Multi-Objective Population Based Training

Arkadiy Dushatskiy [1] **Alexander Chebykin** [1] **Tanja Alderliesten** [2] **Peter A.N. Bosman** [1 3]

Abstract

Population Based Training (PBT) is an efficient hyperparameter optimization algorithm. PBT is a single-objective algorithm, but many real-world hyperparameter optimization problems involve two or more conflicting objectives. In this work, we therefore introduce a multi-objective version of PBT, *MO-PBT*. Our experiments on diverse multi-objective hyperparameter optimization problems (Precision/Recall, Accuracy/Fairness, Accuracy/Adversarial Robustness) show that MO-PBT outperforms random search, single-objective PBT, and the state-of-the-art multi-objective hyperparameter optimization algorithm MO-ASHA.

1. Introduction

The computational complexity of machine learning tasks has drastically increased in recent years. This has been caused by larger models (especially, deep neural networks (Dosovitskiy et al., 2020; Kaplan et al., 2020)) and larger available datasets (Thomee et al., 2016; Byeon et al., 2022). At the same time, the problem of tuning model hyperparameters remains crucial for achieving maximal performance (Kadra et al., 2021; Zhang et al., 2021; Liu et al., 2022). Thus, there is a growing demand for efficient algorithms to do hyperparameter tuning. Moreover, in real-world problems, there might be more than one objective that a user is interested in. An example of such a scenario which recently received a lot of attention from the machine learning community is finding a trade-off between the predictive accuracy of a classifier and its fairness (Schmucker et al., 2020; Chuang & Mroueh, 2021). When different objectives are conflicting and the target trade-off is not known a priori, usually no single best model (or hyperparameter setting) exists. Thus, many models with different trade-offs between the objectives should be presented to the user. Finding hyperparameters that result in models with the best trade-offs is a multi-objective optimization problem.

One of the most efficient approaches to single-objective Hyperparameter Optimization (HPO) is Population Based Training (PBT) (Jaderberg et al., 2017). PBT has two features which ensure its efficiency. Firstly, it is a highly parallelizable, asynchronous algorithm, which means that the available hardware can be effectively utilized. Secondly, in contrast to standard optimization techniques which usually train models from scratch in order to estimate the performance of a particular hyperparameter setting, PBT optimizes hyperparameters during model training. In this work, we propose to expand Population Based Training (Jaderberg et al., 2017) to Multi-Objective HPO (MO-HPO). The population of models used in PBT should be especially well suited for solving Multi-Objective (MO) problems, as maintaining a population is naturally helpful for finding a good trade-off front of solutions, which is known from the Evolutionary Algorithms (EAs) literature (Deb, 2001; Morales-Hernández et al., 2022). EAs such as NSGA-II (Deb et al., 2002) have been used for efficiently solving MO optimization problems, including Neural Architecture Search (Lu et al., 2019).

The main contributions of our work are the following:

1. We expand Population Based Training to MO-HPO scenarios. The overview of the proposed Multi-Objective PBT (*MO-PBT*) algorithm is demonstrated in Figure 1.
2. We demonstrate that using single-objective PBT for MO-HPO by transforming it into a single-objective problem (via a scalarization technique or simply optimizing one of the objectives) is an inferior approach to MO-PBT which uses an MO technique of domination-based selection (Deb et al., 2002).
3. On a set of diverse MO-HPO problems, we demonstrate that MO-PBT outperforms the state-of-the-art efficient, parallelizable hyperparameter optimization algorithm, Multi-Objective Asynchronous Successive Halving (MO-ASHA) (Schmucker et al., 2021).

[1]Centrum Wiskunde & Informatica, Amsterdam, the Netherlands [2]Leiden University Medical Center, Leiden, the Netherlands [3]Delft University of Technology, Delft, the Netherlands. Correspondence to: Arkadiy Dushatskiy <arkadiy.dushatskiy@cwi.nl>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

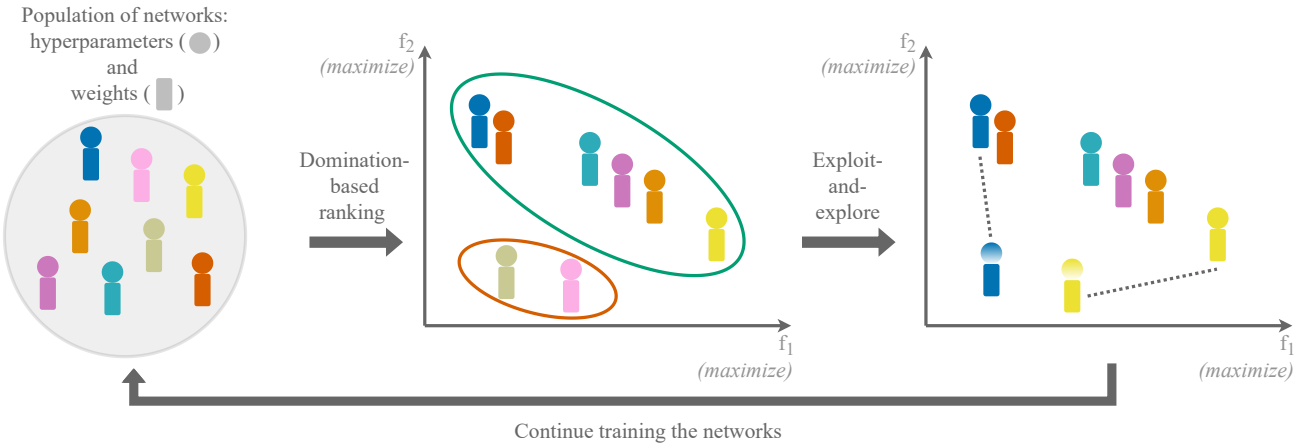


Figure 1. The scheme of the proposed MO-PBT applied to a bi-objective maximization task. After the networks (weights and hyperparameters) in a population have been trained for several epochs, they are ranked using a *domination-based* procedure. Here, each solution inside the smaller, dark-orange oval is dominated by at least one solution inside the larger, green oval (and therefore is considered to be worse). The inferior solutions are replaced with copies of the superior ones (copying is depicted with dotted lines), with hyperparameters perturbed (depicted by adjusted colors). Then the networks are trained for several more epochs, and the loop continues.

Experiments are performed on three different types of MO-HPO problems: precision/recall of a model, the predictive performance of a classifier/its fairness, and accuracy/adversarial robustness.

2. Related work

2.1. Multi-objective hyperparameter optimization applications

There are different scenarios in which a user might be interested in having the option to choose among models with different trade-offs between two (or even more) objectives. The classical example in machine learning is choosing a trade-off between precision and recall of a classifier. The importance of each metric might change depending on the application requirements. Another example is choosing a trade-off between the predictive quality of a classifier and its fairness. It was shown in (Chuang & Mroueh, 2021) that these objectives are conflicting. In (Zhang et al., 2019), it was demonstrated that the classifier accuracy and its robustness to adversarial attacks are conflicting, and therefore, finding a trade-off between them is another interesting MO-HPO application.

2.2. Efficient multi-objective optimization algorithms

A popular class of algorithms for MO-HPO is the Bayesian Optimization (BO) algorithms. Some of them work by reducing an MO optimization problem into a single-objective one by using scalarization techniques (Knowles, 2006; Zhang & Golovin, 2020; Paria et al., 2020; Zhang et al., 2009). An alternative approach to MO optimization with BO is based on expected hypervolume improvement calculation (Emmerich et al., 2011). While BO algorithms are sequential by nature, recently they were extended to the batch-wise calculation of an objective function for solving MO problems (Daulton et al., 2020; 2021). However, the considered batch sizes were moderate (up to 32 solutions), so parallelization capabilities remain limited.

One of the drawbacks of typical BO algorithms is the equal allocation of resources to all evaluated solutions. In contrast to this, it was proposed to greedily stop underperforming model evaluations to save computational resources in Successive Halving (Jamieson & Talwalkar, 2016) and its extension Hyperband (that proposes a more complex resource allocation scheme) (Li et al., 2017). Then, an asynchronous version of Hyperband called Asynchronous Successive Halving Algorithm (ASHA) was proposed, which was shown to achieve a substantial wall-clock time speed-up (Li et al., 2020). Hyperband was extended to MO problems by using random scalarizations in (Schmucker et al., 2020; Guerrero-Viu et al., 2021). Finally, ASHA was extended to MO problems in (Schmucker et al., 2021). Different approaches to adapting ASHA to MO optimization were compared in (Schmucker et al., 2021) and it was concluded that the techniques utilizing the geometry of the Pareto front, in other words, domination-based selection such as in NSGA-II (Deb et al., 2002), outperform scalarization-based techniques.

It was proposed to integrate BO algorithms into Hyperband: (Falkner et al., 2018) replaces random sampling of new candidate solutions by using a Bayesian sampler (TPE (Bergstra et al., 2011)) for more efficient search space exploration. MO-BOHB extends this idea to a multi-objective TPE sampler (MOTPE) (Ozaki et al., 2020).

2.3. Population Based Training

A general formulation of PBT was proposed in (Jaderberg et al., 2017). It was shown to be an efficient way to jointly optimize hyperparameters and model weights of agents in reinforcement learning tasks, Generative Adversarial Networks, and Transformer networks applied to the machine translation task. Later it was shown that it can be also used to efficiently optimize data augmentation parameters for standard image classification datasets such as CIFAR-10/100 (Ho et al., 2019) and 3D object detection (Cheng et al., 2020). In (Liang et al., 2021) it was proposed to incorporate an exploration component in the evaluation procedure of the solutions and add a crossover operator to recombine hyperparameter vectors. In (Dalibard & Jaderberg, 2021) a more complex training scheme with multiple populations was proposed in order to improve the original PBT on problems where the greedy nature of the algorithm might lead to suboptimal results. Other modifications of PBT aim at improving its efficiency by integrating BO techniques (Parker-Holder et al., 2020; Wan et al., 2022). However, we would like to emphasize that all existing PBT modifications are single-objective and are not well-suited to solve multi-objective problems.

In this work, we follow the original design of PBT, which is simpler than the later proposed alternatives and was shown to work well on a diverse set of problems (Jaderberg et al., 2017; Ho et al., 2019; Cheng et al., 2020). However, our approach is general and can potentially be used with any PBT modification.

