LibMOON: A Gradient-based MultiObjective OptimizatioN Library in PyTorch

Xiaoyuan Zhang♣, Liang Zhao♣, Yingying Yu♣, Xi Lin♣, Yifan Chen♠, Han Zhao♡, Qingfu Zhang♣*
♣ CityUHK, ♠ HKBU, ♡ UIUC.

Abstract

Multiobjective optimization problems (MOPs) are prevalent in machine learning, with applications in multi-task learning, fairness, robustness, and more. Unlike single-objective optimization, which aggregates objectives into a scalar through weighted sums, MOPs focus on generating specific or diverse Pareto solutions and learning the entire Pareto set directly. Existing MOP benchmarks primarily focus on evolutionary algorithms, which are zeroth-order or meta-heuristic methods that fail to leverage higher-order objective information and cannot scale to large models. To address these challenges, we introduce `LibMOON`, the first multiobjective optimization library supporting state-of-the-art gradient-based methods, offering a fair and comprehensive benchmark, and open-sourced for the community.

1 Introduction

Multiobjective Optimization Problems (MOPs) are ubiquitous in machine learning. For instance, trustworthy machine learning (e.g., algorithmic fairness) problems balancing the fairness level and the accuracy level [1, 2]; in robotics, it is necessary to balance several objectives (e.g., forward speed and energy consumption) [3, 4]; similarly, recommendation systems face potentially conflicting objectives, such as novelty, accuracy, and diversity [5, 6, 7]. For all the applications above, the underlying optimization problem involves an MOP with m objectives and can be (informally) defined as:

$$\min_{\boldsymbol{\theta}} \boldsymbol{L}(\boldsymbol{\theta}) = (L_1(\boldsymbol{\theta}), \ldots, L_m(\boldsymbol{\theta})), \tag{1}$$

where $L_1(\boldsymbol{\theta}), \ldots, L_m(\boldsymbol{\theta})$ denote m (potentially) conflicting objectives and we denote the size of the model parameter as $n := |\boldsymbol{\theta}|$. Note that as informally defined above, Equation (1) is a vector optimization problem that does not necessarily admit a total ordering. For a non-trivial MOP, no single solution can attain the minimum of all objectives simultaneously. To compare two solutions for an MOP, we introduce the concepts of *dominance* and *Pareto optimality* [8]. We say that solution $\boldsymbol{\theta}^{(a)}$ dominates $\boldsymbol{\theta}^{(b)}$ when $\boldsymbol{\theta}^{(a)}$ satisfies $L_i(\boldsymbol{\theta}^{(a)}) \leq L_i(\boldsymbol{\theta}^{(b)})$ for all $1 \leq i \leq m$, with at least one strict inequality. A solution is Pareto optimal if no other solution in the feasible region dominates it. The set of all Pareto optimal solutions is called the Pareto set (PS), and its image set is called the Pareto front (PF).

Over the last few decades, multiobjective evolutionary algorithms (MOEAs) emerged as a widely used methodology for addressing MOPs due to their ability to find diverse and approximate PS. Several popular MOEA libraries have emerged, including PlatEMO (Matlab) [9], Pagmo (C++) [10], and Pymoo (Python) [11]. Compared to MOEAs, gradient-based multiobjective optimization (MOO) methods are particularly suitable for large-scale machine learning tasks involving thousands to millions of neural network parameters. In contrast, gradient-based MOO methods can only find

*Corresponding to Prof. Qingfu Zhang. Contact: {xzhang2523-c@my.,qingfu.zhang@}cityu.edu.hk.

Pareto *stationary* solutions—solutions that cannot be *locally* improved in all objectives, in practice, Pareto stationary solutions well approximate global Pareto solutions for deep learning tasks.

With the growing need for gradient-based MOO methods (e.g., [12, 13, 4, 14]) for large-scale neural networks, there is a pressing need for the development of a standard library to benchmark related algorithms and problems. For this reason, we introduce `LibMOON`, the first modern gradient-based MOO library supporting over twenty state-of-the-art (SOTA) methods. We summarize our contributions as follows:

1. We introduce the *first* modern gradient-based MOO library which is implemented in PyTorch [15] and carefully designed to support GPU acceleration. `LibMOON` supports MO machine learning problems such as MO classification, MO regression, MO distribution matching, etc, along with their real-world applications.
2. `LibMOON` supports over twenty state-of-the-art (SOTA) gradient-based MOO methods for constructing the PS and PF, including MOO solvers that use *finite* solutions to approximate the entire PS/PF [16, 17]; Pareto Set Learning (PSL) solvers [18, 19] that approximate the *entire* PS/PF with a single neural model; and multi-objective Bayesian Optimization (MOBO) solvers, which are designed to minimize the need for avoiding frequent function evaluations.
3. We have open-sourced `LibMOON` on Github[2] with document at LibMOON Docs[3]. `LibMOON` can be installed via "`pip install libmoon`" as an off-the-shelf gradient-based multiobjective package for easy use.

Notation. Bold letters represent vector (e.g., $\boldsymbol{\lambda}$ denotes a preference vector). $\boldsymbol{x}^{(k)}$ denotes vector $\boldsymbol{x}$ at k-th iteration and x_k denotes the k-th entry of $\boldsymbol{x}$. The preference vector $\boldsymbol{\lambda}$ lies in the m-dim simplex ($\boldsymbol{\Delta}_m$), satisfying $\sum_{i=1}^{m} \lambda_i = 1$ and $\lambda_i \geq 0$. The decision network parameter $\boldsymbol{\theta}$ has a size of n. For two m-D vectors $\boldsymbol{x}^{(a)}$ and $\boldsymbol{x}^{(b)}$, $\boldsymbol{x}^{(a)} \preceq \boldsymbol{x}^{(b)}$ means $\boldsymbol{x}_i^{(a)} \leq \boldsymbol{x}_i^{(b)}$ for all $i \in [m]$; $\boldsymbol{x}^{(b)}$, $\boldsymbol{x}^{(a)} \preceq_{\text{strict}} \boldsymbol{x}^{(b)}$ means that $\boldsymbol{x}_i^{(a)} \leq \boldsymbol{x}_i^{(b)}$ for all $i \in [m]$ and for at least one strict inequality. $\boldsymbol{x}^{(a)} \prec \boldsymbol{x}^{(b)}$ means that $\boldsymbol{x}_i^{(a)} < \boldsymbol{x}_i^{(b)}$ for all $i \in [m]$. Refer to Table 11 for full notations.

2 Related works

2.1 Gradient-based multiobjective optimization

Gradient-based MOO has a rich research literature. The well-known convex optimization book [20, Chap 4] outlines how linear scalarization can transform a MOO problem into a single-objective optimization (SOO) problem. However, for much of the past few decades, gradient-based methods have not been the primary focus for MOO. Instead, MOEAs have gained more attention due to their population-based nature, which is well-suited for approximating the PS and finding diverse solutions. In recent years, however, gradient-based MOO has experienced a resurgence, particularly in (deep) machine learning, where these methods scale better with the number of decision variables. A pivotal contribution in this area is the MGDA-UB [12] method, which introduced MOO techniques into deep learning by casting multi-task learning (MTL) as a MOO problem. Since then, many approaches have followed, including EPO [16], Pareto Multi-Task Learning (PMTL) [13], MOO with Stein Variational Gradient Descent (MOO-SVGD) [17], and some methods for learning the entire PS (Pareto set learning) [19, 18, 21, 22, 23, 24].

