported in this part of the literature. As evident from Table 6, most multi-objective HPO articles use a Gaussian Process (GP) metamodel. GPs use a covariance function, or kernel, to compute the *spatial correlation* among several output observations for a given performance measure (i.e., a given objective of the HPO algorithm; see Fig. 3). In this approach, it is assumed that HPO input configurations that differ only slightly from one another (i.e., they are *close* to each other in the search space) are strongly positively correlated w.r.t. their outputs; as the configurations are further apart in the search space, the correlation dies out. The choice of the kernel in a GP is important, as it determines the shape of the assumed correlation function. In general, the most common kernels used in GP-based metamodels are the Gaussian kernel and the Matérn kernel (Ounpraseuth 2008). Using the kernel, the analyst can not only *predict* the estimated outputs (i.e., in our case, the performance measures) at non-observed input locations (i.e., hyperparameter configurations), but can also estimate the *uncertainty* on these output predictions. Both the predictions and their uncertainty are reflected in the acquisition function to search for new hyperparameter settings. We refer the reader to Rojas-Gonzalez and Van Nieuwenhuyse (2020) for a detailed review of acquisition functions, for (general, non-HPO related) single and multi-objective optimization problems.

Table 6 Overview of Metamodel-based HPO algorithms

Metamodel	Acquisition function	Ref.	HP	Target ML algorithm	Performance measures						Application field
					Error	Complexity	Model size	Time	Hardware-based	Other	
Gaussian Process (deterministic observations)	EI (scalarized objectives)	Salt et al. (2019)	**N:**18, **D:** -, **C:** -	SNN	Accuracy Reward of the spiking trace SSE	–	–	–	–	–	Classification
	Dominance rank of GP predictions	Parsa et al. (2019)	**N:** 3, **D:** 6, **C:** -	AlexNET	Error	–	–	–	Energy consumption	–	Image classification
			N: -, **D:**11, **C:** -	VGG19							
	Preferences-based EHI	Abdolsh et al. (2019)	**N:** 4, **D:** 2, **C:** -	NN	Prediction error	–	–	Prediction time	–	–	Image classification
	CEIPV	Shah and Ghahramani (2016)	**N:** 1, **D:** 2, **C:** -	NN	Accuracy	–	–	–	Memory footprint	–	Classification
	LCB	Richter et al. (2016)	**N:** -, **D:** 2, **C:** -	SVM	Classification error	–	–	Running time	–	–	Classification
	UCB (scalarized objectives)	Chin et al. (2020)	**N:** 6, **D:** -, **C:** -	Slimmable NN	Cross entropy loss	FLOPs	–	–	–	–	Classification
	PES	Hernández et al. (2016)	**N:** 4, **D:** 2, **C:** -	NN	Prediction error	–	–	Prediction time	–	–	Image classification
		Garrido and Hernández (2019)	**N:** 2, **D:** 3, **C:** -	Ensemble of Decision Trees	Prediction error	–	Ensemble size	–	–	–	Classification
		Hernández-Lobato et al. (2016)	**N:** 4, **D:** 4, **C:** -	NN	Prediction error	–	–	–	Energy consumption	–	Classification

Table 6 (continued)

Metamodel	Acquisition function	Ref.	HP	Target ML algorithm	Performance measures						Application field
					Error	Complexity	Model size	Time	Hardware-based	Other	
Random Forest	LCB	Binder et al. (2020)	**N:** -, **D:** 2, **C:** -	SVM	Generalization error	Fraction of selected features	–	–	–	–	OpenML benchmark
			N: 5, **D:** 4, **C:** -	XGBoost							
			N: -, **D:** 2, **C:** 1	kkNN							
		Horn and Bischl (2016)	**N:** -, **D:** 3, **C:** -	SVM	FNR FPR	–	–	–	–	–	OpenML Benchmark
			N: -, **D:** 2, **C:** -	Random Forest							
			N: -, **D:** 2, **C:** -	Logistic regression							
Tree Parzen Estimators	Modified EHV (scalarized objectives)	Chandra and Lane (2016)	**N:** 3, **D:** 3, **C:** -	NN-decoder	Word error rate	–	–	Decoding time	Memory consumption	–	Speech recognition
Gaussian Process (noisy observations)	EHI	Horn et al. (2017)	**N:** -, **D:** 3, **C:** -	SVM	FNR FPR	–	–	–	–	–	OpenML benchmark
		Koch et al. (2015)	**N:** 2(5), **D:** -, **C:** -	SVM	Accuracy	–	–	Training time	–	–	UCI benchmark

N, *D*, *C* refer to the number of numeric, discrete, and categorical hyperparameters respectively. The use of scalarization is indicated below the acquisition function (when relevant)

Table 7 Experimental comparisons reported in the literature on metamodel-based MO HPO algorithms

HPO Metamodel	Ref.	Compared against	Quality metrics
Gaussian process (deterministic observations)	Salt et al. (2019)	Random search, DE (Storn and Price 1997), SADE (Qin et al. 2008)	Performance measures not used as objectives
	Abdolsh et al. (2019)	PESMO (Hernández et al. 2016), SMS-EGO (Ponweiser et al. 2008), SUR (Picheny 2014), ParEGO (Knowles 2006)	Descriptive analysis of the Pareto front
	Parsa et al. (2019)	Grid search, Random search, NSGA-II (Deb et al. 2002)	Execution time
	Shah and Ghahramani (2016)	ParEGO (Knowles 2006), Random search, GP-EHV	Hypervolume
	Richter et al. (2016)	Random search	Classification error and running time (both objectives) and CPU usage
	Hernández et al. (2016)	ParEGO (Knowles 2006), SMS-EGO (Ponweiser et al. 2008), SUR (Picheny 2014)	Hypervolume
	Garrido and Hernández (2019)	Random search, BMOO (Feliot et al. 2017)	Hypervolume
	Hernández-Lobato et al. (2016)	Random search, NSGA-II (Deb et al. 2002)	Hypervolume
Gaussian process (noisy observations)	Horn et al. (2017)	RTEA (Fieldsend and Everson 2014) , Random search	Hypervolume and runtime
	Koch et al. (2015)	SMS-EGO (Ponweiser et al. 2008) , Latin Hypercube sampling	Hypervolume
Random Forest	Horn and Bischl (2016)	SMS-EGO (Ponweiser et al. 2008), ParEGO (Knowles 2006), Random sampling, NSGA-II (Deb et al. 2002)	Hypervolume
TPE	Chandra and Lane (2016)	Random sampling, GP, Genetic Algorithm (Zames et al. 1981)	WER, decoding time, and memory consumption (all ojectives)

Table 6 also shows the acquisition functions that have been used so far in multi-objective HPO. Clearly, the most popular one is Expected Improvement (EI, which was originally proposed by Jones et al. 1998). The EI represents the expected improvement over the best outputs found so far, at an (arbitrary) non-observed input configuration. As EI was originally developed for single-objective problems, it is usually applied in multi-objective problems where the objectives are scalarized. Salt et al. (2019), for instance, optimize a Spiking Neural Network (SNN) using a weighted function of three individual objectives (the accuracy, the sum square error of the membrane voltage signal, and the reward of the spiking trace). Three acquisition functions were studied; EI, Probability of Improvement (POI), and Upper Confidence Bound (UCB). The performance obtained with POI was significantly better than that obtained with EI and UCB, and overall, the BO-based approach required significantly fewer evaluations than evolutionary strategies such as SADE.

Another way to use BO in multi-objective HPO is to fit a metamodel to each objective independently. Parsa et al. (2019) use such an approach in their Pseudo Agent-Based multi-objective Bayesian hyperparameter Optimization (PABO) algorithm; they use the dominance rank (based on the predictor values of each objective) as an infill criterion. This evidently yields different infill points for the respective objectives (in their case, an error-based objective and an energy-related objective). The infill point suggested for one objective function is then also evaluated for the other objective function, provided that it is not dominated by any previous HPO configuration analyzed. In this way, the algorithm speeds up the search for Pareto-optimal solutions. The experiments indeed demonstrated that PABO outperforms NSGA-II in terms of speed.

Other authors have studied HPO problems when the performance measures are correlated (Shah and Ghahramani 2016), or when one of the measures is clearly more important than the others (Abdolsh et al. 2019). The algorithm proposed by Shah and Ghahramani (2016) models the correlations between accuracy, memory consumption, and training time of an ANN using a multi-output Gaussian process or Co-Kriging (Liu et al. 2018). The authors propose a modification to the expected hypervolume (EHV) that reflects these correlations; this modified EHV is then used as an acquisition function, preferring the infill point that increases the expected hypervolume of the Pareto front the most. The algorithm is compared to ParEGO, (Knowles 2006), random search, and a GP using the original EHV metric. The results suggest that the modified EHV criterion increases the speed of the optimization, requiring fewer iterations to converge to the Pareto optimal solutions.

The MOBO-PC algorithm proposed by Abdolsh et al. (2019) adjusts the *Expected Hypervolume Improvement* (EHI) acquisition function to account for the probability that the novel HP configuration satisfies a set of user-defined preference-order constraints. In this way, it manages to focus its search on the Pareto solutions that are most relevant for the user, as opposed to the other algorithms that are used as a comparison in the paper (PESMO, Hernández et al. 2016; SMS-EGO, Ponweiser et al. 2008; Stepwise Uncertainty Reduction, Picheny 2014; and ParEGo, Knowles 2006), which try to find solutions across the entire Pareto front.

Other acquisition functions used in metamodel-based algorithms are the Lower Confidence Bound (LCB) or Upper Confidence Bound (UCB). These use a (user-defined) confidence bound to focus the search on local areas or explore the search space more globally. Richter et al. (2016) use a multipoint LCB which simultaneously generates q hyperparameter configurations. A GP is used to model the misclassification error and the logarithmic runtime. The results demonstrated an improvement in CPU utilization (and, thus, an increase in the number of hyperparameter evaluations) within the same time budget. Confidence bounds are also used by Chin et al. (2020) to optimize the hyperparameters

of Slimmable Neural Networks. The algorithm fits a GP to each individual performance measure, hence obtaining information to compute individual UCBs. These UCBs are then scalarized, and the resulting single objective function is minimized to obtain the next infill point. The proposed algorithm succeeds in reducing the complexity of the NNs studied; yet, the authors did not compare its performance with any other multi-objective HPO algorithms.