3. Preliminaries

3.1. Multi-objective optimization

MO optimization problems are characterized by the presence of multiple conflicting objectives. Thus, solving the optimization problem entails finding the best possible trade-offs between the objectives. An MO optimization problem (without loss of generality, we consider maximization) with K objectives can be formulated as follows:

$$\max_{x \in X} f(x) = \max_{x \in X} (f_1(x), f_2(x), \ldots, f_K(x)),$$

where $X \subseteq \mathcal{S}$ is a search space of solutions considered feasible ($\mathcal{S}$ is a search space of all solutions). It is said that a solution x' dominates a solution x ($x' \succ x$) if $\forall i \; f_i(x') \geq f_i(x)$ and $\exists i \; s.t. \; f_i(x') > f_i(x)$.

The Pareto set $\mathcal{P}_s$ of f is a set of all non-dominated solutions, i.e. $\mathcal{P}_s = \{x \in X | \nexists \, x' : x' \succ x\}$ while the Pareto front $\mathcal{P}_f$ is a set of objective values of solutions in $\mathcal{P}_s$: $\mathcal{P}_f = \{(f_1(x), f_2(x), \ldots, f_K(x)) | x \in \mathcal{P}_s\}$. While the Pareto front is often not known, the considered tangible goal of MO optimization algorithms is to obtain a good approximation of it. A popular measure of approximation quality is the dominated hypervolume (Zitzler, 1999). The hypervolume of a finite set of solutions S is calculated as follows: $HV_r(S) = \lambda_K(z \in \mathbb{R}^K : \exists y \in S, r \prec z \prec f(y))$, where $r \in \mathbb{R}^K$ is a chosen reference point and λ_K is a Lebesgue measure. Intuitively, the hypervolume represents the volume (the area in the bi-objective case) between the reference point and the non-dominated trade-off front of solutions (see Appendix F, Figure 17 for visualization).

3.2. Scalarization techniques

Scalarization is a commonly used technique for MO optimization, which transforms a multi-objective problem into a single-objective one: $\max_{x \in X} V(f(x), w)$, where V is a scalarization function and w is a scalarizing weight vector. Following MO optimization literature (Karl et al., 2022), we use ParEGO scalarization function (Knowles, 2006) (also called augmented Chebyshev scalarization (Steuer & Choo, 1983)): $V_{ParEGO} = \rho V_{WS} + V_{Chebyshev}$, where V_{WS} is Weighted Sum scalarization: $V_{WS}(f(x), w)) = \sum_i w_i f_i(x)$ and $V_{Chebyshev}$ is the Chebyshev scalarization: $V_{Chebyshev}(f(x), w)) = \min_i(w_i f_i(x))$, ρ is set to 0.05 in the original ParEGO implementation. Following MO-ASHA (Schmucker et al., 2021), we also use Golovin scalarization (Zhang & Golovin, 2020): $V_{Golovin}(f(x), w)) = \min_i(\max(0, f_i(x)/w_i))^K$ (K is the number of objectives).

4. Multi-Objective Population Based Training

We start with a short summary of PBT and then describe our extension of it to the MO setting.

The goal of PBT is to optimize an objective function f. PBT has a population of N solutions $\mathcal{P} = \{p_i\}_{i=1}^N$, where each individual p_i comprises a tuple of model weights and hyperparameters: $(\theta_i, \mathcal{H}_i)$. The main working principle of PBT is to optimize weights and hyperparameters in an interleaved fashion, which is achieved via two key operators: *exploit* and *explore*. The *exploit* operator replaces a bad solution with a copy of a good one (both weights and hyperparameters are copied). The solution quality is determined by a ranking procedure. The *explore* operator creates a new solution by, e.g., perturbing the hyperparameters of the existing one. Between exploit-and-explore steps, the weights of the models are trained as usual, e.g., using gradient descent.

How solutions are ranked needs to be changed when going from single- to multi-objective optimization. In the single-objective scenario, the population members can be ranked according to the optimization objective value, but with multiple objectives, ranking becomes less trivial.

The first approach we consider is using a scalarization technique, i.e., mapping an objective vector into a scalar. It is then used for ranking solutions, just as in the single-objective case. Secondly, we consider domination-based

ranking, as used for example in NSGA-II (Deb et al., 2002). The main component of such an approach is the *non-dominated sort* of solutions. The idea of the non-dominated sort is to partition a population of solutions $\mathcal{P}$ into non-dominated fronts of solutions, i.e., $\mathcal{P} = F^1 \cup F^2, \ldots, \cup F^R; F^i \cap F^j = \emptyset \; \forall i, j$ such that:

1. All solutions in each front are non-dominated by each other: $\forall k : \forall v_1, v_2 \in F^k \; v_1 \not\succ v_2$ and $v_2 \not\succ v_1$

2. In the kth front ($k > 1$), all solutions are dominated by a solution from a front with a smaller index: $\forall k, 2 \leq k \leq R : \forall v_1 \in F^k \; \exists v_2 \in F^m, m < k : v_2 \succ v_1$

In the sorting procedure, all solutions from F^1 are ranked higher than the solutions from F^2, the ones from F^2 are ranked higher than the solutions from F^3, etc. Within each front, the solutions are ranked according to an additional ranking criterion. In the original NSGA-II algorithm, the *crowding distance* criterion was used. However, in (Schmucker et al., 2021) it was shown that the greedy scattered subset selection (Bosman & Thierens, 2003) (called $\epsilon-$network in (Schmucker et al., 2021; Salinas et al., 2021)) ranking performs better when integrated into the MO-ASHA algorithm (compared to MO-ASHA with the crowding distance). We also experimentally found that MO-PBT with the greedy scattered subset selection performs slightly better than MO-PBT with the crowding distance, as shown in Appendix A.3.

The main idea behind the greedy scattered subset selection is to rank higher the solutions that are further away from the others. Specifically, the next solution is iteratively chosen in a greedy way such that it has the largest Euclidean distance (in the *objective space*) to the closest already ranked solution. The visualization of this ranking procedure is shown in Figure 2, its pseudocode is listed in Appendix G, Algorithm 1.

The *exploit* and *explore* operators of MO-PBT are described in Section 6.1.

5. Multi-objective hyperparameter optimization tasks

5.1. *Precision/Recall* in classification

Balancing between precision and recall of a model is a classical trade-off problem in machine learning (Karl et al., 2022; Lévesque et al., 2012). In this work, we use modern FT-Transformer (Feature Tokenizer Transformer) neural networks (Gorishniy et al., 2021) and three diverse binary classification datasets: Adult (Dua & Graff, 2017), Higgs (Baldi et al., 2014), and Click prediction (Vanschoren et al., 2013). We optimize the regularization parameters: weight decay and dropout, and additionally the class weights in the cross-entropy-loss function (which is a natural way to balance between class-wise performances, and therefore precision and recall). The training procedure for FT-Transformer is adopted from (Gorishniy et al., 2021) (but without early stopping). The early stopping is not included because it is not well-suited for a standard PBT setup (also used here), where a predetermined number of exploit-and-explore steps (and therefore training epochs) is performed.

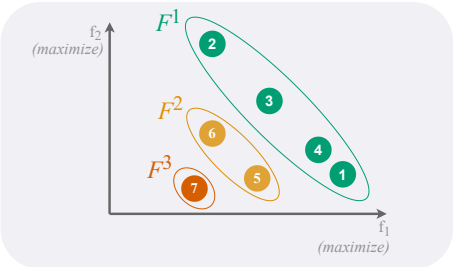


Figure 2. The ranking procedure of solutions (shown with circles) in MO-PBT. Numbers inside circles show the assigned rank (smaller is better). First, solutions are sorted using non-dominated sort. Here, it partitions the solutions into the non-dominated front F^1 (green), the second front F^2 (light orange), and the third front F^3 (dark orange). The solutions in the first front are considered first. The solution with the largest f_1 value is ranked first. Then other solutions from F^1 are ranked one-by-one such that the solution which is the furthest away from the already ranked ones is picked next. This distance-based ranking is continued in the second (third, etc.) fronts.

5.2. Model *Accuracy/Fairness*

The fairness of a model is understood as its ability to predict the target attribute, e.g., income, without a bias on the sensitive attribute, e.g., gender or race. In our experiments, we consider the standard setup of model fairness in binary classification, where labels $Y \in \{0, 1\}$, sensitive attributes $A \in \{0, 1\}$, and model predictions are $\hat{Y}$. Different fairness metrics have been proposed (Garg et al., 2020). Two of the most popular ones are Statistical Parity (SP) and Equalized Odds (EO). SP requires the independence of predictions $\hat{Y}$ on the sensitive attribute A: $\mathrm{P}(\hat{Y}|A = 0) = P(\hat{Y}|A = 1)$. EO requires conditional independence of $\hat{Y}$ and A with respect to Y: $P(\hat{Y}|A = 1, Y = y) = P(\hat{Y}|A = 0, Y = y)$ for $y \in \{0, 1\}$.

Following (Madras et al., 2018; Schmucker et al., 2021; Chuang & Mroueh, 2021), we optimize the relaxed versions of SP and EO called Difference in Statistical Parity (DSP) and Difference in Equalized Odds (DEO).

$$DSP(f) = |E_{x \sim P_0} f(x) - E_{x \sim P_1} f(x)|$$

$$DEO(f) = \sum_{y \in \{0,1\}} |E_{x \sim P_0^y} f(x) - E_{x \sim P_1^y} f(x)|,$$

where $P_a = P(\cdot|A = a)$ and $P_a^y = P(\cdot|A = a, Y = y)$.

Following (Chuang & Mroueh, 2021), the loss during training can be composed of standard Cross-Entropy (CE) and weighted DSP (gap regularization):

$$\mathcal{L}_{fairness}(f(x), y) = CE(f(x), y) + \lambda DSP(f),$$

where x is a training sample, y is the target, and λ is a trade-off parameter.

We consider the Adult dataset with gender as the sensitive attribute and income as the target. We use the same setup as in 5.1 (FT-Transformer neural networks, optimizing regularization), but instead of a class weighting parameter in the cross-entropy loss, we use the $\mathcal{L}_{fairness}$ loss and optimize the λ parameter. Also, we use the CelebA dataset (Liu et al., 2015) with gender as the sensitive attribute and Attractiveness as the binary classification target. The training setup is the same as in Section 5.3, but with the $\mathcal{L}_{fairness}$ loss.