2.2 Multiobjective optimization libraries

A number of multiobjective libraries exist in the literature. We summarize the high-level comparison between `LibMOON` and existing libraries in Table 1 and discuss the detailed differences as follows.

LibMTL [25] is a Python-based library for multitask learning. LibMTL aims to find a single network to benefit all tasks, such as finding a benign updating direction or optimizing a network architecture. In contrast, `LibMOON` addresses inherent trade-offs in machine learning problems, where improving one objective inevitably worsens others, and explores the distribution of Pareto solutions.

[2]`https://github.com/xzhang2523/libmoon`

[3]`https://libmoondocs.readthedocs.io/en/latest/`

jMetal [26], **Pymoo** [27] and **PlatEMO** [9] are Java, python and Matlab frameworks for MOEAs, supporting popular methods such as NSGA-III [28, 29], MOEA/D [30], and SMS-EMOA [31]. Pymoo allows flexible algorithm customization with user-defined operators and data visualization. PlatEMO is a MATLAB-based multiobjective optimization tool supporting over 160 MOEAs and a comprehensive test problems, including sparse, high-cost, large-scale, and multimodal. PlatEMO also contains a number of metrics and supporting visualization during the optimization process.

Pagmo [10] is a C++ library for parallel multiobjective global optimization, utilizing evolutionary algorithms and gradient-based methods like simplex, SQP, and interior-point techniques. It supports constrained, unconstrained, single- and multi-objective, continuous, integer, stochastic, and deterministic optimization problems.

EvoTorch [32] and **EvoX** [33]. EvoTorch accelerates evolutionary algorithms in PyTorch, while EvoX scales them to large, complex problems with GPU-accelerated parallel execution for single and multiobjective tasks, including synthetic problems and reinforcement learning.

Table 1: Previous MOO libraries and `LibMOON`.

Name	Language	Year	Key Features
Pymoo	Python	2020	(1) Evolutionary computation (EC) (2) Zero-order methods (3) Diverse problem types
jMetal	Java	2011	(1) Single-/multi-objective optimization (2) Parallel algorithms (3) Diverse problem types
PlatEMO	Matlab	2017	(1) Over 160 MOEAs (2) Various figure demonstrations (3) Powerful and friendly GUI
Pagmo	C++	2020	(1) Global optimization (2) Parallel optimization
LibMTL	Python	2023	(1) Unified codebase (2) Comprehensive SOTA MTL methods (3) Flexible extension for new methods
EvoTorch	Python	N/A	(1) Distribution-based search algorithms (2) Population-based search algorithms (3) Multiple CPUs, GPUs, computers
EvoX	Python	2024	(1) GPU acceleration optimization (2) Single-/multi-objective optimization (3) Neuroevolution/RL tasks
LibMOON	Python	2024	(1) GPU-accelerated gradient solvers (2) Pareto set learners (3) Large-scale (millions # params.) ML tasks

3 `LibMOON`: A gradient-based MOO library in PyTorch

This section introduces `LibMOON`. We introduce its framework in Section 3.1, and briefly introducing its supporting **problems** and **metrics**. Then we introduce supported **solvers** in Sections 3.2 to 3.4.

3.1 Framework

Figure 1 demonstrates the components of `LibMOON`, including three categories of solvers: MOO solvers aiming to find a finite set of Pareto solutions satisfying certain requirements, Pareto set learning (PSL) solvers aiming to learn whole PS with a single model, and MOBO solvers aiming to solve expensive MO problems. Each solver category is designed in a highly modulized way so that new solvers can be easily incorporated into `LibMOON` by rewriting only a small portion of code, e.g., the specific algorithm of gradient manipulations[4]. MOO and PSL solvers support all synthetic, MTL, and real-world (RE) problems, while MOBO solvers support synthetic and RE problems.

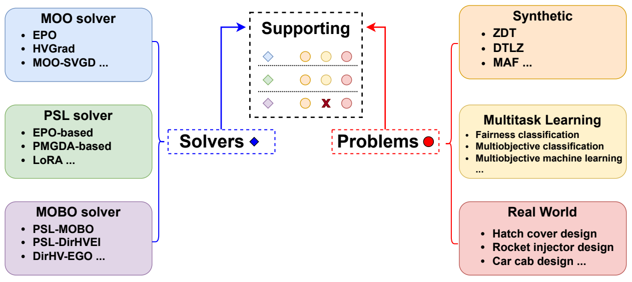


Figure 1: **Supported solvers and problems in** `LibMOON`: `LibMOON` addresses synthetic, real-world and MTL problems with three categories of solvers: MOO, PSL and MOBO solvers.

Supported problems. `LibMOON` currently supports three categories of methods, synthetic problems, MTL problems, and RE problems.

[4]An example of adding a new solver is provided in the `LibMOON` Doc.

Supported metrics. `LibMOON` supports several metrics for evaluation, (1) hypervolume (HV), (2) inverted general distance (IGD), (3) fill distance (FD), (4) minimal distance ($\mathrm{l_{min}}$), (5) smooth minimal distance ($\mathrm{sl_{min}}$), (5) Spacing, (6) Span, (7) penalty-based intersection (PBI), (8) inner product (IP), (9) cross angle (ϑ). Full descriptions of these metircs are provided in Appendix A.3.

Table 2: Supported MO machine learning problems. L_1: the first objective. L_2: the second objective.

Method	L_1	L_2
Fairness classification [34]	BCE	DEO
MO classification [13]	CE - BR	CE - UL
MO regression [35]	MSE	MSE
MO distribution matching	$D(\cdot\|\cdot)$	$D(\cdot\|\cdot)$

BCE: Binary Cross Entropy; DEO [36]: Difference of Equality of Opportunity; CE: Cross Entropy; BR: Bottom Right; UL: Upper Left; MSE: Mean Square Error.

3.2 MOO solvers

In this paper, MOO solvers refer to solvers that find a set of Pareto solutions. The simplest and most commonly used method is *linear scalarization* [35], which converts a MOO problem to a single objective optimization problem through some aggregation functions.

Aggregation-based methods. A straightforward way is to use some aggregation functions $g_{\boldsymbol{\lambda}}(\cdot) : \mathbb{R}^m \mapsto \mathbb{R}$ to convert a MOP to a single objective optimization problem. The reason that optimizing this converted single objective optimization problem will yield Pareto optimal solutions is due to the following two theorems.