The Predictive Entropy Search (PES) criterion is used by multiple authors, as an infill criterion for different algorithms. Hernández et al. (2016) use PESMO (multi-objective PES) to optimize a NN with six hyperparameters, in view of minimizing the prediction error and the training time. PESMO seeks to minimize the uncertainty in the *location of the Pareto set*. The algorithm is compared with ParEGO, SMS-EGO, and SUR, showing that PESMO gives the best overall results in terms of hypervolume and the number of expensive evaluations required for training/testing the neural network. Garrido and Hernández (2019) use PESMOC (a modified version of PESMO which takes into account constraints) to optimize an ensemble of Decision Trees. The experiments show that PESMOC is able to obtain better results than a state-of-the-art method for constrained multi-objective Bayesian optimization (Feliot et al. 2017), in terms of the hypervolume obtained and the number of evaluations required. Finally, Hernández-Lobato et al. (2016) used PES to design a neural network with three layers. While most of the HPO methods collect data in a *coupled* way by always evaluating all performance measures jointly at a given input, these authors consider a *decoupled* approach in which, at each iteration, the next infill configuration is selected according to the maximum value of the acquisition functions across all objectives. The results showed that this approach obtains better solutions (compared to NSGA-II and random search) when computational resources are limited; yet, the trade-offs found among the performance measures may be affected and one of the objectives can turn out to be prioritized over the others.

Random forests (RFs) (Ho 1995) are an ensemble learning method that trains a set of decision trees having low computational complexity. Each tree is trained with different samples, taken from the initial set of observations. For classification outputs, the RF uses a voting procedure to determine the decision class; for regression output, it returns the average value over the different trees. As for GP, RFs allows the analyst to obtain an uncertainty estimator for the prediction values. Some examples are the quantile regression forests method (Meinshausen and Ridgeway 2006), which estimates the prediction intervals, and the U-statistics approach (Mentch and Hooker 2016). Horn and Bischl (2016) use RFs as metamodel to optimize the hyperparameters of three ML algorithms: SVM, Random Forest, and Logistic regression. Using LCB as an acquisition function, the authors show that SMS-EGO and ParEGO outperform random sampling and NSGA-II.

Whereas GP-based approaches model the density function of the resulting outcomes (performance measures) given a candidate input configuration, Tree-structured Parzen Estimators (TPE) (Bergstra et al. 2011) model the probability of obtaining an input configuration, given a condition on the outcomes. TPEs naturally handle not only continuous but also discrete and categorical inputs, which are difficult to handle with a GP. Moreover, TPE also works well for conditional search spaces (where the value of a given hyperparameter may depend on the value of another hyperparameter), and has demonstrated good performance on HPO problems for single-objective optimization (Bergstra et al. 2013; Thornton et al. 2013; Falkner et al. 2018). While it can, in theory, also be applied to multi-objective settings by scalarizing the performance measures, Chandra and Lane (2016) obtained disappointing results when comparing this approach with random sampling, GP and Genetic Algorithms for optimizing an Augmented Tchebycheff scalarized

function (Miettinen 2012) (using fixed weights) of three performance measures for ANNs: GP performed best, while TPE performed worst. Unfortunately, the authors reported the performance based solely on the scalarized value of the three performance measures; they did not report on any other quality metrics, such as hypervolume. They also did not discuss the reason for the poor TPE performance, such that it remains unclear whether this is due to the scalarization function, or to the characteristics of the search space. A (non-scalarized) multi-objective version of TPE has been proposed by Ozaki et al. (2020) and is included in the software Optuna (Akiba et al. 2019).

Strikingly, the majority of current HPO algorithms routinely ignore the fact that the obtained performance measures are *noisy*. The noise can be due to either the target ML algorithm itself (when it contains randomness in its procedure, such as a NN that randomly initializes the weights), but even if there is no randomness involved, there will be noise on the outcomes due to the use of k-fold cross-validation during the training of the algorithm. This type of cross-validation is common in HPO: it involves the creation of different *splits* of the data into a training and validation set. This process is repeated k times; the performance measures of a given hyperparameter combination will thus differ for each split. Current HPO algorithms focus simply on the *average* performance measures *over the different splits* during the search for the Pareto-optimal points; the inherent uncertainty on these performance measures is ignored. Horn et al. (2017) are one of the few authors to highlight the presence of noise. The paper assumes, though, that noise is homogenous (i.e., it doesn't differ over the search space), and only focuses on different strategies for handling this noise. These strategies are used in combination with the SMS-EGO algorithm (Ponweiser et al. 2008) and compared with the rolling tide evolutionary algorithm (RTEA) (Fieldsend and Everson 2014) and random search. The results show that simply ignoring the noise (by evaluating a given HPO combination only once, and considering the resulting performance measures as deterministic) performs poorly, even worse than a repeated random search. The best strategy is to reevaluate the (most promising) HP settings. According to the authors, this can likely be explained by the fact that the *true* noise on the performance measures in HPO settings is heterogeneous (i.e., its magnitude differs over the search space). Reevaluation of already observed HP settings is then required to improve the reliability of the observed performance measures. The interested reader is referred to Jalali et al. (2017) for a discussion of the impact of noise magnitude and noise structure on the performance of (general) optimization algorithms.

Koch et al. (2015) adapt SMS-EGO (Ponweiser et al. 2008) and SExI-EGO (Emmerich et al. 2011) for noisy evaluations, to optimize the hyperparameters of a SVM. The authors again assume that the noise is homogenous, and compare the performance of both algorithms with different noise handling strategies (the reinterpolation method proposed by Forrester et al. (2006), and static resampling). Both algorithms use the expected hypervolume improvement (EHI) as an infill criterion, though the actual calculation of the criterion is different (causing Sexi-EGO to require larger runtimes). The results show that both SMS-EGO and SExI-EGO work well with the reinterpolation method, yielding comparable results in terms of hypervolume.

4.3 Hybrid HPO algorithms

A limited number of papers have combined aspects of metamodel-based and population-based HPO approaches: these are referred to in Table 8, summarizing their main

Table 8 Overview of hybrid HPO algorithms

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measures						Application field
				Error	Complexity	Model size	Time	Hard-ware-based	Other	
ANN + DSE	Smithson et al. (2016)	**N:** 1, **D:** 2, **C:** 1	MLP	Accuracy	Number of parameters		–	–	–	Image classification
		N: 1, **D:** 4, **C:** 2	CNN							
GP + GA Parsimony	Martinez-de Pison et al. (2017)	**N:** 5, **D:** 3, **C:** -	XGBoost	RMSE	Number of features	–	–	–	–	Image classification
(MLP/CART/ RBF/GP) + NSGA-II	Lu et al. (2020)	**N:** -, **D:** 4, **C:** -	CNN	Accuracy	MAdds	–	–	–	–	Image classification
Random Forest + ES	Calisto and Lai-Yuen (2021)	**N:** -, **D:** 6, **C:** 4	CNN	Segmentation error	Number of parameters		–	–	–	Image segmentation

N, *D*, *C* refer to the number of numeric, discrete, and categorical hyperparameters respectively

Table 9 Experimental comparisons reported in the literature on hybrid MO HPO algorithms

HPO Algorithm	Ref.	Compared against	Quality metrics
ANN + DSE	Smithson et al. (2016)	Exhaustive search	Generational Distance
(MLP/CART/RBF/GP) + NSGA-II	Lu et al. (2020)	NAS algorithms	Cumulative hypervolume, model size, and CPU/GPU latency

characteristics. Table 9 gives an overview of the experimental comparisons reported in these papers.

Smithson et al. (2016) use an ANN as a metamodel to estimate the performance of the target ML algorithm. The neural network is embedded into a Design Space Exploration (DSE) metaheuristic, and is used to intelligently select new solutions that are likely to be Pareto optimal. The algorithm starts with a random solution, and iteratively generates new solutions that are evaluated with the ANN. DSE decides if the solution should be used to update the ANN knowledge, or should be discarded. Compared with manually designed networks from the literature, the proposed algorithm yields results with nearly identical performance, while reducing the associated costs (in terms of energy consumption).

The algorithm proposed by Martinez-de Pison et al. (2017) combines HPO with feature selection (as opposed to other algorithms, e.g., Ekbal and Saha 2015; León et al. 2019; Guo et al. 2019). First, a GP (with UCB as an acquisition function) is used to obtain the best HPO setting (according to the RMSE), considering the full set of features. Next, a variant of GA (GA-PARSIMONY, Sanz-García et al. 2015) is used to select the best features of the problem, given the hyperparameters obtained in the first step. In this way, the final model has high accuracy and lower complexity (i.e., fewer features), and optimization time is significantly reduced. In our opinion, however, this approach is still suboptimal, as the two optimization problems (HPO and feature selection) are solved sequentially, instead of jointly. Calisto and Lai-Yuen (2021) use an evolutionary strategy combined with a Random Forest metamodel, to optimize 10 hyperparameters of a CNN. In the beginning of the optimization, the algorithm updates the population of solutions using the evolutionary strategy; only after some iterations, the selection of the new candidates is guided by the RF, which is updated each time with all new Pareto front solutions. The final networks found by the algorithm perform better than (or equivalent to) state-of-the-art architectures, while the size of the architectures and the search time are significantly reduced.

Although most NAS algorithms are out of scope for this survey, we include the work by Lu et al. (2020), as it can be considered an HPO algorithm. The algorithm (NSGANetV2) simultaneously optimizes the architectural hyperparameters and the model weights of a CNN, using a bi-level approach consisting of NSGA-II combined with a metamodel. The metamodel is used to estimate performance measures, which are then optimized by an evolutionary algorithm (such approaches have also been applied successfully to non-HPO settings, see e.g., Jin 2011; Dutta and Gandomi 2020). In the upper level of the optimization, the metamodel is built using an initial set of candidate solutions. In each iteration of the upper level, NSGA-II is executed on the metamodel to detect the Pareto-optimal HP settings (configuration of layers, channels, kernel size, and input resolution of the CNN). At the lower level, the weights of the CNN are trained on a subset of the Pareto-optimal solutions. The metamodel is then updated with the results of the actual performance evaluations. Four different metamodels were studied; Multilayer Perceptron (MLP), Classification and Regression Trees (CART), Radial Basis Functions (RBF), and GP. Given that none of

them consistently outperformed the others, the authors propose to select the best metamodel in every iteration. On standard datasets (CIFAR-10, CIFAR-100, and ImageNet), the resulting algorithm matches the performance of state-of-the-art NAS algorithms (et al. 2019; Mei et al. 2020), but at a reduced search cost.

5 Multi-objective HPO algorithms: pros and cons

In this section, we discuss the weakness and strengths of the different algorithms. We focus on four different aspects: (1) the computational complexity of the algorithm, (2) the ability to accommodate high dimensional input spaces, (3) the ability to handle mixed input spaces, and (4) the ease of use of parallel computations. Unfortunately, none of the papers studied in this review provides explicit details on these aspects in the publication. In general, we often observed a surprising lack of detail with respect to many methodological aspects (such as the nature of the hyperparameters being optimized, the nature of the genetic operators and the design of the initial population in metaheuristic-based algorithms, the design of experiments used, the final Pareto-optimal solutions provided by the algorithm, etc.). In many cases, there is even no pseudocode provided for the algorithm, and detailed descriptions of novel metrics (if any) used to measure the performance of the target ML algorithm are lacking. This lack of detail is likely caused by the fact that most papers aim to solve a particular practical application and the hyperparameter optimization was usually not seen as the main contribution of the paper.