5.3. *Accuracy/Adversarial robustness*

It has been shown that standard model accuracy and its adversarial robustness (accuracy on samples generated by an adversarial attack) are conflicting objectives (Zhang et al., 2019). In this task, we use the TRADES loss (Zhang et al., 2019):

$$\mathcal{L}_{TRADES}(f(x), y) = CE(f(x), y) + \max_{x' \in \mathcal{B}(x,\epsilon)} \lambda CE(f(x), f(x')),$$

where x is a training sample, x' is a generated adversarial sample in the $\epsilon-$neighborhood of x, and y is the target. The parameter λ affects the trade-off between accuracy and adversarial robustness. We use the same adversarial attack and TRADES loss parameters as in (Zhang et al., 2019). We search for data augmentation parameters: parameters of the RandAugment augmentation strategy (Cubuk et al., 2020) and Cutout (DeVries & Taylor, 2017) (probability and size). Experiments are performed for CIFAR-10/100 datasets using the WideResNet-28-2 (Zagoruyko & Komodakis, 2016) and the training setup from (Zhang et al., 2019).

5.4. Search spaces

In this work, we perform search in discretized search spaces. Such an approach was successfully used, for instance, for augmentations search (Ho et al., 2019; Cubuk et al., 2020). For all described optimization tasks, search spaces of hyperparameters are specified in Appendix H.

6. Experimental setup

6.1. PBT operators

Here we describe the operators of MO-PBT following the notation from (Jaderberg et al., 2017).

Exploit We use the simple truncation selection operator used in the original PBT (Jaderberg et al., 2017) algorithm. After the population is sorted according to some criterion (non-dominated sort followed by the greedy scattered subset selection in the case of MO-PBT), each of the bottom $\tau\%$ of solutions in the population is randomly replaced by a solution from the top $\tau\%$ (we use the default value of τ, 25). The pseudocode of the used *exploit* operator is listed in Appendix G, Algorithm 2.

Explore We use the *explore* operator previously used in Population Based Augmentations (Ho et al., 2019). It assumes that the encoding of hyperparameter values in a search space is ordinal. The key idea of the operator is locality: the new value of a hyperparameter is chosen from the vicinity of the current value. The pseudocode of the used *explore* operator is listed in Appendix G, Algorithm 3.

Ready In all considered tasks, we perform the exploit-and-explore procedure every 2 epochs of training.

We use a population of size 32 in our main experiments and in Section 7.4 study how the performance scales with increasing population size. Note that we do not specifically tune *exploit* and *explore* operators of MO-PBT, but in Appendix A we analyze how their design impacts the performance and conclude that the considered design options perform similarly.

6.2. Hypervolume as the performance metric

We use the hypervolume, a commonly used metric in MO optimization (Riquelme et al., 2015) (see Section 3.1). We calculate the reference point $r = (r_1, \ldots, r_K)$ with the following approach, which is used, for instance, in (Knowles, 2006; Ishibuchi et al., 2011). First, all non-dominated fronts are collected from all evaluation points of all algorithms and all performed runs and stored in a set $\mathcal{F}$. Then, the reference point r is calculated as $r_i = \min_{x \in \mathcal{F}} f(x_i) - \rho(\max_{x \in \mathcal{F}} f(x_i) - \min_{x \in \mathcal{F}} f(x_i))$, for $i = 1, \ldots, K$, where ρ is typically set to a small value, here we use $\rho = 0.1$. This strategy selects the reference point that is guaranteed to be worse than all points on all fronts. This ensures that all non-dominated points are considered in the hypervolume calculation and are not discarded. Furthermore, the reference point is shifted with respect to the range of values of each objective in order to prevent its positioning too far away from the fronts.

In the experimental evaluation, we use the following common metric (used, e.g., in (Schmucker et al., 2021; Daulton et al., 2021). First, we obtain an approximation of the Pareto front by collecting all evaluated solutions from all runs of all algorithms and selecting the non-dominated subset $\mathcal{P}^*$ of them. The approximation of the optimal hypervolume is then calculated as $HV^* = Hypervolume(\mathcal{P}^*)$. The

reported performance metric of an algorithm run r at timestamp t is the logarithmic difference of the hypervolume of a non-dominated set of solutions obtained by this timestamp and the ideal hypervolume: $log_{10}(HV^* - HV_r^t)$. Finally, this metric is averaged over multiple runs.

Datasets are split into train/validation/test subsets before experiments. In our main results, we report the above-described hypervolume metric on the *validation* subset to evaluate the search performance of the algorithms. Additionally, we provide results on the test subsets in Appendix C.

6.3. Baselines

6.3.1. Random search

First, we consider a trivial search baseline — random search: for each hyperparameter, a random value is sampled at the beginning of model training.

6.3.2. Single-objective PBT

We use modifications of PBT that convert an MO problem into a SO one. First, we use one of the objectives as the fitness function of PBT. Comparing against this baseline can show that the considered MO problems are challenging, and optimizing just one objective is inferior to using MO techniques. Secondly, we implement different scalarization functions in PBT. The first technique we use is random scalarization as, for instance, in the ParEGO algorithm (Knowles, 2006): at each invocation of the evaluation procedure, the scalarization vector is sampled randomly. Here we use the ParEGO scalarization function (as defined in Section 3.2) as it was originally proposed to use for random scalarizations in (Knowles, 2006). Secondly, we use the maximum scalarization technique proposed in (Schmucker et al., 2020): the objective value is calculated as $\max_{w \in W, ||w||=1} V(f(x), w)$, where W is a set of randomly sampled unit vectors and V is a scalarization function. Following (Schmucker et al., 2021), we use $|W| = 100$ and the Golovin scalarization, which was demonstrated to outperform other scalarization functions.

6.3.3. MO-ASHA variants

We consider MO-ASHA with greedy scattered subset selection ranking (ϵ-network), which was shown to perform better than alternative MO-ASHA variants in (Schmucker et al., 2021). Secondly, to compare MO-PBT against a strong BO baseline that is well parallelizable we adapt the MO-BOHB (Guerrero-Viu et al., 2021) approach to MO-ASHA. We refer to this MO-ASHA modification as BO-MO-ASHA.

6.4. Evaluation setup

The main design principle of our evaluation of algorithms is to compare the achieved performance with respect to elapsed wall-clock time instead of the performed number of training epochs. We choose this approach because in practice we are more interested in the achieved performance by a specific time point rather than a specific epoch. We do not set a time limit for all PBT variants and random search but rather allow them to fully finish the training cycle of all solutions in the population. For MO-ASHA, we allocate the time budget equal to the run time of the slowest PBT run. We ran each algorithm 10 times on tabular datasets (Adult, Higgs, Click prediction), and 5 times on image ones (CIFAR-10/100, CelebA). When plotting performance over time, we plot mean performance, with the area between the worst and the best runs shaded.

Further experimental setup details are provided in Appendix, B. The code is available at `https://github.com/ArkadiyD/MO-PBT`.

7. Results

7.1. Overall performance

Results of hypervolume-based performance evaluation (as described in 6.2) are shown in Figures 3,4,5. On every considered task, MO-PBT outperforms baselines (the standard deviations of the hypervolume are provided in Appendix I, Table 5). Noteworthy, on the three-objective problems MO-PBT is also the best-performing algorithm. The consistently good performance of MO-PBT on the considered diverse tasks empirically demonstrates its generality.

On Accuracy/Fairness tasks, optimizing the fairness objective with SO PBT leads to obtaining mostly inaccurate models, and, therefore, poor hypervolume values. Similarly, on the Accuracy/Robustness task, if only accuracy is optimized, the results achieved for the robustness objective are poor. PBT with scalarization techniques performs, in general, better than single-objective PBT.

Comparing MO-ASHA variants, we cannot conclude that BO-MO-ASHA performs better than MO-ASHA. We note that on CIFAR-10/100 Accuracy/Robustness tasks (Figure 5), MO-ASHA and BO-MO-ASHA perform better than MO-PBT in the beginning of the search as they train networks in a different order compared to MO-PBT: some selected networks are fully trained earlier in time than in MO-PBT, where all the networks are trained simultaneously. However, as soon as the population is trained for more epochs, MO-PBT catches up and at the end of the search substantially outperforms MO-ASHA. We note that this behavior occurs only because the number of parallel workers in our experiments is smaller than the population size.

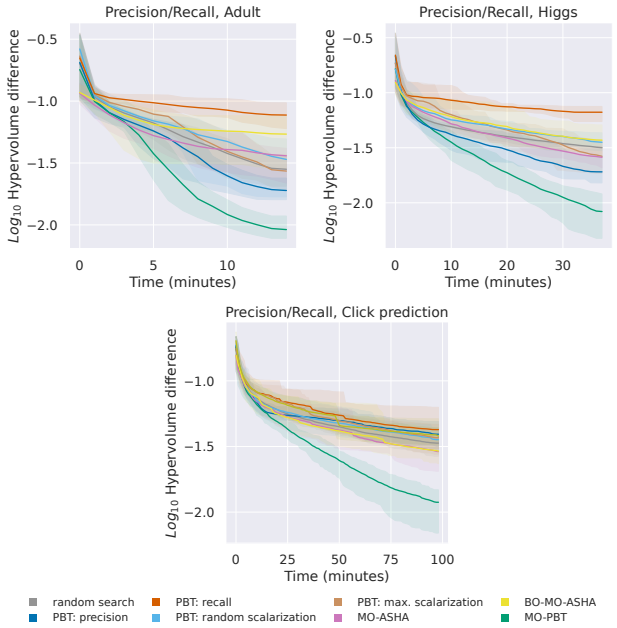


Figure 3. Optimization results on the Precision/Recall task.

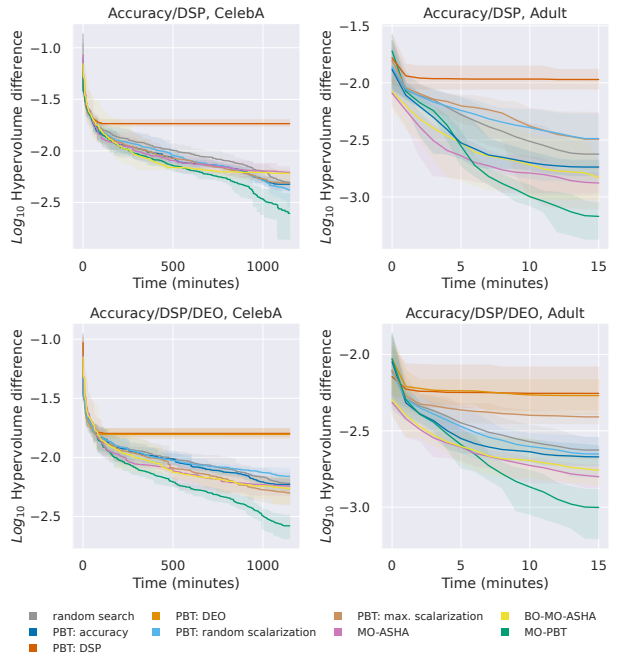


Figure 4. Optimization results on the Accuracy/Fairness task.