Theorem 1 (Adapted from Theorem 2.6.2 [37])**.** *If $g_{\boldsymbol{\lambda}}(\cdot)$ is **strictly decreasing** w.r.t vector $\boldsymbol{L}(\boldsymbol{\theta})$, i.e., $g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(a)})) < g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(b)}))$ when $\boldsymbol{L}_i(\boldsymbol{\theta}^{(a)}) \preceq_{\text{strict}} \boldsymbol{L}_i(\boldsymbol{\theta}^{(b)})$, then the optimal solution θ^* of $g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))$ serves as a **Pareto optimal** solution for the original MOP.*

Proof. See Mitten's book [37], Page 22. Similarly, for decreasing aggregation functions, we have the following theorem.

Theorem 2. *If $g_{\boldsymbol{\lambda}}$ is **decreasing** w.r.t. vector $\boldsymbol{L}(\boldsymbol{\theta})$ (i.e., $g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(a)})) \leq g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(b)}))$ when $\boldsymbol{L}(\boldsymbol{\theta}^{(a)}) \preceq_{\text{strict}} \boldsymbol{L}(\boldsymbol{\theta}^{(b)})$, then the optimal solution θ^* of $g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))$ serves as a **weakly Pareto optimal** solution for the original MOP.*

Proof. Similar to the proof of Mitten's book [37], Page 22.

In Theorem 2, *weakly Pareto optimality* means that for a solution $\boldsymbol{\theta}^{(a)}$, no other solutions $\boldsymbol{\theta}'$ can strictly dominate it, i.e., $\boldsymbol{L}_i(\boldsymbol{\theta}') < \boldsymbol{L}_i(\boldsymbol{\theta}^{(a)})$ for all $i \in [m]$. Some common aggregation functions include the linear scalarization (LS) function, where $g^{\text{LS}}_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta})) = \sum_{i=1}^m \lambda_i L_i(\boldsymbol{\theta})$, the Tchebycheff function, where $g^{\text{Tche}}_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta})) = \max_{i\in[m]} \lambda_i \cdot (L_i(\boldsymbol{\theta}) - z_i)$ ($\boldsymbol{z}$ is a reference point), Penalty-Based Intersection (PBI) function [30], and COSMOS function [34]. For expressions and other aggregation functions, please refer to Appendix A.2.

For a preference vector $\boldsymbol{\lambda} \succ \boldsymbol{0}$, $g^{\text{LS}}_{\boldsymbol{\lambda}}(\cdot)$ is a *strictly decreasing function*, hence directly optimizing $g^{\text{LS}}_{\boldsymbol{\lambda}}(\cdot)$ yields *Pareto optimal solutions* (by Theorem 1). However, for any $\boldsymbol{\lambda} \in \boldsymbol{\Delta}_m$, optimizing $g^{\text{Tche}}_{\boldsymbol{\lambda}}(\cdot)$ only yields weakly *Pareto optimal solutions* (by Theorem 2). For an improper setting of the weight factor μ (see Appendix A.2 item 1 and 6), the optimal solution of $g^{\text{PBI}}_{\boldsymbol{\lambda}}(\cdot)$ or $g^{\text{COSMOS}}_{\boldsymbol{\lambda}}(\cdot)$ can be non-(weakly) Pareto optimal solutions of the original MOP.

An aggregation function is optimized by gradient descent in `LibMOON` via backpropagation, i.e, $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \boldsymbol{d}_k = \boldsymbol{\theta}^{(k)} - \eta \frac{\partial g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}^{(k)}}$, where $\boldsymbol{d}^{(k)}$ is called the updating direction at the kth iteration. The gradient term $\frac{\partial g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}$ can be decomposed into two parts: $\frac{\partial g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} = \frac{\partial g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))}{\partial \boldsymbol{L}(\boldsymbol{\theta})} \frac{\partial \boldsymbol{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, assuming that $\frac{\partial \boldsymbol{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ exists. If $g_{\boldsymbol{\lambda}}(\cdot)$ is differentiable, then $\frac{\partial g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))}{\partial \boldsymbol{L}(\boldsymbol{\theta})}$ is the standard gradient. In cases where $g_{\boldsymbol{\lambda}}(\cdot)$ is non-differentiable, $\frac{\partial g_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}))}{\partial \boldsymbol{L}(\boldsymbol{\theta})}$ could be taken as a sub-gradient.

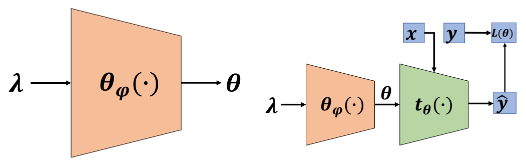
Gradient manipulation-based methods. Besides directly optimizing the aggregation function, several so-called *"gradient manipulation methods"* solve an updating direction $\boldsymbol{d}^{(k)}$ using gradient information for each iteration for some specific purpose. For example, as listed in Table 3, EPO [16] aims to find "exact Pareto solutions" (the intersection points of the Pareto front and preference vectors), HVGrad [38] aims to maximize the hypervolume of a set of solutions, MOO-SVGD [17] aims to find diverse solutions, PMTL [13] aims to identify sector-constrained Pareto solutions, and ExcessMTL [39] aims to find a Pareto solution with the same excess risk across all the objectives. Interestingly, until now, all these gradient manipulation methods can be implemented in two steps: ① calculating a dynamic weight vector $\tilde{\boldsymbol{\alpha}}$ and then ② performing backpropagation on a generalized aggregation function $\tilde{g}_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(k)}))$, where $\tilde{g}_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(k)})) = \sum_{i=1}^{m} \tilde{\alpha}_i L_i(\boldsymbol{\theta})$. At each iteration, gradient manipulation methods can be equivalently expressed as updating the gradient of its induced generalization aggregation function $\tilde{g}_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}^{(k)}))$. The weight vector $\tilde{\boldsymbol{\alpha}}$ are achieved by solving a linear programming (LP) problem (e.g., [16, Eq. 24]) in EPO, a quadratic programming (QP) problem (e.g., [13, Eq. 14]) in PMTL, or other more complex algorithms as used to calculate the hypervolume gradient [43].

Table 3: MOO solvers, properties, and complexities.

Method	**Solution Property**	**Complexity**	**Pref.**
EPO [16]	Exact solutions	$O(m^2nK)$	✓
HVGrad [38]	Solutions with maximal HV	$O(m^2nK^2)$	×
MGDA-UB [12]	Random solutions	$O(m^2nK)$	×
MOO-SVGD [17]	Diversity by particles repulsion	$O(m^2nK^2)$	×
PMGDA [40]	Solutions under specific demands	$O(m^2nK)$	✓
PMTL [13]	Solutions in sectors	$O(m^2nK^2)$	×
Random [41]	Random solutions	$O(m^2nK)$	×
Agg-LS [37]	Convex parts of a PF	$O(mnK)$	✓
Agg-Tche [30]	Exact solutions	$O(mnK)$	✓
Agg-mTche [42]	Exact solutions	$O(mnK)$	✓
ExcessMTL [39]	Exact solutions	$O(mnK)$	✓
Agg-PBI [30]	Approximate exact solutions	$O(mnK)$	✓
Agg-COSMOS [34]	Approximate exact solutions	$O(mnK)$	✓
Agg-SmoothTche [22]	Approximate exact solutions	$O(mnK)$	✓

m: number of objectives. n: number of decision variables. K: number of subproblems. m is usually small (e.g., 2-4), K is relatively large (e.g., 20-40), and n is particularly large (e.g., 10,000). Therefore, m^2 is not a big concern, while K^2 and n^2 are big concerns. Complexity is for time complexity, and Pref. denotes whether this method is preference-based or not.