Consequently, the discussion in this section remains quite general, and relies largely on the results of our own independent research, based on the information found in *methodological* papers for the algorithms considered. This information also allowed us to outline rough pseudocodes of the algorithms (which are presented in Appendix 1). Although we emphasize (again) that these pseudocodes do *not* necessarily reflect the accurate details of the algorithms, we find them helpful, in particular, to estimate the complexity of the algorithms. For black-box algorithms, this complexity can be measured by means of their *worst-case expected running time* (Doerr 2020). The running time (or *optimization time* of an algorithm for a function f is defined as the *number of function evaluations* that the algorithm performs until (and including) the evaluation of an optimal solution for f. For HPO algorithms, the running time is largely proportional to the number of training and validation steps performed, as these are the most expensive steps in the HPO procedure. The training and validation steps need to be performed for *each* HPO configuration studied by the HPO algorithm. Consequently, in what follows, we propose to use the (worst-case) number of HPO configurations evaluated by the algorithm as a proxy for the algorithm's expected worst-case running time. The result is expressed as a function $g(n, I, N)$, which is influenced by three parameters: (1) the number of initial HP configurations n required to start the optimization (e.g., the size of the initial population in evolutionary algorithms, or the size of a Latin hypercube sample for Bayesian optimization), (2) the number of iterations I allowed during the search, and (3) the number of new HPO configurations N generated per iteration. Table summarizes the results of our analysis.

Clearly, the number of costly function evaluations in a typical metamodel10-based optimization is much lower than in a metaheuristic-based algorithm, as usually only a single new solution is evaluated in each iteration. MADE, the metaheuristic-based algorithm by Pathak et al. (2020), can be particularly expensive, as it performs a chaotic local search to generate N additional solutions for each solution present in the

Table 10 Analysis of pros and cons for the MO HPO algorithms studied

HPO Algorithm	Parallelization	Number of HP configurations evaluated	High dimensional input space	Mixed search space
GA, NSGA-II, CoDeepNeat, SPEA-II, ENS-MOEA, MOEA/D, (MLP/CART/RBF/GP) + NSGA-II		$g(n, I, N) = n + IN$		
MADE		$g(n, I, N) = n + 2IN$	Unknown	Requires implementation of specific genetic operators
OMOPSO, PSO		$g(n, I, N) = n + IN$	Poor (Gad 2022)	Requires a mixed-variable encoding scheme and specific reproduction methods (Wang et al. 2021)
ACO		$g(n, I, N) = n + IN$	Poor (Ab Wahab et al. 2015)	Mainly designed for discrete search spaces
CMA-ES for MOO	Yes (for new solutions)	$g(n = 0, I, N) = IN$	Poor (due to the covariance matrix inversion (Shimizu and Toyoda 2021))	Requires adjustments for non-real variables
GP-based metamodel	Yes (for initial set of solutions, metamodel training)		Poor (Tripathy et al. 2016)	Requires specific kernel and specific optimization procedure for the acquisition function
RF-based metamodel			Good (Belgiu and Drăguţ 2016)	No issues
TPE	Yes (for initial set of solutions)	$g(n, I, N = 1) = n + I$	Unknown	No issues
MODE	Yes (for each configuration, both in the initial set and new ones)	$g(n, I, N = 1) = n + I$	Unknown	Requires specific operators to generate new solutions
MOSA	Yes (to run the algorithm with different starting points, to explore new solutions in the neighborhood of the best solution)	$g(n = 0, I, N) = IN$	Unknown	Requires specific operators for neighborhood generation
NMA	Yes (to run the algorithm with different starting points, Joorabian and Afzalan (2014))	$g(n, I, N) = n + 1 + I(N + 1)$	Unknown	Requires redefinition of the reflection, expansion, contraction and shrink operators

Table 10 (continued)

HPO Algorithm	Parallelization	Number of HP configurations evaluated	High dimensional input space	Mixed search space
GP + GA Parsimony	Yes (for each configuration, both in the initial set and new ones)	$g(n, I, N) = 2n + I(N + 1)$	Poor (due to the metamodel)	Requires specific kernel and specific optimization procedure for the acquisition function in the metamodel-based optimization phase
ANN + DSE	Yes (for the initial set of solutions)	$g(n, I, N = 1) = n + I$	Unknown	Requires specific operator to generate new solutions
RF + ES	Yes (for the initial set of solutions, new solutions in each iteration, and metamodel training)	$g(n, I, N) = n + IN$	Unknown	Requires specific operator to generate new solutions

The number of HP configurations evaluated is expressed as a function $g(n, I, N)$, where n is the number of initial solutions, I is the number of iterations, and N is the number of new HPO configurations per iteration

population of a given iteration. However, using a metamodel to reduce the number of HP configurations that need to be evaluated does not ensure a lower execution time. For instance, the hybrid algorithm GP + GA_Parsimony (Sanz-García et al. 2015) tries to optimize both hyperparameters and features used to train the ML model; the running time remains high, however, as the feature selection is performed in a separate phase after the HPO has been performed: this leads to a drastic increase in the number of HP configurations evaluated, compared with other algorithms such as NSGA-II and GP-based optimization.

The use of parallel computations may be considered to decrease the total execution time of the optimization. For metaheuristic-based algorithms, this is usually implemented by parallelizing the evaluation of novel configurations in each population generation (Durillo et al. 2008; Wang et al. 2018). Parallelization has been observed in metaheuristic-based optimization algorithms such as CMA-ES (Tanaka et al. 2016; Qin et al. 2017), CoDeepNeat (Liang et al. 2019), GA (Deighan et al. 2021), and NSGA-II (Kim et al. 2017); it has also been suggested in (Albelwi and Mah 2016; Baldeon and Lai-Yuen 2020) for DNN optimization. Bayesian Optimization approaches, by contrast, are inherently serial as they use past observations to determine the next point(s) to sample. Parallelization can be used to some extent, though, e.g. in the evaluation of the initial set of configurations, or in batch BO (Richter et al. 2016; Binder et al. 2020; Horn and Bischl 2016). Parallel computations can also be introduced during the training/validation of the ML algorithm (by training/validating the model simultaneously on the different data splits in the cross-validation protocol (Mostafa et al. 2020)), or during the training of the metamodel [e.g., for Random Forests (Chen et al. 2016) and for Gaussian Processes (Dai et al. 2014)].

The ability of an algorithm to handle mixed input spaces is not evident. For metaheuristic-based optimization procedures, for instance, this requires a proper coding of the solutions (e.g., the chromosomes in GAs or the particles in PSO), and consequently a reformulation of the evolutionary operators. For algorithms such as ACO, NMA, and CMA-ES, we expect that handling mixed search spaces is not straightforward, given that they were originally designed for a specific type of variables (ACO for discrete variables that can be easily structured in a graph, and NMA and CMA-ES for continuous variables). In metamodel-based optimization approaches using GPs, a proper kernel needs to be used to accommodate mixed input spaces. Metamodel-based approaches that rely on Random Forests or TPE, by contrast, can handle a mix of discrete, categorical, and numerical variables quite straightforwardly.

To judge the ability of the algorithms to handle high dimensional search spaces, we relied on the findings of other studies (see the references in Table 10). We categorize the results into poor (meaning that the ability to handle high dimensional search spaces is problematic), good, or unknown (meaning that no discussions on this aspect were found).

6 Conclusions and future research

This paper has reviewed the literature on multi-objective HPO algorithms, categorizing relevant papers into metaheuristic-based, metamodel-based, and hybrid approaches. The literature on MO HPO is not as abundant as on single-objective HPO; yet, MO HPO is highly relevant in practice. Taking a multi-objective perspective on HPO not only allows the analyst to optimize trade-offs between different performance measures, but it may also even

Table 11 Summary of research opportunities for multi-objective hyperparameter optimization

Type	Recommendations
Methodological	Use of hybrid algorithms
	Use of ensembles (of metamodels, acquisition functions, etc.)
	Multi-fidelity methods and/or bandit-based methods
	Use of early stopping criteria
	Use of algorithms that account for heterogeneous noise in performance objectives
General	Use of individual performance metrics instead of aggregated metrics
	Include a clear description of search space characteristics (type and range of considered HPs), algorithmic details (with pseudocode), performance objectives, and final optimal solutions obtained (optimal configurations, a quality metric for the Pareto front, etc.)
	Benchmark novel algorithms w.r.t. existing algorithms

yield *better* solutions than the corresponding single-objective HPO problem. For instance, it has been shown that including complexity as an objective in multi-objective HPO does not necessarily compromise the loss-based performance of the ML algorithm w.r.t. the task for which it is trained: particularly, the minimization of the number of features used for training can *improve* the performance of the ML algorithm (Sopov and Ivanov 2015; Binder et al. 2020; Bouraoui et al. 2018; Faris et al. 2020).

As the field of multi-objective HPO is gaining speed, it presents diverse opportunities for further research. We present recommendations here, distinguishing between (1) methodological recommendations (focusing on the use of more advanced optimization approaches), and (2) general recommendations (focusing on shortcomings or pitfalls that currently occur in the literature, and that—in our opinion—hamper the reproducibility, usability, and interpretability of the results). The recommendations are outlined in Table 11.

In the current literature, metaheuristic-based HPO approaches are clearly the most popular. This is quite striking, as such approaches require the evaluation of a large amount of HP configurations, and training/testing the target algorithm for any given HP configuration is usually the most expensive step in the HPO algorithm (due to, e.g., the k-fold cross-validation, the optimization steps required for the algorithm's internal parameters, the evaluation of potentially expensive performance measures such as energy consumption or inference time, etc.). Further research on hybrid HPO algorithms appears promising here. So far, research on these algorithms remains scarce; yet, one would expect that such algorithms combine the best of two worlds, providing low computational cost (as the metamodel provides inexpensive function evaluations) along with a heuristic search that avoids the challenge of optimizing an acquisition function.

Current results have also demonstrated that using *ensembles* of optimal HP configurations can yield improvements (Ekbal and Saha 2015; Sopov and Ivanov 2015; Ekbal and Saha 2016; Zhang et al. 2016). Yet, this evidently increases the number of HP evaluations required. In future research, it may be promising to look at ensembles of multiple metamodels (Wistuba et al. 2018; Cho et al. 2020), multiple acquisition functions (Cowen-Rivers et al. 2020), or even multiple optimization procedures (Liu et al. 2020).