7.2. Analysis of the obtained trade-off fronts

The comparison of non-dominated fronts of solutions obtained by different algorithms is shown in Figure 6. Quantitative results of the front diversity evaluation (using the coverage metric introduced in (Scriven et al., 2009) and described in Appendix F) are shown in Appendix I, Table 7.

We observe that on the Precision/Recall and the Accuracy/Robustness tasks, MO-PBT achieves substantially better coverage of the trade-off front compared to other algorithms. On Accuracy/DSP tasks, MO-ASHA on average has slightly better coverage than MO-PBT (though the variance of the results is large and in some runs MO-PBT has better coverage). Nevertheless, it should be noted that the quality of the most points on the trade-off fronts obtained by MO-ASHA is worse in terms of domination (can be seen on Figure 6).

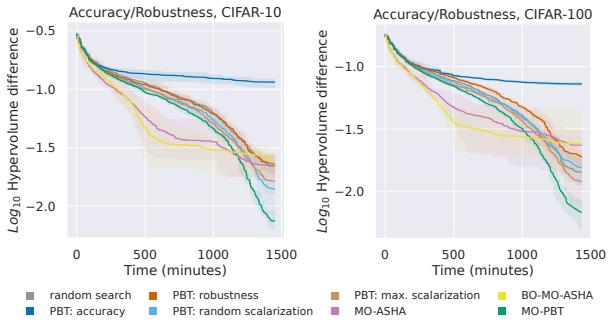


Figure 5. Optimization results on the Accuracy/Robustness task.

These results demonstrate that MO-PBT can not only find solutions closer to the reference front than MO-ASHA (which is reflected in the better hypervolume performance), but also produce more diverse fronts along the entire trade-off curve. For practical usage, this means that more options for trade-offs between objectives are available for the user to choose from.

7.3. Where do different algorithms focus their search?

We analyze how algorithms differ in their search behavior by plotting all solutions collected during the search in the objective space and highlighting areas where more solutions are concentrated. These plots are shown in Figure 7. They show a clear difference between approaches which turn an MO problem into an SO one (scalarization and optimizing one objective) and MO-PBT. MO-PBT obtains solutions scattered more uniformly along the entire trade-off trajectory between two objectives, in contrast to concentrating on one area of it.

7.4. Scalability

7.4.1. Population size

We investigate whether MO-PBT benefits from increasing the population size. The scaling experimental results are shown in Figure 8. We find that for the population sizes we considered (16, 32, 64), performance keeps improving. For reference, we also ran MO-ASHA with correspondingly increased time budgets and found that it scales similarly. We note that the performance gains when the population size is increased (from 16 to 32 and from 32 to 64) are stronger

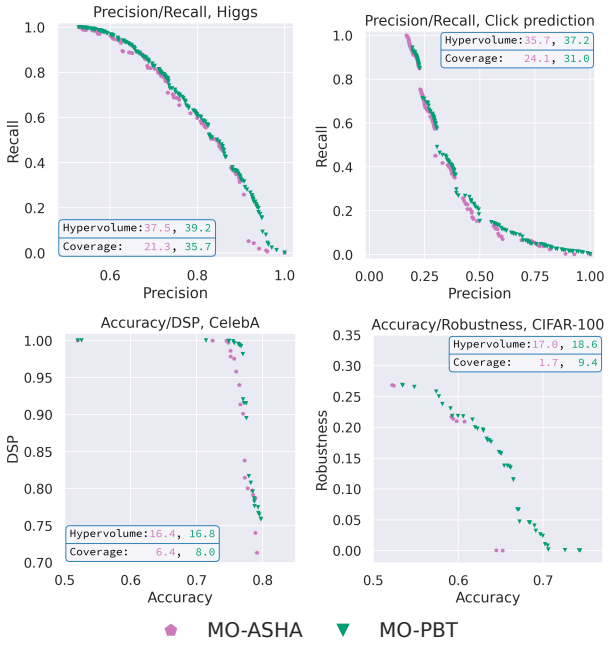


Figure 6. Comparison of the non-dominated fronts obtained by different algorithms. For each algorithm, the run with the median hypervolume value is shown. For each front, the values of the hypervolume and coverage metrics (multiplied by 100) are reported in the corresponding color.

on the task with 3 objectives. This is expected behavior, as the algorithm needs more solutions to scatter along 3D approximation fronts compared to 2D in the bi-objective case.

7.4.2. Search space size

In our main experiments on image datasets, we search for the two parameters (number of augmentations and their magnitude) of the RandAugment augmentation policy, which was shown to be effective (Cubuk et al., 2020). The whole search space has in that case 5 variables. To additionally study whether MO-PBT is capable of performing search efficiently in larger search spaces, we construct a substantially larger search space (comprising 31 variables) by replacing the RandAugment policy with an augmentation policy similar to the one used in (Ho et al., 2019): the magnitude and probability of each augmentation can be adjusted separately; additionally, the number of applied augmentations is searchable too. The results are shown in Figure 9. We observe that in a larger search space, MO-PBT does not lose its efficiency and has even a slightly better performance.

7.5. Further experiments to demonstrate the effectiveness of MO-PBT

In addition to our main experiments, we compare MO-PBT to the algorithms which are not (fully) parallel. In Appendix D we demonstrate that MO-PBT outperforms state-of-the-art BO algorithm for MO optimization: Parallel Noisy Expected Hypervolume Improvement (qNEHVI) (Daulton et al., 2021). Furthermore, we conduct experiments to ensure that MO-PBT is an efficient optimization algorithm even in the scenario when it is executed sequentially. In Appendix E, MO-PBT is shown to outperform common MO baselines NSGA-II (Deb et al., 2002) and ParEGO (Knowles, 2006) in the sequential setup.

8. Discussion and limitations

We have proposed MO-PBT and compared it with various baselines, including a prominent parallelizable algorithm for MO-HPO, MO-ASHA, reaching the conclusion that MO-PBT performs better. We note however that an advantage of MO-ASHA (and similar algorithms such as Hyperband) is its ability to perform not only HPO, but also architecture search, and, furthermore, joint optimization of the architecture and hyperparameters. In PBT and MO-PBT all architectures are assumed to be identical in the population, therefore architecture search cannot be performed (without additional modifications).

We note that quantifying the results of MO algorithms is, in, general, challenging. Many metrics have been proposed (Audet et al., 2021) and each has its own pros and cons. While hypervolume remains, arguably, the most commonly used metric, its downside is the dependence on a user-selected reference point. Thus, while we ensured that the proposed MO-PBT outperformed alternative algorithms in terms of hypervolume, we also visually analyzed the obtained non-dominated fronts of solutions and quantified the results using a coverage metric (Scriven et al., 2009). This analysis also demonstrated good performance of MO-PBT in terms of solutions diversity and density (they are well spread across different areas of the objective space).

In principle, MO-PBT (as well as the original PBT) can operate with any type of search space as long as an *explore* operator is defined (moreover, the search spaces can be defined separately for each hyperparameter). One of the benefits of the discretized search space used in this work is (in contrast to a real-valued search space), its ability to explicitly set some values: e.g., zero value of λ in the $\mathcal{L}_{fairness}$ turns this loss into the standard cross-entropy. Thus, more interpretable hyperparameter search results can be obtained. However, a real-valued search space can, potentially, enable performing a more fine-grained search which in some cases might be more important than the interpretability of results.

For the main experiments of this work, we used MO-PBT with population size of 32. We additionally observed that its performance scales with increasing population size. However, population size remains a hyperparameter of MO-PBT

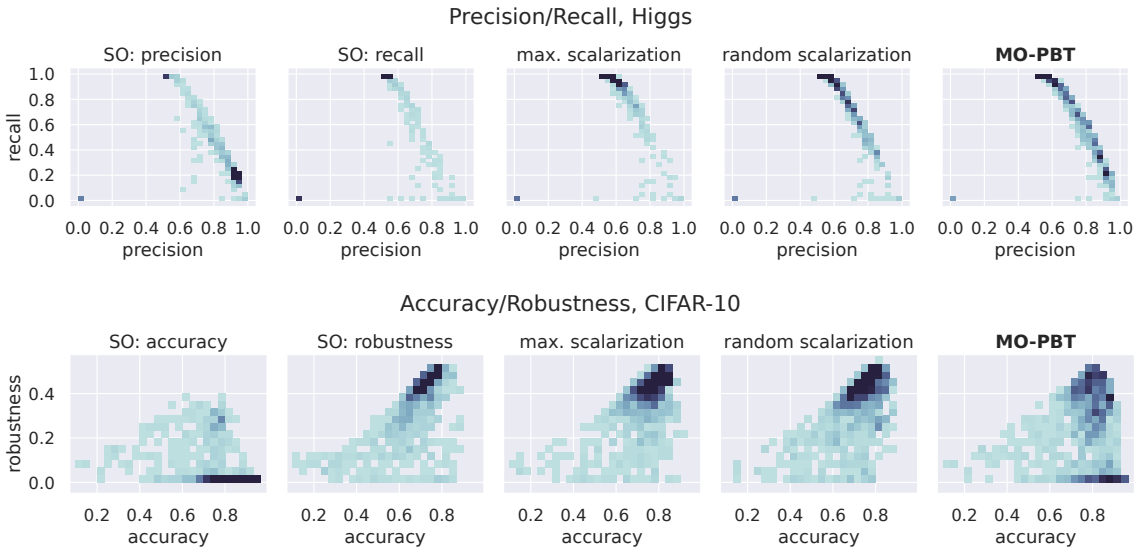


Figure 7. 2D histograms of solutions (in the objective space) collected during one run of each algorithm. Darker color denotes more solutions in the corresponding bin (for visualization purposes, bins with more solutions than the 95th percentile of the bin counts have the darkest color on the plot). For each algorithm, solutions obtained during the run with the median hypervolume value are plotted. SO denotes PBT applied to optimizing one of the objectives.