Some MOO solvers accept preference vectors $\boldsymbol{\lambda}$ as input, termed *preference-based*, affecting Pareto solution positions. The others, called *preference-free*, do not accept preferences, such as those maximizing dominated hypervolume. A summary of these solvers is in Table 3.


$\boldsymbol{\lambda}$
$\boldsymbol{\theta}_{\boldsymbol{\varphi}}(\cdot)$
$\boldsymbol{\theta}$
$\boldsymbol{x}$
$\boldsymbol{y}$
$\boldsymbol{L}(\boldsymbol{\theta})$
$\boldsymbol{t}_{\boldsymbol{\theta}}(\cdot)$
$\hat{\boldsymbol{y}}$

(a) Synthetic problem (b) MultiTask Learning

Figure 2: Architecture of Pareto models.

Zero-order optimization. In the previous discussion, we assume that all the gradients of objective functions $\nabla L_i(\boldsymbol{\theta})$ can be easily achieved via backward propagation. However, for some black-box optimization problems, $\nabla L_i(\boldsymbol{\theta})$'s may not easily be achieved. Therefore, `LibMOON` not only supports first-order optimization, but also supports zero-order optimization methods with estimated gradients $\hat{\nabla} L_i(\boldsymbol{\theta})$ such as evolutionary strategy (ES) [44].

3.3 Pareto set learning solvers

`LibMOON` also supports Pareto Set Learning (PSL) solvers, which trains a model with parameter $\boldsymbol{\phi}$ to approximate the *entire* PS/PF. A Pareto model is denoted as $\boldsymbol{\theta}_{\boldsymbol{\phi}}(\cdot) : \boldsymbol{\Delta}_m \mapsto \mathbb{R}^n$ with input as a preference vector and output as a Pareto solution.

PSL Architecture. Pareto models vary in structure. For synthetic problems, the simplest model is a fully-connected neural network that takes a preference as input and outputs the corresponding Pareto solution. In multitask learning, the input is a pair $(\boldsymbol{x}, \boldsymbol{y})$ from dataset $\mathcal{D}$, and the decision variable $\boldsymbol{\theta}$ represents the target network's parameter, with ϕ as the hypernetwork's parameter [45]. The loss vector is calculated as $\boldsymbol{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}(\ell(\boldsymbol{t}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}))$, where ℓ is a basic loss function like cross-entropy or mean square error. Structures of these two models are illustrated by Figure 2. Besides these two models, `LibMOON` also supports LoRA (low rank adaptation)-based PSL [46, 21, 47], which admits a low rank adaptation structure and other structures. PSL structures are decoupled from the training loss and used as plug-ins.

PSL Training. Goal of PSL is to find a model with parameter $\boldsymbol{\phi}$ optimizing the PSL loss ℓ_{psl},

$$\min_{\boldsymbol{\phi}} \ell_{\text{psl}} = \mathbb{E}_{\boldsymbol{\lambda}\sim\text{Dir}(\boldsymbol{p})} \tilde{g}_{\boldsymbol{\lambda}}(\boldsymbol{L}(\boldsymbol{\theta}_{\boldsymbol{\phi}}(\boldsymbol{\lambda}))), \qquad \text{where} \quad \boldsymbol{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}(\ell(\boldsymbol{t}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})).$$

In the above formulation, $\text{Dir}(\boldsymbol{p})$ is a Dirichlet distribution with hyperparameter $\boldsymbol{p}$. $\tilde{g}_{\boldsymbol{\lambda}}(\cdot)$ can either be a generalized aggregation function as introduced in the previous section or a normal aggregation function. The gradient of ℓ_{psl} can be estimated by the chain rule:

Table 4: Comparison of different PSL methods.

Method	**Property**	Vector $\tilde{\boldsymbol{\alpha}}$	Matrix $\boldsymbol{B}$
EPO-based PSL [18]	Exact solutions	MOO solvers	BP
PMGDA-based PSL [40]	Solutions under specific demands	MOO solvers	BP
Aggregation-based PSL [18]	Solutions with optimal aggregation values	BP	BP
Evolutionary PSL [48]	Mitigating local minima by ES	BP	ES
LoRA PSL [46, 21, 47]	A lighter Pareto model architecture	BP	BP

BP: backward propagation, ES: evolutionary strategy.

$$\underbrace{\frac{\partial \ell_{\mathrm{psl}}}{\partial \boldsymbol{\phi}}}_{1\times D} = \mathbb{E}_{\boldsymbol{\lambda}\sim\mathrm{Dir}(\boldsymbol{p})} \underbrace{\frac{\partial \tilde{g}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{L}}}_{\tilde{\boldsymbol{\alpha}}:(1\times m)} \cdot \underbrace{\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{\theta}}}_{\boldsymbol{B}:(m\times n)} \cdot \underbrace{\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\phi}}}_{\boldsymbol{C}:(n\times D)} . \tag{2}$$

Empirically, the expectation involved in Equation (2) is estimated using a mini-batch of K preferences. The gradient vector $\tilde{\boldsymbol{\alpha}} = (\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m)$ is computed by MOO solvers as introduced in the previous section. The gradient matrix $\boldsymbol{B}$ can be calculated either via backpropagation (when gradients are easily obtained) or using a zero-order method such as ES. $\boldsymbol{C}$ can always be estimated through backward propagation, since $\boldsymbol{\theta}$ is a continuous vector function of $\boldsymbol{\phi}$. Gradient calculations in existing PSL methods is summarized in Table 4. Parameter $\boldsymbol{\phi}$ is iteratively updated by gradient descent: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta \frac{\partial \ell_{\mathrm{psl}}}{\partial \boldsymbol{\phi}}$.

3.4 Multiobjective Bayesian optimization solvers

When the evaluation of objective functions is costly, multiobjective Bayesian optimization (MOBO) is often the preferred methodology for tackling such challenges. While there are several existing libraries, such as BoTorch [49] and HEBO [50], that facilitate MOBO, they largely overlook algorithms that leverage decomposition techniques like PSL. To bridge this gap, `LibMOON` also includes three decomposition-based MOBO algorithms, including PSL-MOBO [23], PSL-DirHVEI [23, 51], and DirHV-EGO [51]. In each iteration, these methods build Gaussian process (GP) models for each objectives and generate a batch of query points for true function evaluations.