Furthermore, multiple opportunities exist to extend recent advanced approaches for single-objective HPO towards multi-objective HPO. Recent research has shown potential benefits in studying cheaply available (yet lower fidelity) information, obtained for

instance by evaluating only a fraction of the training data or a small number of iterations. Low fidelity methods such as bandit-based approaches (Li et al. 2017) have, to the best of our knowledge, not yet been applied in multi-objective HPO. Also, early stopping criteria (Dai et al. 2019) could be considered to ensure more intelligent use of the available computational budget. This has already been applied in single-objective optimization (Kohavi and John 1995; Provost et al. 1999), by considering the algorithm's learning curve: the training procedure for a given hyperparameter configuration is then stopped when adding further resources (training instance, iterations, training time, etc) is predicted to be futile. Early stopping criteria have also been used to reduce the overfitting level of the ML algorithm (Makarova et al. 2021). To the best of our knowledge, none of these methodological approaches has been applied so far in multi-objective HPO algorithms.

Finally, apart from the work of Koch et al. (2015) and Horn et al. (2017), the uncertainty in the performance measures is commonly ignored in HPO optimization. These two algorithms have mainly explored the impact of different noise handling strategies on the results of *existing* algorithms, while it may be more beneficial to account for the noise by adjusting the metamodels used, and/or the algorithmic approach. Furthermore, they assume homogenous noise, which is likely not the case in practice. Stochastic algorithms (such as Binois et al. 2019; Gonzalez et al. 2020) can potentially be useful to determine the number of (extra)replications dynamically during HPO optimization, thus ensuring that computational budget is spent in (re-)evaluating the configuration that yields most information.

Apart from these methodological recommendations, we also outline some general recommendations. To improve the interpretability of the results, we recommend using individual performance measures as objectives in HPO settings, rather than an aggregate measure such as the *F-measure* (combining recall and precision for classification problems Ekbal and Saha 2015, 2016) or the *Area Under the Curve* measure (AUC), which combines the False Positive rate and the True Positive rate. Such aggregated measures reflect a fixed relationship between the individual measures, which may result in solutions that perform really well on the aggregated measure (for instance, the F-measure), but are suboptimal for the individual measures (recall and precision). Moreover, the aggregation of multiple performance measures into a single objective by means of scalarization should be done carefully, as not all scalarization methods (e.g., weighted sum) allow the detection of all parts of the Pareto front. The Augmented Tchebycheff function (Miettinen 2012), for instance, is recommended when the front contains non-convex areas. The nonlinear term in the scalarization function ensures that these areas can be detected, while the linear term ensures that weak Pareto optimal solutions are less rewarded (see Miettinen and Mäkelä 2002 for a further discussion on scalarization functions).

Furthermore, we noticed a surprising lack of detail in the current HPO papers (i.e., in the description of the methodological approaches, the experimental designs, and the corresponding results). To improve the reproducibility of the research, and facilitate comparisons among different HPO algorithms, we recommend a clear description and analysis of four basic elements in each future HPO research paper: (1) search space characteristics (type and range of the considered HPs), (2) algorithmic details (accompanied by pseudocode), (3) description/definition of performance objectives, (4) details on the final optimal solutions obtained for the test problems (optimal HPO configurations, quality metrics for the Pareto front, etc.).

Finally, we noticed that only about half of the papers studied benchmark the algorithm under study w.r.t. other existing algorithms. Such experimental comparisons have

substantial added value for the research community. We therefore clearly advocate their inclusion in future multi-objective HPO research.

Appendix: Pseudocodes of MO-HPO algorithms

The details of the optimization algorithms are provided as a pseudocode. This was obtained from the description included in the papers surveyed and the paper where the algorithm was proposed initially. The function indicating the number of HP configurations evaluated during the optimization is included as part of the heading of the pseudocode.

Pseudocode 1 Metamodel-based optimization $g(n, I, N = 1) = n + I$

Require: n : initial design, I: number of iterations
1: $P \leftarrow \{\}$
2: **for** $i = 1$ to n **do** ▷ **Sample initial HP configurations**
3: $hp \leftarrow$ Generate HP configuration
4: $f_{hp} \leftarrow$ Evaluate performance of hp
5: $P \leftarrow P \cup \{hp, f_{hp}\}$
6: **end for**
7: **for** $i = 1$ to I **do**
8: $M \leftarrow$ Train metamodel using P
9: $new_{hp} \leftarrow$ Obtain a new HP configuration by optimizing an acquisition function using the metamodel predictions
10: $f_{hp} \leftarrow$ Evaluate performance of new_{hp}
11: $P \leftarrow P \cup \{new_{hp}, f_{hp}\}$
12: **end for**
return HP configurations in the Pareto front

Pseudocode 2 NSGA-II, OMOPSO, PSO, SPEA-II, MO-RACACO, CoDeepNEAT, ENS-MOEA/D, MOEA/D, GA with scalarized objectives, and the hybrid algorithm (MLP/CART/RBF/GP) + NSGA-II $g(n, I, N) = n+IN$

Require: n : population size, I: number of iterations, N: number of new configurations per iteration

$P \leftarrow \{\}$
for $i = 1$ to n **do** ▷ **Create initial population**
  $hp \leftarrow$ Generate HP configuration
  $f_{hp} \leftarrow$ Evaluate performance of hp
  $P \leftarrow P \cup \{hp, f_{hp}\}$
end for
for $i = 1$ to I **do**
  for $q = 1$ to N **do** ▷ **Generate new HP configurations**
    $new_{hp} \leftarrow$ Generate HP configuration
    $f_{new_{hp}} \leftarrow$ Evaluate performance of new_{hp}
    $P \leftarrow P \cup \{new_{hp}, f_{new_{hp}}\}$
  end for
  if NSGA-II or CoDeepNeat or GA or hybrid algorithm **then**
    $P \leftarrow$ Select HP configurations considering non-dominated and crowding distance sorting
  end if
  if MOEA/D or ENS-MOEA/D **then**
    $P \leftarrow$ Select HP configurations considering non-dominated sorting
  end if
  if OMOPSO or PSO **then**
    Update velocity and particle position
  end if
  if MO-RACACO **then**
    Update pheromone paths
  end if
end for
  return HP configurations in the Pareto front

Pseudocode 3 Random Forest + ES $g(n, I, N) = n + IN$

Require: n : initial solutions, I: number of iterations, N: number of new configurations per iteration

$P \leftarrow \{\}$
$STP \leftarrow \{\}$ ▷ **Set of observed HP configurations to train the metamodel**
for $i = 1$ to n **do** ▷ **Create initial Population**
    $hp \leftarrow$ Generate an HP configuration
    $f_{hp} \leftarrow$ Evaluate performance of hp
    $P \leftarrow P \cup \{hp, f_{hp}\}$
end for
$STP \leftarrow STP \cup P$
for $i = 1$ to $\frac{I}{2}$ **do** ▷ **Perform Evolutionary Strategy without the metamodel during the first half of the iterations**
    for $q = 1$ to N **do**
        $new_{hp} \leftarrow$ Generate new HP configuration by applying genetic operators
        $f_{new_{hp}} \leftarrow$ Evaluate performance of new_{hp}
        $P \leftarrow P \cup \{hp, f_{new_{hp}}\}$
        $STP \leftarrow STP \cup \{hp, f_{new_{hp}}\}$
    end for
    $P \leftarrow$ Select HP configurations considering non-dominated and crowding distance sorting
end for
for $i = 1$ to $\frac{I}{2}$ **do** ▷ **Perform Evolutionary Strategy using the metamodel during the second half of the iterations**
    for $q = 1$ to N **do**
        Train a metamodel (Random Forest) using STP
        $hp \leftarrow$ Generate a HP configuration using genetic operators and the metamodel prediction
        $f_{hp} \leftarrow$ Evaluate performance of hp
        $P \leftarrow P \cup \{hp, f_{hp}\}$
        $STP \leftarrow STP \cup \{hp, f_{hp}\}$
    end for
    $P \leftarrow$ Select HP configurations considering non-dominated and crowding distance sorting
end for
    return HP configurations in the Pareto front

Pseudocode 4 MODE for scalarized objectives $g(n, I, N = 1) = n + I$

Require: n : initial design, I: number of iterations

$P \leftarrow \{\}$
for $i = 1$ to n **do** ▷ **Sample initial HP configurations**
    $hp \leftarrow$ Generate HP configuration
    $f_{hp} \leftarrow$ Evaluate performance of hp
    $P \leftarrow P \cup \{hp, f_{hp}\}$
end for
for $i = 1$ to I **do**
    $challenger \leftarrow$ Select a random configuration from P
    $a, b, c \leftarrow$ Select three random configurations from P
    $new_{hp} \leftarrow$ Obtain a new HP configuration as a linear combination of a, b, c
    $f_{new_{hp}} \leftarrow$ Evaluate performance of new_{hp}
    if $f_{new_{hp}}$ is better than $f_{challenger}$ **then**
        replace $challenger$ with new_{hp} in P
    end if
end for
    return HP configurations in the Pareto front

Pseudocode 5 ANN + DSE $g(n, I, N = 1) = n + I$

Require: n : initial design, I: number of iterations

$P \leftarrow \{\}$
for $i = 1$ to n **do** ▷ **Sample initial HP configurations**
  $hp \leftarrow$ Generate HP configuration
  $f_{hp} \leftarrow$ Evaluate performance of hp
  $P \leftarrow P \cup \{hp, f_{hp}\}$
end for
$previous \leftarrow null$
$i = 1$
while $i < I$ **do**
  $M \leftarrow$ Train a metamodel (ANN) using P
  $new_{hp} \leftarrow$ Sample the next HP configuration from a Gaussian distribution centered around the previously explored solution (or sample a random configuration if $previous = null$)
  $\widehat{f}_{new_{hp}} \leftarrow$ Predict the performance of new_{hp} using the metamodel
  if new_{hp} is predicted to be Pareto dominated **then**
    Select with certain probability α the configuration $\{new_{hp}\}$ to add to P
  end if
  if new_{hp} is accepted in P **then**
    $previous = \{new_{hp}, f_{new_{hp}}\}$
    $f_{hp} \leftarrow$ Evaluate performance of hp
    $P \leftarrow P \cup \{hp, f_{hp}\}$
    $i+ = 1$
  end if
end while
  return HP configurations in the Pareto front

Pseudocode 6 GP + GA Parsimony $g(n, I, N) = n + I + n + IN = 2n + I(N + 1)$

Require: n : initial design for BO, I: number of iterations, N: number of new configurations per iteration

▷ **HPO using BO and training the ML with all the features**

$P \leftarrow \{\}$
for $i = 1$ to n **do** ▷ **Sample initial HP configurations**
  $hp \leftarrow$ Generate HP configuration
  $f_{hp} \leftarrow$ Evaluate performance of hp
  $P \leftarrow P \cup \{hp, f_{hp}\}$
end for
for $i = 1$ to I **do**
  $M \leftarrow$ Train a metamodel (GP) using P
  $new_{hp} \leftarrow$ Obtain a new HP configuration by optimizing an acquisition function using the metamodel predictions
  $f_{new_{hp}} \leftarrow$ Evaluate performance of new_{hp}
  $P \leftarrow P \cup \{new_{hp}, f_{new_{hp}}\}$
end for