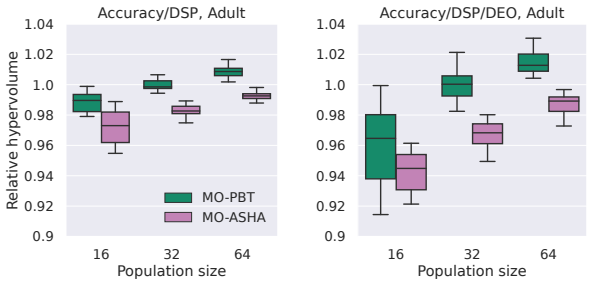


Figure 8. Comparison of MO-PBT with increasing population size and MO-ASHA. The time budget for MO-ASHA was adjusted accordingly. The hypervolume is normalized by the average hypervolume performance of MO-PBT with population 32.

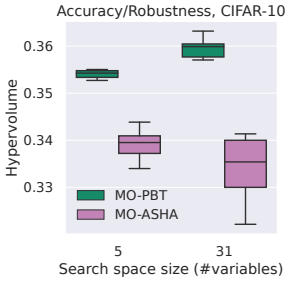


Figure 9. Comparison of MO-PBT and MO-ASHA applied to the search spaces of different sizes.

that needs to be set by the user. Adjusting it automatically (for example, as done in EA literature (Harik et al., 1999)) could be an interesting direction for future work.

9. Conclusion

We introduced a multi-objective version of Population Based Training: MO-PBT. We considered diverse multi-objective hyperparameter optimization tasks and found that a multi-objective approach to ranking solutions, non-dominated sort, outperforms more simple ones such as scalarization techniques. This was demonstrated by not only better hypervolume performance, but also a better tradeoff front coverage by MO-PBT. MO-PBT was shown to outperform MO-ASHA variants (standard and Bayesian optimization based), single-objective PBT, and random search.

Acknowledgements

The work in this paper is supported by: the Dutch Research Council (NWO) through project OCENW.GROOT.2019.015 ”Optimization for and with Machine Learning (OPTIMAL)”; and project DAEDALUS funded via the Open Technology Programme of the NWO, project number 18373; part of the funding is provided by Elekta and ORTEC LogiqCare.

References

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., and Salomon, L. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422, 2021.

Baldi, P., Sadowski, P., and Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011.

Bosman, P. A. and Thierens, D. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):174–188, 2003.

Byeon, M., Park, B., Kim, H., Lee, S., Baek, W., and Kim, S. COYO-700M: Image-text pair dataset, 2022.

Cheng, S., Leng, Z., Cubuk, E. D., Zoph, B., Bai, C., Ngiam, J., Song, Y., Caine, B., Vasudevan, V., Li, C., et al. Improving 3d object detection through progressive population based augmentation. In *European Conference on Computer Vision*, pp. 279–294. Springer, 2020.

Chuang, C.-Y. and Mroueh, Y. Fair Mixup: Fairness via interpolation. *arXiv preprint arXiv:2103.06503*, 2021.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. RandAugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Dalibard, V. and Jaderberg, M. Faster improvement rate population based training. *arXiv preprint arXiv:2109.13800*, 2021.

Daulton, S., Balandat, M., and Bakshy, E. Differentiable expected hypervolume improvement for parallel multiobjective bayesian optimization. *Advances in Neural Information Processing Systems*, 33:9851–9864, 2020.

Daulton, S., Balandat, M., and Bakshy, E. Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems*, 34:2187–2200, 2021.

Deb, K. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, Inc., 2001.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with CutOut. *arXiv preprint arXiv:1708.04552*, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Emmerich, M. T., Deutz, A. H., and Klinkenberg, J. W. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 2147–2154. IEEE, 2011.

Falkner, S., Klein, A., and Hutter, F. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446. PMLR, 2018.

Garg, P., Villasenor, J., and Foggo, V. Fairness metrics: A comparative analysis. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 3662–3666. IEEE, 2020.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34: 18932–18943, 2021.

Guerrero-Viu, J., Hauns, S., Izquierdo, S., Miotto, G., Schrodi, S., Biedenkapp, A., Elsken, T., Deng, D., Lindauer, M., and Hutter, F. Bag of baselines for multiobjective joint neural architecture search and hyperparameter optimization. *arXiv preprint arXiv:2105.01015*, 2021.

Harik, G. R., Lobo, F. G., et al. A parameter-less genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 99, pp. 258–267, 1999.

Ho, D., Liang, E., Chen, X., Stoica, I., and Abbeel, P. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pp. 2731–2741. PMLR, 2019.

Ishibuchi, H., Akedo, N., and Nojima, Y. A many-objective test problem for visually examining diversity maintenance behavior in a decision space. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation*, pp. 649–656, 2011.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning,

I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Jamieson, K. and Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pp. 240–248. PMLR, 2016.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. Well-tuned simple nets excel on tabular datasets. *Advances in Neural Information Processing Systems*, 34:23928–23941, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karl, F., Pielok, T., Moosbauer, J., Pfisterer, F., Coors, S., Binder, M., Schneider, L., Thomas, J., Richter, J., Lang, M., et al. Multi-objective hyperparameter optimization–an overview. *arXiv preprint arXiv:2206.07438*, 2022.

Knowles, J. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

Lévesque, J.-C., Durand, A., Gagné, C., and Sabourin, R. Multi-objective evolutionary optimization for generating ensembles of classifiers in the ROC space. In *Proceedings of the 14th annual conference on Genetic and Evolutionary Computation*, pp. 879–886, 2012.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-Tzur, J., Hardt, M., Recht, B., and Talwalkar, A. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020.

Liang, J., Gonzalez, S., Shahrzad, H., and Miikkulainen, R. Regularized evolutionary population-based training. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 323–331, 2021.

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., and Stoica, I. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., and Banzhaf, W. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 419–427, 2019.

Madras, D., Creager, E., Pitassi, T., and Zemel, R. Learning adversarially fair and transferable representations. In *International Conference on Machine Learning*, pp. 3384–3393. PMLR, 2018.

Morales-Hernández, A., Van Nieuwenhuyse, I., and Rojas Gonzalez, S. A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artificial Intelligence Review*, pp. 1–51, 2022.

Ozaki, Y., Tanigaki, Y., Watanabe, S., and Onishi, M. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 533–541, 2020.

Paria, B., Kandasamy, K., and Póczos, B. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pp. 766–776. PMLR, 2020.

Parker-Holder, J., Nguyen, V., and Roberts, S. J. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in Neural Information Processing Systems*, 33:17200–17211, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Riquelme, N., Von Lücken, C., and Baran, B. Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, pp. 1–11. IEEE, 2015.

Salinas, D., Perrone, V., Cruchant, O., and Archambeau, C. A multi-objective perspective on jointly tuning hardware and hyperparameters. *arXiv preprint arXiv:2106.05680*, 2021.

Schmucker, R., Donini, M., Perrone, V., and Archambeau, C. Multi-objective multi-fidelity hyperparameter optimization with application to fairness. 2020.

Schmucker, R., Donini, M., Zafar, M. B., Salinas, D., and Archambeau, C. Multi-objective asynchronous successive halving. *arXiv preprint arXiv:2106.12639*, 2021.

Scriven, I., Lewis, A., and Mostaghim, S. Dynamic search initialisation strategies for multi-objective optimisation in peer-to-peer networks. In *2009 IEEE Congress on Evolutionary Computation*, pp. 1515–1522. IEEE, 2009.

Steuer, R. E. and Choo, E.-U. An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical programming*, 26(3):326–344, 1983.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

Wan, X., Lu, C., Parker-Holder, J., Ball, P. J., Nguyen, V., Ru, B., and Osborne, M. Bayesian generational population-based training. In Guyon, I., Lindauer, M., van der Schaar, M., Hutter, F., and Garnett, R. (eds.), *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pp. 14/1–27. PMLR, 25–27 Jul 2022.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, B., Rajan, R., Pineda, L., Lambert, N., Biedenkapp, A., Chua, K., Hutter, F., and Calandra, R. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 4015–4023. PMLR, 2021.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482. PMLR, 2019.

Zhang, Q., Liu, W., Tsang, E., and Virginas, B. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2009.

Zhang, R. and Golovin, D. Random hypervolume scalarizations for provable multi-objective black box optimization. In *International Conference on Machine Learning*, pp. 11096–11105. PMLR, 2020.

Zitzler, E. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Shaker Ithaca, 1999.

A. Ablation studies

A.1. *Explore* operator

We compare the used local *explore* operator from PBA (Ho et al., 2019) with random mutation from classic EAs: for each of P variables, the new value is randomly sampled from the corresponding search domain $\mathcal{H}^{\mathcal{P}}$ with probability $\frac{1}{P}$. As demonstrated in Figure 10, the local mutation performs slightly better.

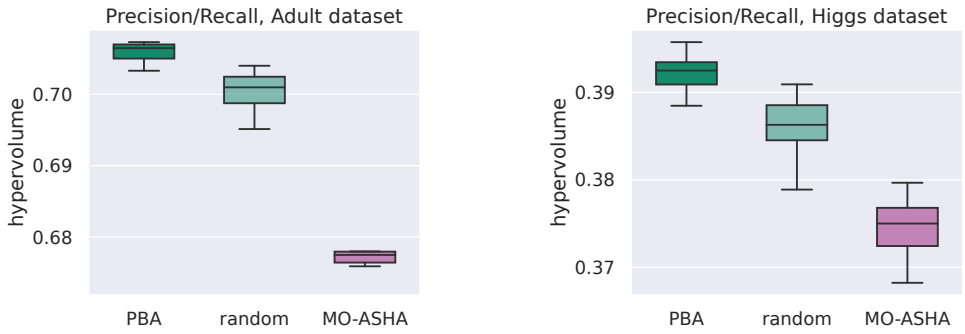


Figure 10. Comparison of MO-PBT with different *explore* operators (mutations) and MO-ASHA. PBA denotes the mutation used in PBA (Ho et al., 2019) (local mutation), random denotes simple and random mutation.

A.2. *Exploit* operator

In (Jaderberg et al., 2017) it was shown that using truncation selection as the *exploit* operator works best out of considered options. Here, we study whether the truncation selection parameter (τ in Algorithm 2, larger value means more solutions are replaced) has a significant impact on performance. As demonstrated in Figure 11, the value of $\tau = 25$ used in (Jaderberg et al., 2017) performs slightly better than values 10 and 50.