PSL-MOBO is an extension of PSL method for expensive MOPs. It optimizes the preference-conditional acquisition function (AF) $\alpha(\boldsymbol{\theta}|\boldsymbol{\lambda})$ over an infinite number of preference vectors to generate a set of promising candidates: $\min_{\boldsymbol{\phi}} \ell_{\mathrm{psl}} = \mathbb{E}_{\boldsymbol{\lambda}\sim\mathrm{Dir}(\boldsymbol{p})}\left[\alpha(\boldsymbol{\theta}_{\boldsymbol{\phi}}(\boldsymbol{\lambda})|\boldsymbol{\lambda})\right], \boldsymbol{\theta}_{\boldsymbol{\phi}}(\boldsymbol{\lambda}) : \boldsymbol{\Delta}_m \mapsto \mathbb{R}^n$.

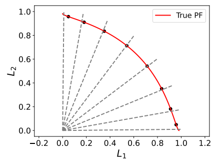
Table 5: **Supported preference-conditional AFs.**

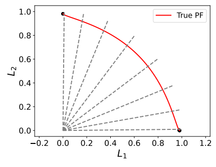
	preference-conditional AFs
TLCB	$\alpha_{\mathrm{TLCB}}(\boldsymbol{\theta}|\boldsymbol{\lambda}) = \max_{i\in[m]}\{\lambda_i(\hat{\mu}_i(\boldsymbol{\theta}) - \beta\hat{\sigma}_i(\boldsymbol{\theta}) - z_i^*)\}$
DirHV-EI	$\alpha_{\mathrm{DirHVEI}}(\boldsymbol{\theta}|\boldsymbol{\lambda}) = \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y}|\boldsymbol{\theta},\mathcal{D})}\left[\prod_{i=1}^{m}[\xi_i - y_i]_+\right]$

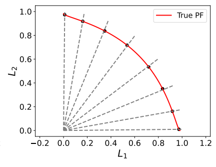
Currently, our library supports two representative preference-conditional AFs: the Tchebycheff scalarization of lower confidence bound (TLCB) [52, 23, 53] and the expected direction-based hypervolume improvement (DirHV-EI) [51]. We note that DirHV-EI can be regarded as an unbiased estimation of a weighted expected hypervolume improvement. Our library also supports DirHV-EGO, which employs a finite set of predetermined reference vectors as in [30].

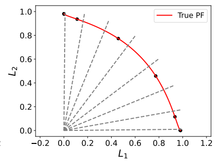
4 Empirical studies

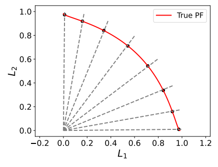
In this section, we present the empirical results of `LibMOON`. The experiments were conducted on a personal computer with an i7-10700 CPU and a 10GB RTX3080 GPU. GPU acceleration analysis was performed using RTX3080, 4060, and 4090. The results cover six areas: MOO solvers for synthetic problems, Pareto set learning solvers for synthetic problems, MOO solvers for MTL problems, Pareto set learning solvers for MTL problems, MOBO for synthetic problems, and GPU acceleration (in Appendix A.4).

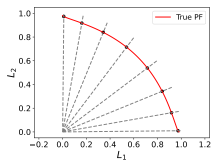

(a) Agg-COSMOS

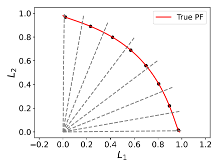

(b) Agg-LS

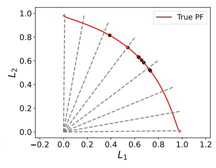

(c) Agg-PBI

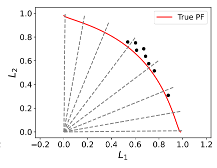

(d) Agg-SmoothTche


(e) Agg-Tche


(f) EPO


(g) HVGrad


(h) MGDA-UB


(i) MOO-SVGD

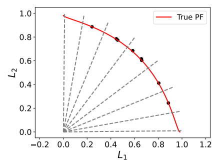

(j) PMTL

Figure 3: Finite Pareto solutions by ten MOO solvers on VLMOP2 problem.

4.1 MOO solvers for synthetic problems

We report the performance of various finite Pareto solution solvers using the VLMOP2 problem, where $f_1(\boldsymbol{\theta}) = 1 - \exp(-\sum_{i=1}^{n}(\theta_i - \frac{1}{\sqrt{n}})^2)$, $f_2(\boldsymbol{\theta}) = 1 - \exp(-\sum_{i=1}^{n}(\theta_i + \frac{1}{\sqrt{n}})^2)$. VLMOP2 has been widely studied in the literature [16, 13] since its PF is non-convex [5]. Some methods (e.g., Agg-LS) fail immediately on this problem since its PF is non-convex. Visualization results are shown by Figure 3 and the numerical results are shown by Table 6. We present the following key findings:

Table 6: Numerical results of finite Pareto solutions for the VLMOP2 problem.

Method	$l_{\min}$	$sl_{\min}$	Spacing	Sparsity	HV	IP	Cross Angle	PBI
EPO	0.162 (0.000)	0.061 (0.000)	0.029 (0.001)	0.043 (0.000)	0.283 (0.000)	0.776 (0.000)	**0.046 (0.041)**	**0.930 (0.003)**
MGDA-UB	0.012 (0.013)	-0.098 (0.011)	0.036 (0.011)	**0.006 (0.001)**	0.228 (0.008)	0.606 (0.010)	31.278 (1.533)	2.986 (0.088)
PMGDA	0.150 (0.001)	0.055 (0.000)	0.034 (0.001)	0.042 (0.000)	0.283 (0.000)	0.775 (0.000)	0.318 (0.037)	0.952 (0.003)
Random	0.000 (0.000)	-0.161 (0.002)	**0.000 (0.000)**	0.272 (0.000)	0.044 (0.000)	0.410 (0.127)	52.290 (12.938)	3.894 (0.590)
MOO-SVGD	0.060 (0.002)	-0.077 (0.004)	0.033 (0.018)	0.009 (0.003)	0.212 (0.003)	0.633 (0.024)	29.647 (3.305)	2.963 (0.197)
PMTL	0.014 (0.010)	-0.068 (0.009)	0.061 (0.020)	0.018 (0.007)	0.260 (0.012)	0.706 (0.004)	15.036 (1.270)	1.993 (0.093)
HVGrad	**0.182 (0.000)**	**0.067 (0.000)**	0.016 (0.000)	0.041 (0.000)	**0.286 (0.000)**	0.578 (0.069)	34.090 (8.607)	3.062 (0.465)
Agg-LS	0.000 (0.000)	-0.159 (0.001)	0.002 (0.001)	0.272 (0.001)	0.043 (0.002)	**0.227 (0.008)**	71.168 (0.958)	4.764 (0.047)
Agg-Tche	0.158 (0.001)	0.061 (0.000)	0.031 (0.001)	0.043 (0.000)	0.283 (0.000)	0.348 (0.000)	55.174 (0.049)	3.889 (0.004)
Agg-PBI	0.113 (0.074)	0.032 (0.046)	0.045 (0.030)	0.042 (0.002)	0.281 (0.002)	0.657 (0.097)	11.374 (9.125)	1.434 (0.402)
Agg-COSMOS	0.141 (0.000)	0.045 (0.000)	0.035 (0.000)	0.039 (0.000)	0.285 (0.000)	0.771 (0.000)	1.085 (0.000)	1.011 (0.000)
Agg-SmoothTche	0.004 (0.000)	-0.074 (0.000)	0.154 (0.001)	0.074 (0.000)	0.244 (0.000)	0.276 (0.000)	63.106 (0.018)	4.253 (0.001)

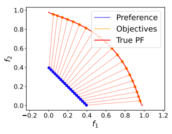
All methods were run five times with random seeds; results are presented in the format of (mean)(std).