▷ **Feature selection using the best model HPs using a Genetic Algorithm**

$T \leftarrow \{\}$
for $i = 1$ to n **do** ▷ **Create initial population**
  $hpf \leftarrow$ Select HP configuration from P and select a set of features of the ML problem
  $f_{hpf} \leftarrow$ Evaluate performance of hpf
  $T \leftarrow T \cup \{hpf, f_{hpf}\}$
end for
for $i = 1$ to I **do**
  $new_{hp1}, new_{hp2} \leftarrow$ Generate two HP configurations by applying genetic operators in two random HP configurations $hp1, hp2$ selected from T
  $f_{hp1}, f_{hp2} \leftarrow$ Evaluate performance of new_{hp1}, new_{hp2}
  $T \leftarrow T \cup \{\{new_{hp1}, f_{hp1}\}, \{new_{hp2}, f_{hp2}\}\}$
  Reduce T to keep the same population size on each iteration
end for
return HP configurations in the Pareto front

Pseudocode 7 CMA-ES for MOO $g(n = 0, I, N) = IN$

Require: I: number of iterations, N: number of new configurations per iteration

Create a multivariate normal distribution $\mathcal{N}(.)$ of k hyperparameters configurations
for $i = 1$ to I **do**
  $\{hp_1, \cdots, hp_N\} \leftarrow \mathcal{N}(.)$ ▷ **Sample N candidates from** $\mathcal{N}(.)$
  $\{f_{hp_1}, \cdots, f_{hp_N}\} \leftarrow$ Evaluate performance of each HP configuration
  Keep only the l highest/lowest from $\{hp_1, \cdots, hp_N\}$ configurations using their non-dominating sort
  Update the multivariate normal distribution $\mathcal{N}(.)$ using the selected HP configurations
end for
return HP configurations in the Pareto front

Pseudocode 8 Multi-objective SA $g(n = 0, I, N) = IN$

Require: I: number of iterations, N: number of new configurations per iteration

for $i = 1$ to I **do**
  for $q = 1$ to N **do**
    $hp \leftarrow$ Generate a new HP configuration from the neighborhood of the current best HP configuration X. A random configuration is used at the beginning of the optimization
    $f_{hp} \leftarrow$ Evaluate performance of hp
    Determine if hp can be considered as the current best solution X (acceptance rule defined to consider the dominance of hp over X and the configurations in an external archive of non-dominated solutions)
    Update, if needed, the external archive of non-dominated solutions
    Update the "*temperature*" of the system
  end for
end for
return HP configurations in the Pareto front

Pseudocode 9 Nelder Mead algorithm with scalarized objectives $g(n, I, N) = n + 1 + I(N + 1)$

Require: n : initial solutions, I: number of iterations, N: number of new configurations per iteration

$S \leftarrow \{\}$
for $i = 1$ to $n + 1$ **do** ▷ **Create initial Simplex**
  $hp \leftarrow$ Generate an HP configuration
  $f_{hp} \leftarrow$ Evaluate performance of hp
  $S \leftarrow S \cup \{hp, f_{hp}\}$
end for
for $i = 1$ to I **do**
  Sort vertices in S in descending order
  $Sc \leftarrow$ Compute centroid vertex without the worst vertex
  $Sr \leftarrow$ Compute reflection of Sc
  $f_{Sr} \leftarrow$ Evaluate performance of Sr
  if f_{Sr} is between the best and worst solution (excluding them) **then**
    Replace the worst solution in S with $\{Sr, f_{sr}\}$
  else if f_{Sr} is better than the best solution **then**
    $Se \leftarrow$ Expand using Sr and Sc
    $f_{Se} \leftarrow$ Evaluate performance of Se
    Replace the worst solution with Se if this is better than Sr. Otherwise, use Sr
  else if f_{Sr} is worst than the current worst solution **then**
    $Scr \leftarrow$ Contract using Sr and Sc
    $f_{Scr} \leftarrow$ Evaluate performance of Scr
    if f_{Scr} is better than f_{Sr} **then**
      Replace the worst solution with Scr
    else ▷ **Shrink toward the best solution**
      for $j = 2$ to $n + 1$ **do**
        $Si \leftarrow$ shrink vertex i
        $f_{Si} \leftarrow$ Evaluate performance of Si
      end for
    end if
  end if
end for
  return HP configurations in the Pareto front

Pseudocode 10 MADE $g(n, I, N) = n + 2IN$

Require: n : initial solutions, I: number of iterations, N: number of new configurations per iteration

$P \leftarrow \{\}$
for $i = 1$ to n **do** ▷ **Create initial Population**
  $hp \leftarrow$ Generate an HP configuration
  $f_{hp} \leftarrow$ Evaluate performance of hp
  $P \leftarrow P \cup \{hp, f_{hp}\}$
end for
for $i = 1$ to I **do**
  for $q = 1$ to N **do** ▷ **Generate new HP configurations**
    $new_{hp} \leftarrow$ Generate a new HP configuration by applying mutation and crossover operators, using a random configuration Rc selected from P
    $f_{new_{hp}} \leftarrow$ Evaluate performance of new_{hp}
    Replace Rc with new_{hp} if new_{hp} dominates Rc
  end for
  $S \leftarrow$ select non-dominated HP configurations
  for $q = 1$ to $\mid S \mid$ **do** ▷ **Perform a chaotic local search around the solutions in the Pareto front**
    $hp_{cls} \leftarrow$ Generate a new HP configuration around S_q
    $f_{hp_{cls}} \leftarrow$ Evaluate performance of hp_{cls}
    $P \leftarrow P \cup \{hp_{cls}, f_{hp_{cls}}\}$
  end for
  $P \leftarrow$ Select HP configurations considering non-dominated and crowding distance sorting
end for
return HP configurations in the Pareto front

Acknowledgements This work was supported by the Flanders Artificial Intelligence Research Program (FLAIR), and by the Research Foundation Flanders (FWO Grant 1216021N). The authors would like to thank Gonzalo Nápoles from Tilburg University for his comments on a previous version of this paper.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

References

Abdolsh M, Shilton A, Rana S, Gupta S, Venkatesh S (2019) Multi-objective Bayesian optimisation with preferences over objectives. Advances in neural information processing systems pp 12235–12245

Ab Wahab MN, Nefti-Meziani S, Atyabi A (2015) A comprehensive review of swarm optimization algorithms. PLoS ONE 10(5):e0122827

Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp 2623–2631

Alaya I, Solnon C, Ghedira K (2007) Ant colony optimization for multi-objective optimization problems. In: 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007) vol 1, pp 450–457. https://doi.org/10.1109/ICTAI.2007.108

Albelwi S, Mah A (2016) Automated optimal architecture of deep convolutional neural networks for image recognition. In: 2016 15th IEEE international conference on machine learning and applications (icmla) pp 53–60. https://doi.org/10.1109/ICMLA.2016.0018

Andreopoulos A, Tsotsos JK (2013) 50 years of object recognition: directions forward. Comput Vis Image Understand 117(8):827–891. https://doi.org/10.1016/j.cviu.2013.04.005

Babu B, Gujarathi AM (2007) Multi-objective differential evolution (mode) algorithm for multi-objective optimization: parametric study on benchmark test problems. J Future Eng Technol 3(1):47–59. https://doi.org/10.26634/jfet.3.1.697

Baldeon M, Lai-Yuen SK (2020) Adaresu-net: multiobjective adaptive convolutional neural network for medical image segmentation. Neu-rocomputing 392:325–340. https://doi.org/10.1016/j.neucom.2019.01.110

Barán B, Schaerer M (2003) A multiobjective ant colony system for vehicle routing problem with time windows. Applied informatics pp 97–102

Belgiu M, Drăguţ L (2016) Random forest in remote sensing: a review of applications and future directions. ISPRS J Photogramm Remote Sens 114:24–31

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: 25th annual conference on neural information processing systems (NIPS 2011) vol 24

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(1):281–305

Bergstra J, Yamins D, Cox D (2013) Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision archi-tectures. In: International conference on machine learning, pp 115–123. http://proceedings.mlr.press/v28/bergstra13.html

Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ (2009) A survey on metaheuristics for stochastic combinatorial optimization. Natural Comput 8(2):239–287. https://doi.org/10.1007/s11047-008-9098-4

Binder M, Moosbauer J, Thomas J, Bischl B (2020) Multi-objective hyperparameter tuning and feature selection using filter ensembles. vol 1050, p 13. https://doi.org/10.1145/3377930.3389815

Binois M, Huang J, Gramacy RB, Ludkovski M (2019) Replication or exploration? Sequential design for stochastic simulation experiments. Technometrics 61(1):7–23. https://doi.org/10.1080/00401706.2018.1469433

Bischl B, Mersmann O, Trautmann H, Weihs C (2012) Resampling methods for meta-model validation with recommendations for evolutionary computation. Evol Comput 20(2):249–275. https://doi.org/10.1162/EVCO_a_00069

Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1987) Occam's razor. Inf Process Lett 24(6):377–380. https://doi.org/10.1016/0020-0190(87)90114-1

Bouraoui A, Jamoussi S, BenAyed Y (2018) A multi-objective genetic algorithm for simultaneous model and feature selection for support vector machines. Artif Intell Rev 50(2):261–281. https://doi.org/10.1007/s10462-017-9543-9

Bui K-HN, Yi H (2020) Optimal hyperparameter tuning using meta-learning for big traffic datasets. In: Lee W et al. (ed) 2020 IEEE international conference on big data and smart computing (bigcomp 2020) pp 48–54. IEEE. https://doi.org/10.1109/BigComp48618.2020.0-100

Cai X, Hu Z, Zhao P, Zhang W, Chen J (2020) A hybrid recommendation system with many-objective evolutionary algorithm. Expert Syst Appl 159:113648. https://doi.org/10.1016/j.eswa.2020.113648

Calisto MB, Lai-Yuen SK (2020) Adaen-net: an ensemble of adaptive 2d–3d fully convolutional networks for medical image segmentation. Neural Netw. https://doi.org/10.1016/j.neunet.2020.03.007

Calisto MB, Lai-Yuen SK (2021) Emonas-net: efficient multiobjective neural architecture search using surrogate-assisted evolutionary algorithm for 3d medical image segmentation. Artif Intell Med 119:102154. https://doi.org/10.1016/j.artmed.2021.102154

Chandra A, Lane I (2016) Automated optimization of decoder hyper-parameters for online lvcsr. In: 2016 IEEE spoken language technol-ogy workshop (slt) pp 454–460. https://doi.org/10.1109/SLT.2016.7846303