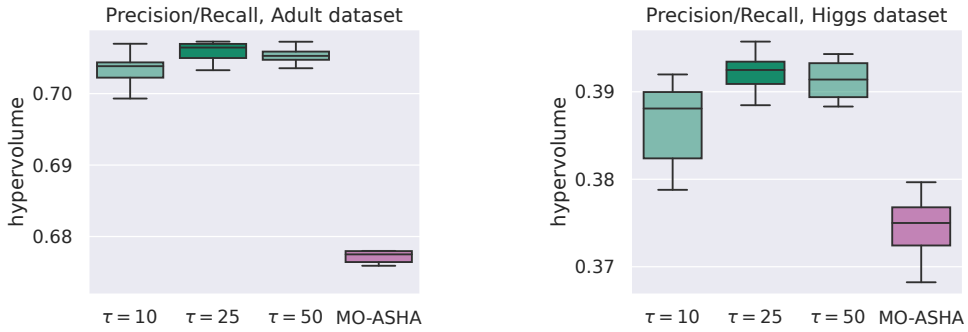


Figure 11. Comparison of MO-PBT with different truncation selection values in the *exploit* operator (parameter τ in Algorithm 2) and MO-ASHA.

A.3. Ranking criterion

We study whether greedy scattered subset selection used for solutions ranking (together with non-dominated sort as described in Section 4) performs better than the crowding distance (the default ranking criterion in NSGA-II).

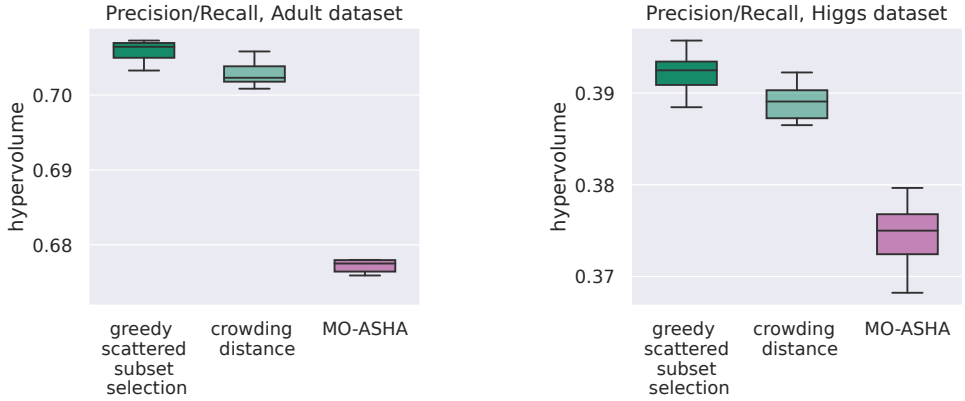


Figure 12. Comparison of MO-PBT with different criteria of ranking solutions and MO-ASHA.

A.4. Ablation studies conclusions

We can conclude that MO-PBT demonstrates robustness to its operators' design: while the default designs of *explore* and *exploit* operators also used in PBT for augmentations policy search (Jaderberg et al., 2017) perform slightly better than considered alternatives, MO-PBT outperforms MO-ASHA for all considered operators. A similar result is observed for the ranking criterion of the solutions: MO-PBT with the greedy scattered subset selection performs better than MO-PBT with the crowding distance, but the difference is not substantial.

B. Implementation and experimental details

We implemented all algorithms using Ray Tune library (Liaw et al., 2018). Network training was performed using PyTorch (Paszke et al., 2019). We used machines with 3 Nvidia A5000 GPUs and trained 4 networks on each GPU simultaneously, i.e., 12 networks could be trained in parallel. The total utilized number of GPU hours to reproduce all of our experiments (8 algorithms and 10 (5) seeds per algorithm on tabular (image) datasets) is 900 GPU hours for the experiments on tabular datasets and 10,000 GPU hours for the experiments on image datasets. One run of MO-PBT took less than 2 wall-clock hours on tabular datasets and less than 25 wall-clock hours on image datasets.

The training procedure for FT-Transformer (in Precision/Recall tasks and Accuracy/Fairness on the Adult dataset) is adapted from (Gorishniy et al., 2021): AdamW (Loshchilov & Hutter, 2017) with learning rate 10^{-5} (no learning rate scheduler is used) but without early stopping. Batch size is set to 512. The training is performed for 100 epochs. On the image datasets, we use standard for WideResNet (used, for instance, in (Cubuk et al., 2020)) cosine learning rate schedule with an initial learning rate 0.1 for SGD with momentum value of 0.9, and batch size 128. The training is performed for 100 epochs.

C. Generalization results

In this section, we inspect how the performance of the algorithms transfers from the validation set to the test set. The solutions on the trade-off front were determined based on the validation metrics. Then, these selected models are evaluated on the test set. Some of them may perform worse than expected, and not be a part of the trade-off front anymore because they are dominated by other solutions on it. The hypervolume of the remaining solutions is computed and visualized in Figure 13 and listed in Table 6.

MO-PBT outperforms the baselines, although on some tasks (Precision/Recall, Accuracy/DSP on the Adult dataset) the difference in final performance becomes smaller.

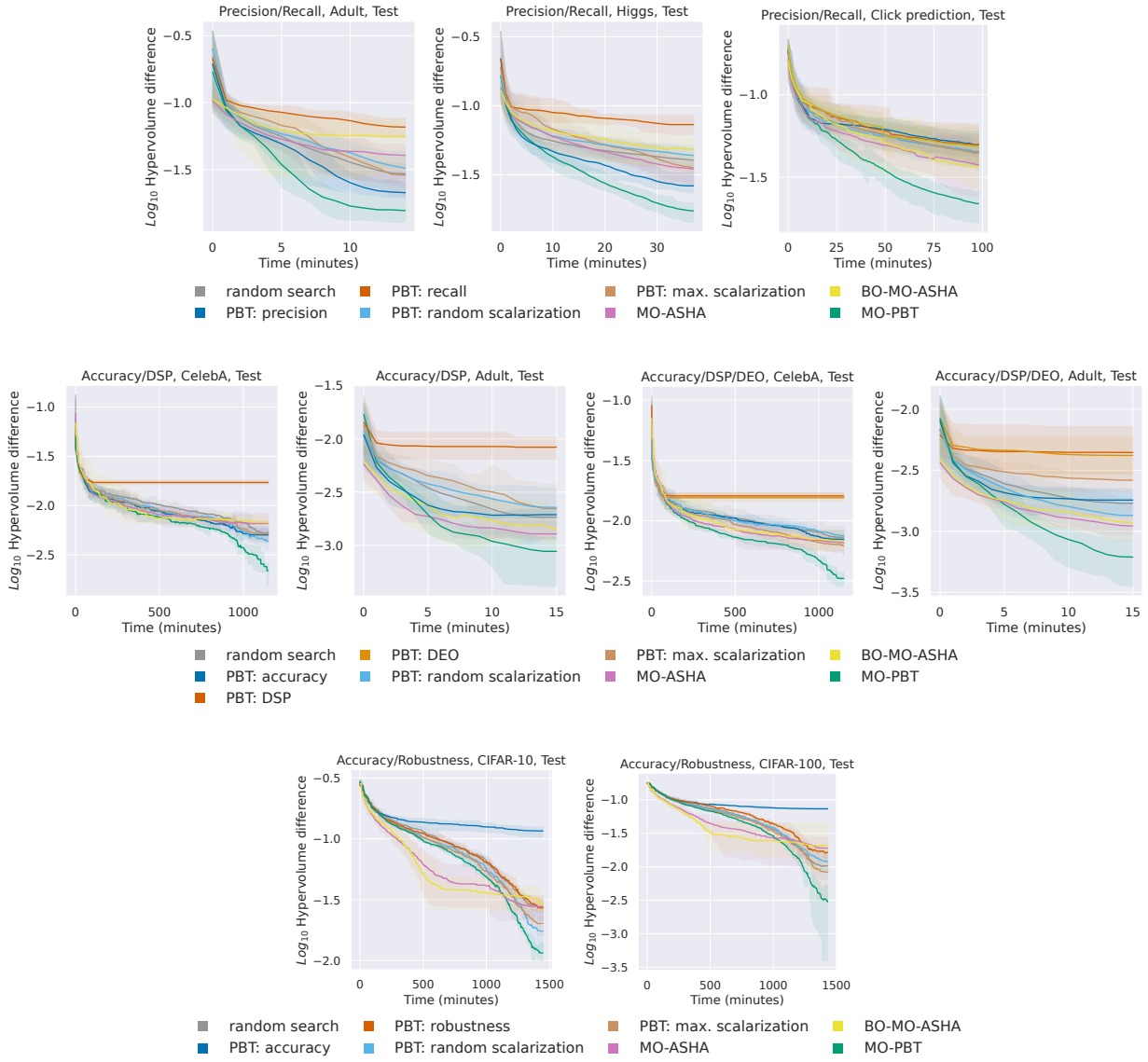


Figure 13. Generalization results (on the *test* data subset) on all tasks: Precision/Recall (top row), Accuracy/Fairness (middle row), Accuracy/Robustness(bottom row).

D. Comparison to parallel BO algorithms

Here we compare MO-PBT to the state-of-the-art BO algorithm qNEHVI (Daulton et al., 2021) which, in contrast to traditional BO algorithms, is capable of evaluating solutions in batches. The used batch sizes for qNEHVI are chosen according to our maximal available parallel capacity: 16 for tabular datasets and 12 for the image ones. These results are shown in Figure 14. They demonstrate that MO-PBT outperforms qNEHVI.

We further note that the parallel capacity of MO-PBT is limited only by the available hardware: the whole population of N networks can be potentially trained in approximately the same amount of time as one network if N parallel workers are available and no bottlenecks appear in the system. This is not the case for, for instance, qNEHVI: first, multiple sequential training iterations (one batch comprises multiple networks) are required to achieve better than random performance; secondly, its performance is expected to deteriorate when the batch size is scaled up to large values (Daulton et al., 2021).