①. Agg-COSMOS produces rough "exact" Pareto solutions due to a cosine similarity term encouraging Pareto objectives to be close to preference vectors. However, the position of Pareto objectives can not be determined. Agg-LS can only find two endpoints of the PF. ②. Agg-PBI generates "exact" Pareto solutions when the coefficient of d_2 exceeds a specific value [54]. However, this parameter is challenging to tune, which is influenced by the curvature of a PF. Additionally, PBI may transform a convex multi-objective optimization problem into a non-convex one, making Agg-PBI less recommended. ③. Agg-Tche generates diverse solutions and produces exact Pareto solutions corresponding to the element-wise inverse of the preference vector. Both Agg-Tche and Agg-SmoothTche retain convexity - their aggregation functions keep convex when all objectives are convex. ④. HVGrad updates the decision variable using the hypervolume gradient, resulting in the largest hypervolume. PMTL is a two-stage method. In the first stage, solutions are updated to specific regions and in the second stage, solutions are updated to Pareto solutions constrained in these specific regions. MOO-SVGD's update direction has two conflicting goals: promoting diversity and ensuring convergence. This conflict makes MOO-SVGD less stable and take more iterations to converge. ⑤. Among these methods, MGDA-UB, Random, Agg-PBI, and MOO-SVGD exhibit relatively large standard deviations. MGDA-UB and Random generate arbitrary Pareto solutions due to their computation nature. Agg-PBI results in non-convex aggregation functions, leading to variable solutions based on different initializations.

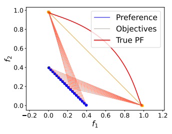
[5]In MOO, a PF is convex or non-convex based on whether the objective space is convex or non-convex. The PF represents the non-dominated boundary of the objective space.

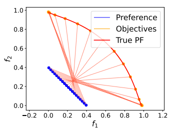
Table 7: Pareto set learning results on VLMOP2 problem.

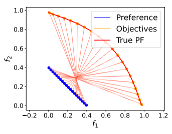
Method	$l_{\min}$	$sl_{\min}$	Spacing	Sparsity	HV	IP	Cross Angle	PBI	Span
COSMOS	0.045 (0.000)	-0.127 (0.000)	1.560 (0.004)	**0.525 (0.000)**	0.318 (0.000)	0.752 (0.000)	0.950 (0.001)	0.995 (0.000)	0.907 (0.000)
Agg-LS	0.000 (0.000)	-0.258 (0.000)	**0.115 (0.094)**	13.245 (1.871)	0.045 (0.003)	**0.239 (0.001)**	70.541 (0.156)	4.811 (0.007)	**0.982 (0.000)**
Agg-Tche	0.047 (0.004)	-0.121 (0.000)	1.476 (0.146)	0.579 (0.008)	0.319 (0.000)	0.383 (0.003)	51.558 (0.280)	3.775 (0.011)	0.955 (0.005)
SmoothTche	0.000 (0.000)	-0.187 (0.000)	6.711 (0.003)	1.060 (0.000)	0.302 (0.000)	0.300 (0.000)	60.386 (0.001)	4.169 (0.000)	0.982 (0.000)
EPO	**0.050 (0.001)**	**-0.120 (0.000)**	1.332 (0.078)	0.583 (0.005)	0.319 (0.000)	0.756 (0.000)	0.388 (0.098)	0.952 (0.008)	0.961 (0.003)
PMGDA	0.047 (0.000)	-0.121 (0.000)	1.446 (0.051)	0.580 (0.002)	**0.319 (0.000)**	0.756 (0.000)	**0.215 (0.062)**	**0.939 (0.005)**	0.958 (0.001)

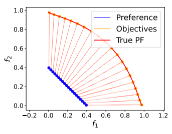


(a) COSMOS


(b) Agg-LS


(c) SmoothTche


(d) Tche


(e) EPO

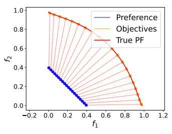

(f) PMGDA

Figure 4: Predicted Pareto solutions by different PSL solvers on VLMOP2 problem.

Conclusion. For synthetic problems, the most recommended method is *Agg-Tche* since (1) it keeps function convexity, (2) it finds "exact" Pareto solutions under quite mild conditions (3) it does not need to calculate the Jacobian matrix for each iteration.

4.2 Pareto set learning on synthetic problems

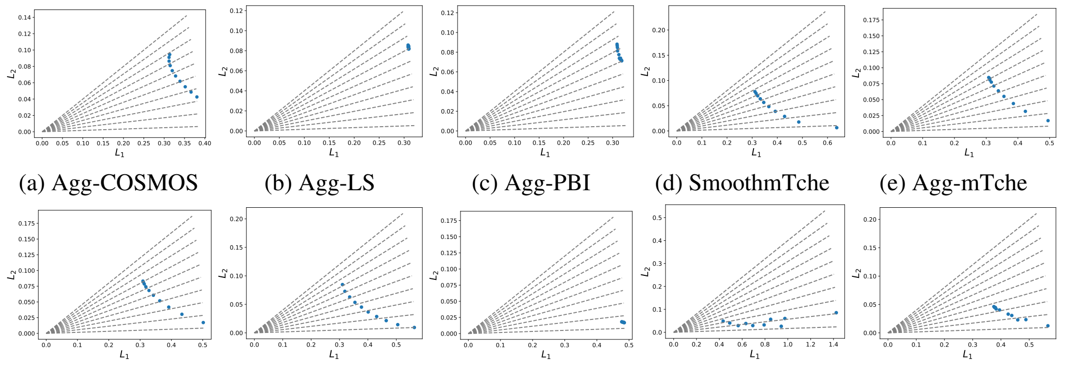
In this section, we present PSL results (Figure 4 and table 7) also on VLMOP2. ①. PSL with the COSMOS aggregation function fails to find all marginal Pareto solutions because COSMOS does not guarantee the discovery of the entire PS/PF. PSL with linear scalarization function could not fit the two endpoints of the PF. Those PSL results inherit from their base MOO solvers. ②. PSL with the smooth Tchebycheff function finds diverse but non-exact Pareto solutions. In contrast, PSL with Agg-Tche, EPO, and PMGDA as base solvers discovers the entire PS/PF, as all three methods find exact Pareto solutions. By traversing the preference simplex, the model accurately fits the entire PS.