Chatelain C, Adam S, Lecourtier Y, Heutte L, Paquet T (2007) Multi-objective optimization for svm model selection. In: Ninth international conference on document analysis and recognition (ICDAR 2007) vol 1, pp 427–431. https://doi.org/10.1109/ICDAR.2007.4378745

Chen J, Li K, Tang Z, Bilal K, Yu S, Weng C, Li K (2016) A parallel random forest algorithm for big data in a spark cloud computing environment. IEEE Trans Parallel Distrib Syst 28(4):919–933

Chen W-C, Jiang X-Y, Chang H-P, Chen H-P (2014) An effective system for parameter optimization in photolithography process of a lgp stamper. Neural Comput Appl 24(6):1391–1401. https://doi.org/10.1007/s00521-013-1353-7

Chin T-W, Morcos AS, Marculescu D (2020) Pareco: Pareto-aware channel optimization for slimmable neural networks. In: 2nd Workshop on Adversarial Learning Methods for Machine Learning and Data Mining, KDD'2020. https://openreview.net/forum?id=SPyxaz%5Fh9Nd

Cho H, Kim Y, Lee E, Choi D, Lee Y, Rhee W (2020) Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks. IEEE Access 8:52588–52608. https://doi.org/10.1109/ACCESS.2020.2981072

Cooney C, Korik A, Folli R, Coyle D (2020) Evaluation of hyperparameter optimization in machine and deep learning methods for decoding imagined speech EEG. Sensors 20(16):4629. https://doi.org/10.3390/s20164629

Cowen-Rivers AI, Lyu W, Wang Z, Tutunov R, Jianye H, Wang J, Ammar HB (2020) Hebo: heteroscedastic evolutionary bayesian optimisation. Workshop at NeurIPS 2020 Competition Track on Black-Box Optimization Challenge

Dai Z, Damianou A, Hensman J, Lawrence N (2014) Gaussian process models with parallelization and GPU acceleration. arXiv:1410.4984

Dai Z, Yu H, Low BKH, Jaillet P (2019) Bayesian optimization meets bayesian optimal stopping. International conference on machine learning. pp 1496–1506. http://proceedings.mlr.press/v97/dai19a.html

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197. https://doi.org/10.1109/4235.996017

Deighan DS, Field SE, Capano CD, Khanna G (2021) Genetic-algorithm-optimized neural networks for gravitational wave classification. Neural Comput Appl. https://doi.org/10.1007/s00521-021-06024-4

de Toro F, Ros E, Mota S, Ortega J (2002) Multi-objective optimization evolutionary algorithms applied to paroxysmal atrial fibrillation diagnosis based on the k-nearest neighbours classifier. Ibero-american conference on artificial intelligence pp 313–318. https://doi.org/10.1007/3-540-36131-6_32

Doerner K, Gutjahr WJ, Hartl RF, Strauss C, Stummer C (2004) Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection. Ann Oper Res 131(1):79–99. https://doi.org/10.1023/B:ANOR.0000039513.99038.c6

Doerr C (2020) Complexity theory for discrete black-box optimization heuristics. Theory of evolutionary computation. Springer pp 133–212

Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2–3):243–278. https://doi.org/10.1016/j.tcs.2005.05.020

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 26(1):29–41. https://doi.org/10.1109/3477.484436

Durillo JJ, Nebro AJ, Luna F, Alba E (2008) A study of master-slave approaches to parallelize nsga-ii. In: 2008 IEEE international symposium on parallel and distributed processing pp 1–8

Dutta S, Gandomi AH (2020) Surrogate model-driven evolutionary algorithms: theory and applications. Evolution in action: past, present and future. Springer. pp 435–451. https://doi.org/10.1007/978-3-030-39831-6_29

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. Mhs'95. In: Proceedings of the sixth international symposium on micro machine and human science. pp 39–43. https://doi.org/10.1109/MHS.1995.494215

Ekbal A, Saha S (2015) Joint model for feature selection and parameter optimization coupled with classifier ensemble in chemical mention recog-nition. Knowl-Based Syst 85:37–51. https://doi.org/10.1016/j.knosys.2015.04.015

Ekbal A, Saha S (2016) Simultaneous feature and parameter selection using multiobjective optimization: application to named entity recogni-tion. Int J Mach Learn Cybern 7(4):597–611. https://doi.org/10.1007/s13042-014-0268-7

Emmerich MT, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Natural Comput 17(3):585–609

Emmerich MT, Deutz AH, Klinkenberg JW (2011) Hypervolume-based expected improvement: Monotonicity properties and exact computation. In: 2011 IEEE congress of evolutionary computation (CEC) pp 2147–2154. https://doi.org/10.1109/CEC.2011.5949880

Ertel W (2018) Introduction to artificial intelligence. Springer, Cham

Falkner S, Klein A, Hutter F (2018) Bohb: robust and efficient hyper-parameter optimization at scale. In: International conference on machine learning pp 1437–1446. http://proceedings.mlr.press/v80/falkner18a.html

Faris H, Habib M, Faris M, Alomari M, Alomari A (2020) Medical speciality classification system based on binary particle swarms and ensemble of one vs rest support vector machines. J Biomed Inform 109:103525. https://doi.org/10.1016/j.jbi.2020.103525

Feliot P, Bect J, Vazquez E (2017) A bayesian approach to constrained single-and multi-objective optimization. J Glob Optim 67(1–2):97–133. https://doi.org/10.1007/s10898-016-0427-3

Feurer M, Hutter F (2019) Hyperparameter optimization. Automated machine learning: methods, systems, challenges. Springer, Cham, pp 3–33

Fieldsend JE, Everson RM (2014) The rolling tide evolutionary algorithm: a multiobjective optimizer for noisy optimization problems. IEEE Trans Evol Comput 19(1):103–117. https://doi.org/10.1109/TEVC.2014.2304415

Forrester AI, Keane AJ, Bressloff NW (2006) Design and analysis of "noisy'' computer experiments. AIAA J 44(10):2331–2339. https://doi.org/10.2514/1.20068

Gad AG (2022) Particle swarm optimization algorithm and its applications: a systematic review. Arch Comput Methods Eng 8:1–31

Garrido EC, Hernández D (2019) Predictive entropy search for multi-objective Bayesian optimization with constraints. Neurocomputing 361:50–68. https://doi.org/10.1016/j.neucom.2019.06.025

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Gonzalez SR, Jalali H, Van Nieuwenhuyse I (2020) A multiobjective stochastic simulation optimization algorithm. Eur J Oper Res 284(1):212–226. https://doi.org/10.1016/j.ejor.2019.12.014

Gravel M, Price WL, Gagné C (2002) Scheduling continuous casting of aluminum using a multiple objective ant colony optimization meta-heuristic. Eur J Oper Res 143(1):218–229. https://doi.org/10.1016/S0377-2217(01)00329-0

Gülcü A, Kuş Z (2021) Multi-objective simulated annealing for hyper-parameter optimization in convolutional neural networks. PeerJ Comput Sci 7:e338. https://doi.org/10.7717/peerj-cs.338

Guo C, Li L, Hu Y, Yan J (2020) A deep learning based fault diagnosis method with hyperparameter optimization by using parallel computing. IEEE Access 8:131248–131256. https://doi.org/10.1109/ACCESS.2020.3009644

Guo J, Yang L, Bie R, Yu J, Gao Y, Shen Y, Kos A (2019) An xgboost-based physical fitness evaluation model using advanced feature selection and bayesian hyper-parameter optimization for wearable running monitoring. Comput Netw 151:166–180. https://doi.org/10.1016/j.comnet.2019.01.026

Gupta S, Shilton A, Rana S, Venkatesh S (2018) Exploiting strategy-space diversity for batch bayesian optimization. In: International conference on artificial intelligence and statistics pp 538–547. http://proceedings.mlr.press/v84/gupta18a.html

Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in neural information processing systems vol 28, pp 1135-1143. Curran Associates, Inc. https://dl.acm.org/doi/10.5555/2969239.2969366

Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11(1):1–18. https://doi.org/10.1162/106365603321828970

Hegde S, Mundada MR (2020) Early prediction of chronic disease using an efficient machine learning algorithm through adaptive probabilistic divergence based feature selection approach. Int J Pervas Comput Commun. https://doi.org/10.1108/IJPCC-04-2020-0018

Hernández D, Hernandez-Lobato J, Shah A, Adams R (2016) Predictive entropy search for multi-objective bayesian optimization. In: International conference on machine learning pp 1492–1501. http://proceedings.mlr.press/v48/hernandez-lobatoa16.html

Hernández-Lobato JM, Gelbart MA, Reagen B, Adolf R, Hernández-Lobato D, Whatmough PN, Adams RP (2016) Designing neural network hardware accelerators with decoupled objective evaluations. Nips workshop on bayesian optimization. p 10

Ho TK (1995) Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition vol 1, pp 278–282. https://doi.org/10.1109/ICDAR.1995.598994

Horn D, Bischl B (2016) Multi-objective parameter configuration of machine learning algorithms using model-based optimization. In: 2016 IEEE symposium series on computational intelligence (SSCI) pp 1–8. https://doi.org/10.1109/SSCI.2016.7850221

Horn D, Dagge M, Sun X, Bischl B (2017) First investigations on noisy model-based multi-objective optimization. International conference on evolutionary multi-criterion optimization pp 298–313. https://doi.org/10.1007/978-3-319-54157-0_21

Hsu C-H, Juang C-F (2013) Multi-objective continuous-ant-colony-optimized fc for robot wall-following control. IEEE Comput Intell Mag 8(3):28–40. https://doi.org/10.1109/MCI.2013.2264233

Hu W, Jin J, Liu T-Y, Zhang C (2019) Automatically design convolutional neural networks by optimization with submodularity and supermodularity. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2019.2939157

Hutter F, Hoos HH, Leyton-Brown K, Stützle T (2009) Paramils: an automatic algorithm configuration framework. J Artif Intell Res 36:267–306. https://doi.org/10.1613/jair.2861

Hutter F, Lücke J, Schmidt-Thieme L (2015) Beyond manual tuning of hyperparameters. KI-Künstliche Intelligenz 29(4):329–337. https://doi.org/10.1007/s13218-015-0381-0

Iredi S, Merkle D, Middendorf M (2001) Bi-criterion optimization with multi colony ant algorithms. International conference on evolutionary multi-criterion optimization pp 359–372. https://doi.org/10.1007/3-540-44719-9_25

Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, et al (2017) Population based training of neural networks. arXiv:1711.09846

Jalali H, Van Nieuwenhuyse I, Picheny V (2017) Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. Eur J Oper Res 261(1):279–301. https://doi.org/10.1016/j.ejor.2017.01.035

Jiang F, Jiang Y, Zhi H, Dong Y, Li H, Ma S, Wang Y (2017) Artificial intelligence in healthcare: past, present and future. Stroke Vasc Neurol 2(4):230–243. https://doi.org/10.1136/svn-2017-000101