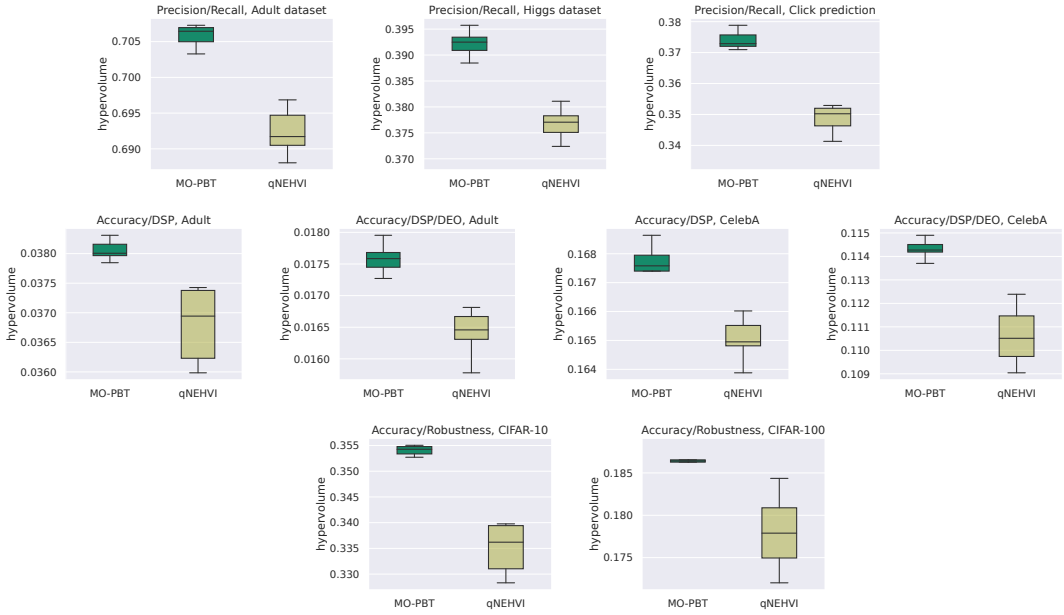


Figure 14. Comparison of MO-PBT against the state-of-the-art BO algorithm, qNEHVI (Daulton et al., 2021). Similar to our main experimental setup described in Section 6.4, time budgets of qNEHVI are equal to the longest run of MO-PBT in the corresponding task.

E. Search effectiveness of MO-PBT

Our main experiments demonstrated the efficiency of MO-PBT for practical applications to MO-HPO tasks. Its highly parallel nature plays an important role in its efficiency. Here, we additionally test the search effectiveness of MO-PBT regardless of its parallelization capabilities. For this purpose, we test it against well-known MO baselines: NSGA-II (Deb et al., 2002) and ParEGO (Knowles, 2006). We allow each algorithm to fully train 32 networks (not taking the required wall-clock time into account) and evaluate the performance of the algorithms based on the hypervolume value of obtained non-dominated fronts of solutions. The results are shown in Figure 15. We can conclude that MO-PBT outperforms the considered alternatives.

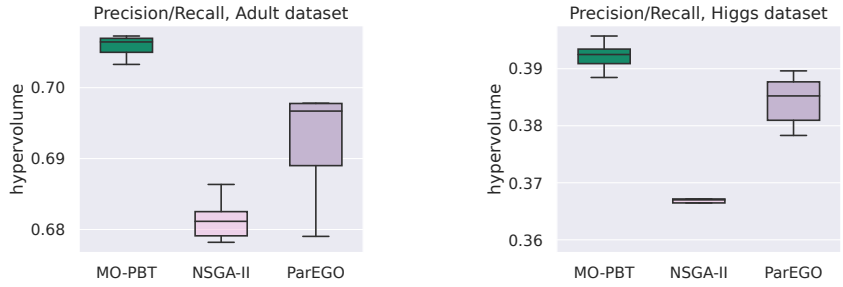


Figure 15. Comparison of MO-PBT against common MO optimization baselines NSGA-II and ParEGO. Here, all algorithms are allowed to fully train 32 networks (the population size in MO-PBT) and the consumed wall-clock time is not taken into account in the evaluation.

F. Visualization of the used performance metrics

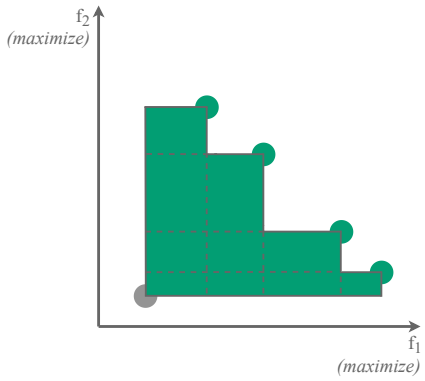


Figure 16. Calculation procedure of the hypervolume metric. Here the green points form the non-dominated front, the gray one is the reference point. The total hypervolume is calculated as the area of the union of rectangles, where each rectangle is formed by a point on the non-dominated front and the reference point.

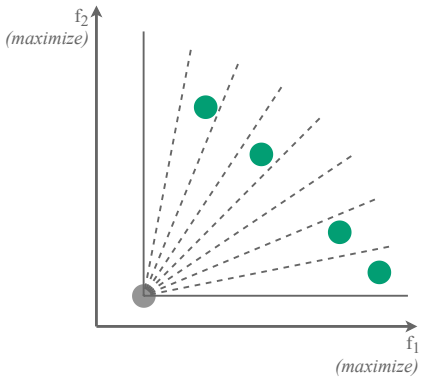


Figure 17. Calculation procedure of the coverage metric (Scriven et al., 2009). Here the green points form the non-dominated front, the reference point is gray. The quadrant in objective space is divided in equal sectors by M dashed lines. The metric value is calculated as the number of sectors with at least one point (here: 4) divided by the total number of sectors $M + 1$ (here: 8). In our experiments, we use $M = 360$ (larger M means a more fine-grained metric calculation (Scriven et al., 2009)).

G. Pseudocode

Algorithm 1 Procedure to sort solutions in MO-PBT (`sortPopulation`)

▷ For the sake of implementation simplicity, we sort all fronts regardless of their sizes (the overhead of this operation is negligible) even though not all of them might be needed to select the top and the bottom solutions of the population.

Input: Population $\mathcal{P}$
Output: Sorted population $\mathcal{P}^*$
$(F^1, \ldots, F^R) \leftarrow$ non-dominated sort of P ▷ F^i is the ith non-dominated front
$\mathcal{P}^* \leftarrow \{\arg\max_{v \in F^1} f_1(v)\}$ ▷ add the solution with the largest f_1 first
$F^1 \leftarrow F^1 \setminus \{v\}$
for $i = 1, \ldots, R$ **do**
  while $F^i \neq \emptyset$ **do**
    $nextToAdd \leftarrow \arg\max_{v \in F^i} \arg\min_{v' \in \mathcal{P}^*} D(f(v), f(v'))$ ▷ D is Euclidean distance in the objective space
    $F^i \leftarrow F^i \setminus \{nextToAdd\}$
    $\mathcal{P}^* \leftarrow \mathcal{P}^* \cup \{nextToAdd\}$
  end while
end for
return $\mathcal{P}^*$

Algorithm 2 Exploit in MO-PBT (`exploit`)

Input: population $\mathcal{P}$, truncation selection parameter τ
Output: population $\mathcal{P}$ with changed weights and hyperparameters (in-place)
$\mathcal{P} \leftarrow$ `sortPopulation`$(\mathcal{P})$
for $p \in \tau|\mathcal{P}|$ bottom solutions of $\mathcal{P}$ **do**
  select a solution d from $\tau|\mathcal{P}|$ top solutions
  $p_\theta \leftarrow d_\theta$ ▷ copy weights
  $p_h \leftarrow d_h$ ▷ copy hyperparameters
  $p_h \leftarrow$ `explore`(p_h) ▷ perturb hyperparameters
end for
return $\mathcal{P}$

Algorithm 3 Explore in MO-PBT (`explore`)

Input: hyperparameter value h, hyperparameter values domain $\mathcal{H} = v_1, \ldots v_M$, resample probability p
Output: perturbed hyperparameter value h'
$x \leftarrow$ sample from uniform distribution $U(0, 1)$
if $x < p$ **then**
  $h' \leftarrow$ uniformly sampled value from $\mathcal{H}$ ▷ with small probability it is resampled
else
  $shift \leftarrow$ uniformly sampled value from $[0, 1, 2, 3]$
  $shift \leftarrow -$ *shift* with probability 0.5
  $h' \leftarrow h+$*shift* ▷ local perturbation
end if
return h'

H. Search spaces

H.1. Precision/Recall

Hyperparameter	Range of values	Scale	Number of values
Attention dropout	[0, 0.8]	linear	10
FFN dropout	[0, 0.8]	linear	10
Residual dropout	[0, 0.8]	linear	10
Weight decay	[0, 0.1]	log	10
Class weight in CE loss	[0.1, 0.9]	linear	10

Table 1. Search space for the Precision/Recall task.

H.2. Accuracy/Fairness

Hyperparameter	Range of values	Scale	Number of values
Attention dropout	[0, 0.8]	linear	10
FFN dropout	[0, 0.8]	linear	10
Residual dropout	[0, 0.8]	linear	10
Weight decay	[0, 0.1]	log	10
Class weight in fairness loss	[0, 10]	log	10

Table 2. Search space for the Accuracy/Fairness tasks on the Adult dataset.

Hyperparameter	Range of values	Scale	Number of values
RandAugment N	[0, 4]	linear	5
RandAugment M	[0, 9]	linear	10
CutOut probability	[0, 1]	linear	10
CutOut magnitude	[0, 9]	linear	10
Class weight in fairness loss	[0, 10]	log	10

Table 3. Search space for the Accuracy/Fairness tasks on the CelebA dataset.

H.3. Accuracy/Adversarial robustness

Hyperparameter	Range of values	Scale	Number of values
RandAugment N	[0, 4]	linear	5
RandAugment M	[0, 9]	linear	10
CutOut probability	[0, 1]	linear	10
CutOut magnitude	[0, 9]	linear	10
Coefficient in the TRADES loss	[0, 10]	log	10

Table 4. Search space of the Accuracy/Robustness task.