Conclusion. The most recommended method is still *Agg-Tche-based PSL* since its basic MOO solver Agg-Tche has attractive properties as mentioned in the previous section.

4.3 MOO solvers for MTL problems

We evaluate the performance of finite Pareto solvers on the Adult dataset, a multitask fairness classification problem. The decision variable $\boldsymbol{\theta}$ represents the parameters of a fully-connected neural network with $|\boldsymbol{\theta}| = 28033$. The first objective is cross-entropy loss, and the second is the DEO loss [34][Eq. 6]. ①. Agg-LS has two drawbacks: (1) it cannot identify the non-convex part of a PF (as previous section mentioned), and (2) the relationship between preference vectors and Pareto objectives is unknown; different preference vectors may yield duplicate solutions. Agg-PBI and Agg-COSMOS only find a small portion of the PF. ②. Agg-Smooth mTche and Agg-mTche perform well on this task, as they can find (approximate) "exact" Pareto solutions. Once the range of PF range is known, diverse solutions can be easily found using uniform preference vectors. The Random and MGDA-UB methods only find a single Pareto solution, since the position of this solution cannot be controlled by these methods. ③. Among the three methods that directly find a set of Pareto solutions (MOO-SVGD, PMTL, and HV-Grad), HV-Grad produces the most diverse solutions with the largest hypervolume. PMTL, being a two-stage method, may fail when solutions fall outside the sector due to stochastic factors. MOO-SVGD optimizes both convergence and diversity but is generally unstable based on our tests.

Conclusion. For convex Pareto fronts in MTL problems, *Agg-LS* is recommended for computational efficiency. However, *PMGDA* or *EPO* may offer better convergence and preference-solution correspondence.


(a) Agg-COSMOS (b) Agg-LS (c) Agg-PBI (d) SmoothmTche (e) Agg-mTche

(f) EPO (g) HVGrad (h) MGDA-UB (i) MOO-SVGD (j) PMTL

Figure 5: Finite Pareto solutions by different solvers on Adult problem.

Table 8: Numerical results of finite Pareto solutions for the Adult problem.

Method	$l_{\min}$	$sl_{\min}$	Spacing	Sparsity	HV	IP	Cross Angle	PBI	Span
Agg-COSMOS	0.004 (0.000)	-0.194 (0.000)	0.463 (0.004)	0.014 (0.001)	0.657 (0.000)	0.295 (0.000)	2.787 (0.013)	0.426 (0.000)	0.052 (0.000)
Agg-LS	0.000 (0.000)	-0.223 (0.000)	**0.016 (0.005)**	0.000 (0.000)	0.636 (0.001)	**0.272 (0.001)**	6.595 (0.028)	0.500 (0.001)	0.002 (0.000)
Agg-PBI	0.001 (0.000)	-0.216 (0.000)	0.107 (0.008)	0.001 (0.000)	0.642 (0.000)	0.277 (0.001)	4.995 (0.007)	0.462 (0.001)	0.010 (0.000)
Agg-SmoothmTche	0.005 (0.001)	-0.163 (0.000)	4.237 (0.029)	0.329 (0.003)	0.675 (0.001)	0.347 (0.001)	3.385 (0.053)	0.500 (0.002)	0.072 (0.000)
Agg-mTche	0.002 (0.000)	-0.177 (0.000)	2.034 (0.072)	0.091 (0.004)	0.674 (0.000)	0.316 (0.000)	**1.962 (0.045)**	**0.422 (0.001)**	0.067 (0.001)
EPO	0.002 (0.000)	-0.175 (0.000)	2.136 (0.054)	0.101 (0.004)	0.674 (0.001)	0.320 (0.000)	2.002 (0.025)	0.426 (0.001)	0.066 (0.000)
MGDA-UB	0.000 (0.000)	-0.222 (0.001)	0.050 (0.039)	0.000 (0.000)	0.510 (0.003)	0.410 (0.004)	9.586 (0.072)	0.878 (0.011)	0.001 (0.000)
Random	0.000 (0.000)	-0.224 (0.000)	0.040 (0.016)	**0.000 (0.000)**	0.633 (0.001)	0.279 (0.000)	5.863 (0.010)	0.491 (0.000)	0.002 (0.000)
PMTL	0.002 (0.001)	-0.176 (0.002)	2.236 (0.045)	0.156 (0.028)	0.617 (0.002)	0.372 (0.004)	7.039 (0.242)	0.675 (0.016)	0.035 (0.001)
MOO-SVGD	**0.049 (0.007)**	**-0.079 (0.003)**	5.382 (4.539)	2.660 (1.857)	0.548 (0.005)	0.657 (0.013)	8.354 (0.211)	1.274 (0.029)	0.065 (0.010)
HVGrad	0.014 (0.002)	-0.153 (0.002)	1.347 (0.041)	0.110 (0.003)	**0.678 (0.001)**	0.347 (0.004)	7.043 (0.260)	0.663 (0.017)	**0.075 (0.001)**

All methods were run five times with random seeds; results are presented in the form of (mean)(std).

4.4 Pareto set learning on MTL

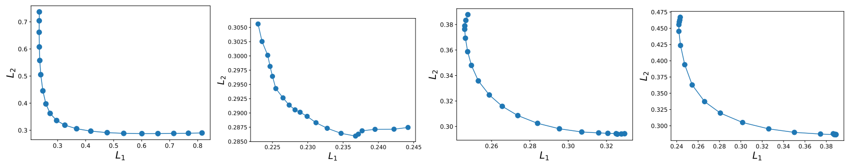
This section presents the Pareto set learning results on the MO-MNIST problem using a hypernet architecture. The Pareto set model was trained for 20 epochs, optimizing approximately 3.24M parameters, with the first and second objectives being the cross-entropy losses for the top-right and bottom-left images. EPO-based or PMGDA-based PSL is not very suitable for this task since manipulating gradient of 3.2M parameters is not efficient. Empirically, for regression tasks, the PF shape is nearly convex. Therefore, Agg-LS is adequate to recover the whole PF. From Figure 6 and table 9, we have that Agg-LS significantly outperforms other methods, evidenced by the HV of Agg-LS is much larger than other methods. Furthermore, the training losses across all methods are almost stable after 40 epochs. From the figure, to further reduce the training loss, it needs much more computational resources.
Conclusion. For MTL problems with a convex PF, it is highly recommended to use *Agg-LS-based* PSL for the sake of computational efficiency.

4.5 MOBO for synthetic and real-world problems

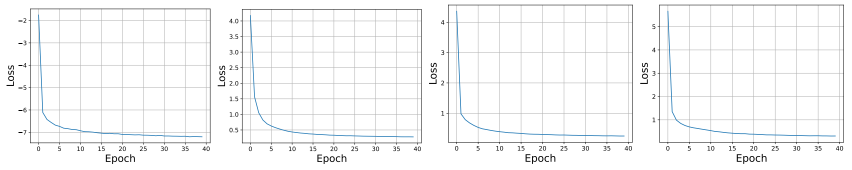
In this section, we test three MOBO algorithms in `LibMOON` on three benchmark problems, including ZDT1, RE21, VLMOP1 and VLMOP2. To ensure a fair comparison, we generate $11n - 1$ initial samples using Latin Hypercube Sampling for each method. The maximum number of function

Table 9: Pareto set learning results on MO-MNITS problem.