Jin Y (2011) Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol Comput 1(2):61–70. https://doi.org/10.1016/j.swevo.2011.05.001

Jing W, Lin J, Wang H (2020) Building nas: Automatic designation of efficient neural architectures for building extraction in high-resolution aerial images. IMAGE AND VISION COMPUTING 103. https://doi.org/10.1016/j.imavis.2020.104025

Jomaa HS, Grabocka J, Schmidt-Thieme L (2019) Hyp-rl: Hyper-parameter optimization by reinforcement learning. arXiv:1906.11527

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Global Optim 13(4):455–492. https://doi.org/10.1023/A:1008306431147

Joorabian M, Afzalan E (2014) Optimal power flow under both normal and contingent operation conditions using the hybrid fuzzy particle swarm optimisation and nelder-mead algorithm (hfpso-nm). Appl Soft Comput 14:623–633

Juang C-F (2002) A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. IEEE Trans Fuzzy Syst 10(2):155–170. https://doi.org/10.1109/91.995118

Juang C-F, Hsu C-H (2014) Structure and parameter optimization of fnns using multi-objective aco for control and prediction. In: 2014 ieee international conference on fuzzy systems (fuzz-ieee) pp 928–933. https://doi.org/10.1109/FUZZ-IEEE.2014.6891545

Karnin Z, Koren T, Somekh O (2013) Almost optimal exploration in multi-armed bandits. International conference on machine learning pp 1238–1246

Kim Y, Reddy B, Yun S, Seo C (2017) Nemo: Neuro-evolution with mul-tiobjective optimization of deep neural network for speed and accuracy. Icml 2017 automl workshop

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680. https://doi.org/10.1126/science.220.4598.671

Knowles J (2006) Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans Evol Comput 10(1):50–66. https://doi.org/10.1109/TEVC.2005.851274

Koch P, Wagner T, Emmerich MT, Bäck T, Konen W (2015) Efficient multi-criteria optimization on noisy machine learning problems. Appl Soft Comput 29:357–370. https://doi.org/10.1016/j.asoc.2015.01.005

Kohavi R, John GH (1995) Automatic parameter selection by minimizing estimated error. Machine learning proceedings 1995 pp 304–312. Elsevier. https://doi.org/10.1016/B978-1-55860-377-6.50045-1

Kong W, Dong ZY, Luo F, Meng K, Zhang W, Wang F, Zhao X (2017) Effect of automatic hyperparameter tuning for residential load forecasting via deep learning. 2017 australasian universities power engineering conference (aupec) (pp 1–6). https://doi.org/10.1109/AUPEC.2017.8282478

Kuncheva LI (2014) Combining pattern classifiers: methods and algorithms. Wiley, Hoboken
Laskaridis S, Venieris SI, Kim H, Lane ND (2020) Hapi: hardware-aware progressive inference. In: 2020 ieee/acm international conference on computer aided design (iccad) (pp 1–9)
Laumanns M, Thiele L, Deb K, Zitzler E (2002) Combining convergence and diversity in evolutionary multiobjective optimization. Evol Comput 10(3):263–282. https://doi.org/10.1162/106365602760234108
León J, Ortega J, Ortiz A (2019) Convolutional neural networks and feature selection for bci with multiresolution analysis. International work-conference on artificial neural networks (pp 883-894). https://doi.org/10.1007/978-3-030-20521-8_72
Li M, Yao X (2019) Quality evaluation of solution sets in multiobjective optimisation: a survey. ACM Comput Surv (CSUR) 52(2):1–38. https://doi.org/10.1145/3300148
Li H, Zhang Q, Tsang E, Ford JA (2004) Hybrid estimation of distribution algorithm for multiobjective knapsack problem. J. Gottlieb & G.R. Raidl (Eds.), Evolutionary computation in combinatorial optimization (pp 145–154). https://doi.org/10.1007/978-3-540-24652-7_15
Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2017) Hyperband: a novel bandit-based approach to hyperparameter optimization. J Mach Learn Res 18(1):6765–6816
Li S, Gong W, Yan X, Hu C, Bai D, Wang L (2019) Parameter estimation of photovoltaic models with memetic adaptive differential evolution. Sol Energy 190:465–474. https://doi.org/10.1016/j.solener.2019.08.022
Liang J, Meyerson E, Hodjat B, Fink D, Mutch K, Miikkulainen R (2019) Evolutionary neural automl for deep learning. Proceedings of the genetic and evolutionary computation conference (pp 401–409). https://doi.org/10.1145/3321707.3321721
Liu H, Cai J, Ong Y-S (2018) Remarks on multi-output gaussian process regression. Knowl-Based Syst 144:102–121. https://doi.org/10.1016/j.knosys.2017.12.034
Liu J, Tunguz B, Titericz G (2020) GPU accelerated exhaustive search for optimal ensemble of black-box optimization algorithms. Workshop at NeurIPS 2020 Competition Track on Black-Box Optimization Challenge
Loni M, Zoljodi A, Sinaei S, Daneshtalab M, Sjödin M (2019) Neuropower: Designing energy efficient convolutional neural network architecture for embedded systems. International conference on artificial neural networks (pp 208-222). https://doi.org/10.1007/978-3-030-30487-4_17
Loni M, Sinaei S, Zoljodi A, Daneshtalab M, Sjödin M (2020) Deepmaker: a multi-objective optimization framework for deep neural networks in embedded systems. Microprocess Microsyst 73:102989. https://doi.org/10.1016/j.micpro.2020.102989
López-Ibáñez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stützle T (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 3:43–58. https://doi.org/10.1016/j.orp.2016.09.002
Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W (2019) Nsga-net: neural architecture search using multi-objective ge-netic algorithm. Proceedings of the genetic and evolutionary computation conference (pp 419–427). https://doi.org/10.1145/3321707.3321729
Lu Z, Deb K, Goodman E, Banzhaf W, Boddeti VN (2020) Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. European conference on computer vision (pp 35–51). https://doi.org/10.1007/978-3-030-58452-8_3
Luo G (2016) A review of automatic selection methods for machine learning algorithms and hyperparameter values. Netw Model Anal Health Inform Bioinform 5(1):18. https://doi.org/10.1007/s13721-016-0125-6
Magda M, Martinez-Alvarez A, Cuenca-Asensi S (2017) Mooga parameter optimization for onset detection in emg signals. International conference on image analysis and processing (pp 171–180). https://doi.org/10.1007/978-3-319-70742-6_16
Makarova A, Shen H, Perrone V, Klein A, Faddoul JB, Krause A, Archambeau C (2021) Overfitting in bayesian optimization: an empirical study and early-stopping solution. https://www.amazon.science/publications/overfitting-in-bayesian-optimization-an-empirical-study-and-early-stopping-solution
Martinez-de Pison FJ, Gonzalez-Sendino R, Aldama A, Ferreiro J, Fraile E (2017) Hybrid methodology based on bayesian optimization and ga-parsimony for searching parsimony models by combining hyperparameter optimization and feature selection. International conference on hybrid artificial intelligence systems (pp 52-62). https://doi.org/10.1016/j.neucom.2018.05.136
McKinnon KI (1998) Convergence of the nelder-mead simplex method to a nonstationary point. SIAM J Optim 9(1):148–158. https://doi.org/10.1137/S1052623496303482
Mei J, Li Y, Lian X, Jin X, Yang L, Yuille A, Yang J (2020) Atomnas: Fine-grained end-to-end neural architecture search. International conference on learning representations. https://openreview.net/forum?id=BylQSxHFwr

Meinshausen N, Ridgeway G (2006) Quantile regression forests. J Mach Learn Res 7(6). http://jmlr.org/papers/v7/meinshausen06a.html
Mentch L, Hooker G (2016) Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. J Mach Learn Res 17(1):841–881
Miettinen K (2012) Nonlinear multiobjective optimization. Springer, Cham
Miettinen K, Mäkelä MM (2002) On scalarizing functions in multiobjective optimization. OR Spectrum 24(2):193–213. https://doi.org/10.1007/s00291-001-0092-9
Mitchell M (1998) An introduction to genetic algorithms. MIT Press, Cambridge
Mitchell TM et al (1997) Machine learning. Burr Ridge 45(37):870–877
Montgomery DC (2017) Design and analysis of experiments. Wiley, Hoboken
Mostafa SS, Mendonça F, Ravelo-Garcia A, Julia-Serda G, Morgado-Dias F (2020) Multi-objective hyperparameter optimization of convolutional neural network for obstructive sleep apnea detection. IEEE Access 8:129586–129599. https://doi.org/10.1109/ACCESS.2020.3009149
Nabil M, Mahmoud M, Ismail M, Serpedin E (2019) Deep recurrent electricity theft detection in ami networks with evolutionary hyper-parameter tuning. 2019 international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata) (pp 1002–1008)
Negrinho R, Gormley M, Gordon GJ, Patil D, Le N, Ferreira D (2019) Towards modular and programmable architecture search. Advances in neural information processing systems (pp 13715-13725). https://dl.acm.org/doi/abs/10.5555/3454287.3455517
Olsson DM, Nelson LS (1975) The Nelder-mead simplex procedure for function minimization. Technometrics 17(1):45–51. https://doi.org/10.1080/00401706.1975.10489269
Ounpraseuth ST (2008) Gaussian processes for machine learning. Taylor & Francis, Milton Park
Ozaki Y, Tanigaki Y, Watanabe S, Onishi M (2020) Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. Proceedings of the 2020 genetic and evolutionary computation conference (pp 533–541). https://doi.org/10.1145/3377930.3389817
Parker-Holder J, Nguyen V, Roberts SJ (2020) Provably efficient online hyperparameter optimization with population-based bandits. Adv Neural Inf Process Syst 33:17200–17211
Parsa M, Ankit A, Ziabari A, Roy K (2019) Pabo: Pseudo agent-based multi-objective bayesian hyperparameter optimization for efficient neural accelerator design. 2019 ieee/acm international conference on computer-aided design (iccad) (pp 1-8). https://doi.org/10.1109/ICCAD45719.2019.8942046
Pathak Y, Shukla PK, Arya K (2020) Deep bidirectional classification model for covid-19 disease infected patients. IEEE/ACM Trans Comput Biol Bioinf. https://doi.org/10.1109/TCBB.2020.3009859
Phillips PJ, Flynn PJ, Scruggs T, Bowyer KW, Chang J, Hoffman K, Worek W (2005) Overview of the face recognition grand challenge. 2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05) (Vol. 1, pp 947-954). https://doi.org/10.1109/CVPR.2005.268
Picheny V (2014) A stepwise uncertainty reduction approach to constrained global optimization. Artificial intelligence and statistics (pp 787–795). http://proceedings.mlr.press/v33/picheny14.html
Ponweiser W, Wagner T, Biermann D, Vincze M (2008) Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection. International conference on parallel problem solving from nature (pp 784-794). https://doi.org/10.1007/978-3-540-87700-4_78
Provost F, Jensen D, Oates T (1999) Efficient progressive sampling. Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining (pp 23-32). https://doi.org/10.1145/312129.312188
Qin AK, Huang VL, Suganthan PN (2008) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417. https://doi.org/10.1109/TEVC.2008.927706
Qin H, Shinozaki T, Duh K (2017) Evolution strategy based automatic tuning of neural machine translation systems. Proceeding of international workshop on spoken language translation (iwslt) (pp 120-128)
Rajagopal A, Joshi GP, Ramachandran A, Subhalakshmi R, Khari M, Jha S, You J (2020) A deep learning model based on multi-objective particle swarm optimization for scene classification in unmanned aerial vehicles. IEEE Access 8:135383–135393. https://doi.org/10.1109/ACCESS.2020.3011502
Richter J, Kotthaus H, Bischl B, Marwedel P, Rahnenführer J, Lang M (2016) Faster model-based optimization through resource-aware scheduling strategies. International conference on learning and intelligent optimization (pp 267-273). https://doi.org/10.1007/978-3-319-50349-3_22
Rojas-Gonzalez S, Van Nieuwenhuyse I (2020) A survey on kriging-based infill algorithms for multiobjective simulation optimization. Comput Oper Res 116:104869. https://doi.org/10.1016/j.cor.2019.104869
Russell S, Norvig P (2010) Artificial intelligence: a modern approach, 3rd edn. Prentice Hall, Hoboken