I. Tabulated results

Problem	Dataset	random search	obj. 1	obj. 2	obj. 3	rand. scalar.	max. scalar.	MO-PBT	MO-ASHA	BO-MO-ASHA
Precision/Recall	Adult	68.69 ± 0.50	69.61 ± 0.27	63.80 ± 0.90	-	68.13 ± 0.97	68.79 ± 0.57	$\mathbf{70.59 \pm 0.13}$	67.89 ± 0.52	66.10 ± 1.56
	Higgs	36.88 ± 0.26	38.14 ± 0.36	33.40 ± 0.61	-	36.50 ± 0.43	37.40 ± 0.56	$\mathbf{39.22 \pm 0.22}$	37.46 ± 0.32	36.34 ± 0.75
	Click	35.22 ± 0.45	34.65 ± 0.50	34.33 ± 0.86	-	35.01 ± 0.39	34.89 ± 0.52	$\mathbf{37.39 \pm 0.24}$	35.69 ± 0.43	35.66 ± 0.58
Fairness: Acc./DSP	CelebA	16.51 ± 0.09	16.53 ± 0.03	15.17 ± 0.12	-	16.59 ± 0.07	16.51 ± 0.02	$\mathbf{16.76 \pm 0.08}$	16.39 ± 0.06	16.40 ± 0.05
	Adult	3.64 ± 0.04	3.69 ± 0.02	2.80 ± 0.11	-	3.55 ± 0.11	3.55 ± 0.17	$\mathbf{3.81 \pm 0.02}$	3.74 ± 0.01	3.73 ± 0.03
Fairness: Acc./DSP/EOdd	CelebA	11.09 ± 0.08	11.11 ± 0.05	10.10 ± 0.11	10.13 ± 0.13	11.00 ± 0.07	11.20 ± 0.10	$\mathbf{11.43 \pm 0.04}$	11.12 ± 0.07	11.15 ± 0.10
	Adult	1.62 ± 0.02	1.64 ± 0.02	1.30 ± 0.11	1.32 ± 0.09	1.63 ± 0.05	1.47 ± 0.02	$\mathbf{1.76 \pm 0.02}$	1.70 ± 0.02	1.68 ± 0.02
Acc./Robustness	CIFAR-10	33.84 ± 0.28	24.65 ± 0.81	33.94 ± 0.26	-	34.74 ± 0.48	34.51 ± 0.29	$\mathbf{35.40 \pm 0.09}$	33.62 ± 0.59	33.82 ± 0.32
	CIFAR-100	17.90 ± 0.27	12.06 ± 0.15	17.44 ± 0.13	-	17.77 ± 0.25	18.13 ± 0.24	$\mathbf{18.65 \pm 0.11}$	16.98 ± 0.53	16.92 ± 1.07

Table 5. Obtained hypervolume (*larger values are better*) data for all algorithms and all tasks. Average and standard deviation values of the best obtained hypervolume over multiple runs are provided. Obj.1, obj 2., and obj. 3 denote single-objective PBT applied to optimizing the corresponding objective of the task. Acc. denotes accuracy. For better readability, all values are multiplied by 100.

Problem	Dataset	random search	obj. 1	obj. 2	obj. 3	rand. scalar.	max. scalar.	MO-PBT	MO-ASHA	BO-MO-ASHA
Precision/Recall	Adult	68.57 ± 0.25	69.38 ± 0.17	64.94 ± 0.70	-	68.27 ± 0.64	68.63 ± 0.52	$\mathbf{69.95 \pm 0.18}$	67.47 ± 0.65	65.92 ± 1.41
	Higgs	36.01 ± 0.53	37.42 ± 0.31	32.74 ± 0.80	-	35.67 ± 0.54	36.49 ± 0.51	$\mathbf{38.31 \pm 0.19}$	36.56 ± 0.38	35.21 ± 0.95
	Click	34.06 ± 0.53	33.60 ± 0.51	33.60 ± 0.74	-	34.14 ± 0.40	34.16 ± 0.56	$\mathbf{36.38 \pm 0.25}$	34.81 ± 0.87	34.95 ± 0.63
Fairness: Acc./DSP	CelebA	16.47 ± 0.09	16.50 ± 0.05	15.28 ± 0.07	-	16.57 ± 0.09	16.49 ± 0.05	$\mathbf{16.79 \pm 0.04}$	16.34 ± 0.10	16.33 ± 0.05
	Adult	3.69 ± 0.02	3.68 ± 0.01	3.04 ± 0.12	-	3.65 ± 0.08	3.65 ± 0.10	$\mathbf{3.79 \pm 0.02}$	3.75 ± 0.02	3.73 ± 0.02
Fairness: Acc./DSP/EOdd	CelebA	10.97 ± 0.09	10.99 ± 0.03	10.09 ± 0.07	10.14 ± 0.10	10.94 ± 0.05	11.08 ± 0.11	$\mathbf{11.36 \pm 0.04}$	11.04 ± 0.07	11.02 ± 0.09
	Adult	1.69 ± 0.01	1.68 ± 0.02	1.42 ± 0.11	1.44 ± 0.10	1.72 ± 0.04	1.60 ± 0.03	$\mathbf{1.80 \pm 0.01}$	1.75 ± 0.01	1.74 ± 0.01
Acc./Robustness	CIFAR-10	33.40 ± 0.40	24.54 ± 0.91	33.45 ± 0.24	-	34.40 ± 0.45	34.13 ± 0.27	$\mathbf{34.99 \pm 0.13}$	32.97 ± 0.69	33.30 ± 0.31
	CIFAR-100	18.30 ± 0.29	11.98 ± 0.20	17.69 ± 0.15	-	18.13 ± 0.27	18.50 ± 0.26	$\mathbf{19.03 \pm 0.16}$	17.42 ± 0.61	17.25 ± 1.28

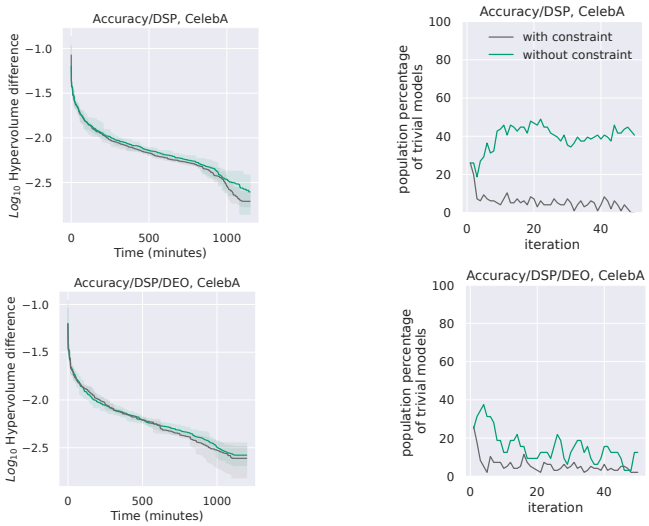
Table 6. Obtained hypervolume (*larger values are better*) data for all algorithms and all tasks on *test* data subsets. Average and standard deviation values of the best obtained hypervolume over multiple runs are provided. Obj.1, obj. 2, and obj. 3 denote single-objective PBT applied to optimizing the corresponding objective of the task. Acc. denotes accuracy. For better readability, all values are multiplied by 100.

Problem	Dataset	random search	obj. 1	obj. 2	rand. scalar.	max. scalar.	MO-PBT	MO-ASHA	BO-MO-ASHA
Precision/Recall	Adult	29.36 ± 1.63	21.39 ± 1.01	23.68 ± 2.02	29.34 ± 3.36	26.12 ± 1.41	$\mathbf{40.39 \pm 2.62}$	28.14 ± 1.71	25.65 ± 2.40
	Higgs	27.87 ± 2.50	21.99 ± 1.83	14.38 ± 1.71	22.33 ± 2.15	17.76 ± 1.21	$\mathbf{35.71 \pm 2.07}$	20.08 ± 1.89	19.22 ± 2.41
	Click	30.11 ± 2.79	17.17 ± 1.81	20.39 ± 2.38	27.70 ± 1.65	18.39 ± 1.92	$\mathbf{34.96 \pm 2.90}$	25.71 ± 1.75	26.29 ± 2.19
Fairness: Acc./DSP	CelebA	$\mathbf{8.14 \pm 0.95}$	7.20 ± 0.63	4.25 ± 0.26	6.26 ± 0.89	5.37 ± 0.62	7.20 ± 0.70	7.42 ± 0.66	7.87 ± 1.22
	Adult	5.90 ± 1.44	6.93 ± 1.26	2.16 ± 0.41	4.60 ± 1.44	3.02 ± 0.84	6.76 ± 1.26	$\mathbf{7.26 \pm 1.09}$	5.57 ± 1.71
Acc./Robustness	CIFAR-10	4.38 ± 0.73	4.80 ± 0.13	3.05 ± 0.80	4.60 ± 0.57	5.60 ± 0.44	$\mathbf{9.58 \pm 0.51}$	2.99 ± 0.54	3.32 ± 0.63
	CIFAR-100	5.43 ± 1.22	5.45 ± 0.47	3.71 ± 0.48	4.99 ± 0.72	6.70 ± 0.27	$\mathbf{10.75 \pm 1.37}$	3.49 ± 1.06	4.38 ± 0.98

Table 7. Obtained coverage metric introduced in (Scriven et al., 2009) and illustrated in Figure 17 (*larger values are better*) data for all algorithms and all bi-objective tasks. Average and standard deviation values of the best obtained coverage value over multiple runs are provided. Obj.1, obj 2. denote single-objective PBT applied to optimizing the corresponding objective of the task. Acc. denotes accuracy. For better readability, all values are multiplied by 100.

J. Effects of the constraints on the performance

We noticed that on the CelebA Accuracy/Fairness task, many trivial classifiers (they always predict one of the classes, which leads to a perfect fairness score) appear in the population. However, we are not interested in obtaining trivial classifiers, as such a classifier is the most straightforward way to obtain a perfectly fair model, but it is not particularly useful. Therefore, we can consider a solution to be feasible if its accuracy is better than the accuracy of a trivial classifier. If constraints are imposed, the domination criterion can be extended to *constraint domination* (Deb et al., 2002) (the solutions that violate constraints are considered to be dominated by the solutions that do not). We hypothesized that with such constraints, the algorithm can become more effective in finding non-trivial solutions. The imposed constraint on the accuracy to be higher than the accuracy of a trivial classifier reduces the number of trivial classifiers in the population, but the hypervolume-measured performance improvement is subtle, as shown in Figure 18. Therefore, we did not use this technique in the main experiments.



(a) Performance over time

(b) Percentage of trivial classifiers in the population

Figure 18. Comparison of MO-PBT performance on the CelebA Accuracy/DSP and Accuracy/DSP/DEO tasks with and without constraint that demands the models to have better accuracy than trivial classifiers.