Salt L, Howard D, Indiveri G, Sandamirskaya Y (2019) Parameter optimization and learning in a spiking neural network for uav obstacle avoidance targeting neuromorphic processors. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2019.2941506

Sanz-García A, Fernández-Ceniceros J, Antonanzas-Torres F, Pernia-Espinoza A, Martinez-De-Pison F (2015) Ga-parsimony: a ga-svr approach with feature selection and parameter optimization to obtain parsimonious solutions for predicting temperature settings in a continuous annealing furnace. Appl Soft Comput 35:13–28. https://doi.org/10.1016/j.asoc.2015.06.012

Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. Proceedings of the 1st international conference on genetic algorithms (p. 93-100). USA: L. Erlbaum Associates Inc

Shah A, Ghahramani Z (2016) Pareto frontier learning with expensive correlated objectives. International conference on machine learning (pp 1919-1927). http://proceedings.mlr.press/v48/shahc16.html

Shimizu H, Toyoda M (2021) Cma-es with coordinate selection for high-dimensional and ill-conditioned functions. Proceedings of the genetic and evolutionary computation conference companion (pp 209–210)

Shinozaki T, Watanabe S, Duh K (2020) Automated development of dnn based spoken language systems using evolutionary algorithms. Deep neural evolution (pp 97-129). Springer. https://doi.org/10.1007/978-981-15-3685-4_4

Sierra MR, Coello CAC (2005) Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. International conference on evolutionary multi-criterion optimization (pp 505-519). https://doi.org/10.1007/978-3-540-31880-4_35

Silva LF, Santos AAS, Bravo RS, Silva AC, Muchaluat-Saade DC, Conci A (2016) Hybrid analysis for indicating patients with breast cancer using temperature time series. Comput Methods Programs Biomed 130:142–153. https://doi.org/10.1016/j.cmpb.2016.03.002

Singh D, Kumar V, Kaur M (2020) Classification of covid-19 patients from chest ct images using multi-objective differential evolution-based convolutional neural networks. Eur J Clin Microbiol Infect Dis. https://doi.org/10.1007/s10096-020-03901-z

Sjöberg A, Önnheim M, Gustavsson E, Jirstrand M (2019) Architecture-aware bayesian optimization for neural network tuning. International conference on artificial neural networks (pp 220-231). https://doi.org/10.1007/978-3-030-30484-3_19

Smithson SC, Yang G, Gross WJ, Meyer BH (2016) Neural networks designing neural networks: multi-objective hyper-parameter optimization. Proceedings of the 35th international conference on computer-aided design (pp 1-8). https://doi.org/10.1145/2966986.2967058

Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25

Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046

Sopov E, Ivanov I (2015) Self-configuring ensemble of neural network classifiers for emotion recognition in the intelligent human-machine in-teraction. 2015 ieee symposium series on computational intelligence (pp 1808-1815). https://doi.org/10.1109/SSCI.2015.252

Srinivas N, Deb K (1994) Multiobjective optimization using nondom-inated sorting in genetic algorithms. Evol Comput 2(3):221–248. https://doi.org/10.1162/evco.1994.2.3.221

Stamoulis D, Cai E, Juan D-C, Marculescu D (2018) Hyperpower: Power-and memory-constrained hyper-parameter optimization for neural networks. 2018 design, automation & test in europe conference & exhibition (date) (pp 19-24). https://doi.org/10.23919/DATE.2018.8341973

Stamoulis D, Chin T-W, Prakash AK, Fang H, Sajja S, Bognar M, Marculescu D (2018) Designing adaptive neural networks for energy-constrained image classification. Proceedings of the international conference on computer-aided design (pp 1-8). https://doi.org/10.1145/3240765.3240796

Storn R, Price K (1997) Differential evolution -a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359. https://doi.org/10.1023/A:1008202821328

Swersky K, Snoek J, Adams RP (2013) Multi-task bayesian optimization. Advances in neural information processing systems, 26

Swersky K, Snoek J, Adams RP (2014) Freeze-thaw bayesian optimization. arXiv:1406.3896

Talbi E-G (2021) Automated design of deep neural networks: a survey and unified taxonomy. ACM Comput Surv (CSUR) 54(2):1–37. https://doi.org/10.1145/3439730

Tanabe R, Fukunaga A (2013) Success-history based parameter adap-tation for differential evolution. 2013 ieee congress on evolutionary computation (pp 71-78). https://doi.org/10.1109/CEC.2013.6557555

Tanaka T, Moriya T, Shinozaki T, Watanabe S, Hori T, Duh K (2016) Automated structure discovery and parameter tuning of neural network language model based on evolution strategy. 2016 ieee spoken language technology workshop (slt) (pp 665-671). https://doi.org/10.1109/SLT.2016.7846334
Tharwat A (2020) Classification assessment methods. Appl Comput Inf. https://doi.org/10.1016/j.aci.2018.08.003
Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining (pp 847-855). https://doi.org/10.1145/2487575.2487629
Tripathy R, Bilionis I, Gonzalez M (2016) Gaussian processes with built-in dimensionality reduction: applications to high-dimensional uncertainty propagation. J Comput Phys 321:191–223
van Rijn JN, Abdulrahman SM, Brazdil P, Vanschoren J (2015) Fast algorithm selection using learning curves. International symposium on intelligent data analysis (pp 298-309). https://doi.org/10.1007/978-3-319-24465-5_26
Vanschoren J (2019) Meta-learning. Automated machine learning: methods, systems, challenges (pp 35-61). Springer, Cham
Victoria AH, Maragatham G (2021) Automatic tuning of hyper-parameters using bayesian optimization. Evolving Systems 217–223. https://doi.org/10.1007/s12530-020-09345-2
Wang D, Tan D, Liu L (2018) Particle swarm optimization algorithm: an overview. Soft Comput 22(2):387–408
Wang B, Sun Y, Xue B, Zhang M (2019) Evolving deep neural networks by multi-objective particle swarm optimization for image classification. Proceedings of the genetic and evolutionary computation conference (pp 490-498). https://doi.org/10.1145/3321707.3321735
Wang B, Xue B, Zhang M (2020) Particle swarm optimization for evolving deep convolutional neural networks for image classification: Single-and multi-objective approaches. Deep neural evolution (pp 155-184). Springer. https://doi.org/10.1007/978-981-15-3685-4_6
Wang F, Zhang H, Zhou A (2021) A particle swarm optimization algorithm for mixed-variable optimization problems. Swarm Evol Comput 60:100808
Wawrzyński P (2017) Asd+ m: automatic parameter tuning in stochastic optimization and on-line learning. Neural Netw 96:1–10. https://doi.org/10.1016/j.neunet.2017.07.007
Wistuba M, Schilling N, Schmidt-Thieme L (2018) Scalable gaussian process-based transfer surrogates for hyperparameter optimization. Mach Learn 107(1):43–78. https://doi.org/10.1007/s10994-017-5684-y
Wu J, Chen S, Liu X (2020) Efficient hyperparameter optimization through model-based reinforcement learning. Neurocomputing 409:381–393
Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316. https://doi.org/10.1016/j.neucom.2020.07.061
Zames G, Ajlouni N, Ajlouni N, Ajlouni N, Holland J, Hills W, Gold-berg D (1981) Genetic algorithms in search, optimization and machine learning. Inf Technol J 3(1):301–302
Zhang C, Lim P, Qin AK, Tan KC (2016) Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Trans Neural Netw Learn Syst 28(10):2306–2318. https://doi.org/10.1109/TNNLS.2016.2582798
Zhang M, Ni Q, Zhao S, Wang Y, Shen C (2020) A combined prediction method for short-term wind speed using variational mode decomposition based on parameter optimization. 2020 ieee international conference on systems, man, and cybernetics (smc) (pp 2607-2614)
Zhang Q, Li H (2007) Moea/d: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731. https://doi.org/10.1109/TEVC.2007.892759
Zhao S-Z, Suganthan PN, Zhang Q (2012) Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes. IEEE Trans Evol Comput 16(3):442–446. https://doi.org/10.1109/TEVC.2011.2166159
Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195. https://doi.org/10.1162/106365600568202
Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271. https://doi.org/10.1109/4235.797969
Zitzler E, Laumanns M, Thiele L (2001) Spea 2: Improving the strength pareto evolutionary algorithm. TIK-report 103. https://doi.org/10.3929/ethz-a-004284029

Alejandro Morales-Hernández is a PhD student at Hasselt University (Belgium). Graduated in Computer Science, his main line of research has been in Machine Learning field, specifically diversity measures for ensemble of classifiers. Recently he has focused on algorithms for multi-objective hyperparameter optimization of ML methods, considering the performance uncertainty and using metamodel-based algorithms.

Inneke Van Nieuwenhuyse is a Full Professor at Hasselt University (Belgium), and (partly) at KU Leuven (Belgium). Her research interests focus on (multi-objective) optimization of stochastic systems, especially in settings that are expensive to evaluate. To that end, she combines knowledge of operations research techniques with machine learning approaches.

Sebastian Rojas Gonzalez is a post-doctoral research fellow at the Surrogate Modeling Lab (Ghent University) and at the Department of Quantitative Methods (Hasselt University), both in Belgium. Since 2016 his research work has centered on developing stochastic optimization algorithms assisted by machine learning techniques for multi-criteria decision making, with a focus on noise handling strategies using Bayesian learning methods.