Method	$l_{\min}$	$sl_{\min}$	Spacing	Sparsity	HV	IP	Cross Angle	PBI	Span
COSMOS	**0.033 (0.002)**	**-0.152 (0.007)**	0.967 (0.202)	0.278 (0.038)	0.512 (0.015)	0.535 (0.028)	**9.208 (0.541)**	1.221 (0.050)	**0.497 (0.043)**
Agg-LS	0.001 (0.000)	-0.286 (0.002)	0.068 (0.029)	0.000 (0.000)	**0.557 (0.004)**	**0.257 (0.004)**	27.536 (0.116)	**1.149 (0.021)**	0.016 (0.003)
Agg-PBI	0.000 (0.000)	-0.295 (0.000)	**0.019 (0.006)**	**0.000 (0.000)**	0.536 (0.008)	0.269 (0.005)	26.270 (0.117)	1.150 (0.019)	0.002 (0.001)
SmoothTche	0.001 (0.000)	-0.248 (0.002)	0.440 (0.040)	0.012 (0.001)	0.538 (0.008)	0.281 (0.005)	32.244 (0.417)	1.471 (0.007)	0.087 (0.012)
Agg-Tche	0.000 (0.000)	-0.235 (0.004)	0.988 (0.110)	0.045 (0.015)	0.533 (0.010)	0.292 (0.008)	35.292 (0.742)	1.693 (0.075)	0.131 (0.014)

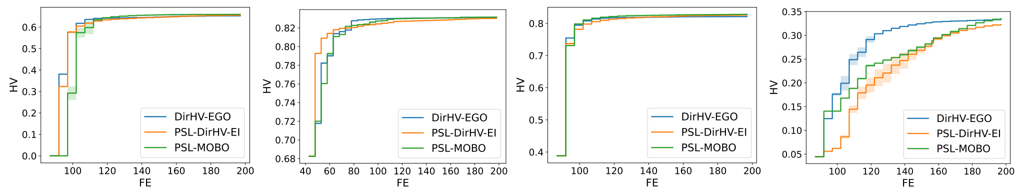


(a) Agg-COSMOS (b) Agg-LS (c) Agg-SmoothTche (d) Agg-Tche



(e) Agg-COSMOS (f) Agg-LS (g) Agg-SmoothTche (h) Agg-Tche

Figure 6: Training process for generating predicted Pareto solutions using different PSL solvers on MO-MNIST problem.



(a) ZDT1 (b) RE21 (c) VLMOP1 (d) VLMOP2

Figure 7: HV curves on MOBO problem. Results are averaged on five random seeds. Reference point to calculate HV : [1.2, 1.2].

evaluations is set as 200. Our experimental results, illustrated in Figure 4, clearly demonstrate the rapid convergence capabilities of all three methods. DirHV-EGO, PSL-DirHV-EI, and PSL-MOBO not only efficiently navigate the solution space but also quickly reach optimal solutions. This highlights the robustness and effectiveness of our implemented algorithms in handling different types of MOPs.

5 Conclusion, limitations, and further works

Conclusion. We introduced the *first* modern gradient-based MOO framework called `LibMOON` in PyTorch for the research community's convenience. `LibMOON` supports more than 20 mainstream gradient-based MOO methods; the modular design of `LibMOON` further allows the library to address various MOPs via various methods in a plug-and-play manner. `LibMOON` can thus be leveraged to quickly yet robustly test new MOO ideas.

Limitations include: (1) rapid developments of gradient-based MOO methods makes it hard to incorporate all methods, so some effective methods may be missing; (2) gradient-based solvers may fail for problems with a number of local optimas.

Future Work includes (1) maintaining a user and development community to address issues and (2) adding newly published methods as quickly as possible.

Acknowledge

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China [GRF Project No. CityU 11215622].

A Appendix

A.1 Full name and notation tables

This section lists the full names of optimization methods and terms for clarity (Table 10) and provides notation in Table 11.

A.2 Aggregation functions

Aggregation function convert an MOP into a single-objective optimization problem under a specific preference vector $\boldsymbol{\lambda}$. Some popular aggregation functions are:

1. **COSMOS**:
$$g_{\boldsymbol{\lambda}}^{\text{cosmos}}(\boldsymbol{\theta}) = \boldsymbol{\lambda}^\top \boldsymbol{L}(\boldsymbol{\theta}) - \mu \frac{\boldsymbol{\lambda}^\top \boldsymbol{L}(\boldsymbol{\theta})}{\|\boldsymbol{\lambda}\| \|\boldsymbol{L}(\boldsymbol{\theta})\|}, \tag{3}$$
where μ is a positive weight factor to align the objective vector $\boldsymbol{L}(\boldsymbol{\theta})$ with the preference vector $\boldsymbol{\lambda}$.

2. **Linear scalarization (LS)**:
$$g_{\boldsymbol{\lambda}}^{\text{LS}}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \lambda_i L_i(\boldsymbol{\theta}). \tag{4}$$

3. **Tchebycheff (Tche)**:
$$g_{\boldsymbol{\lambda}}^{\text{Tche}}(\boldsymbol{\theta}) = \max_{1 \leq i \leq m} \left\{ \lambda_i (L_i(\boldsymbol{\theta}) - z_i) \right\}, \tag{5}$$
where $\boldsymbol{z}$ is a reference point, usually set as the nadir point the minimal value in each objective. Modified Tchebycheff is the same as the original one by simply choosing λ'_i to be $1/\lambda_i$, $g_{\boldsymbol{\lambda}}^{\text{mTche}}(\boldsymbol{\theta}) = g_{\boldsymbol{\lambda}'}^{\text{Tche}}(\boldsymbol{\theta})$.

4. **Smooth Tchebycheff (STche)**:
$$g_{\boldsymbol{\lambda}}^{\text{STche}}(\boldsymbol{\theta}) = \frac{1}{h} \log \left(\sum_{i=1}^{m} \exp(h \cdot \lambda_i (L_i(\boldsymbol{\theta}) - z_i)) \right). \tag{6}$$
The Smooth Tchebycheff function uses a relaxed Smooth $\max$ operator. The advantage of this approach is that $g_{\boldsymbol{\lambda}}^{\text{STche}}(\boldsymbol{\theta})$ becomes a smooth function if each objective function $L_i(\boldsymbol{\theta})$ is smooth, unlike the non-Smooth $g_{\boldsymbol{\lambda}}^{\text{Tche}}(\boldsymbol{\theta})$. Smooth functions generally have faster convergence rate compared to non-Smooth ones. Similarly, we can define the Smooth modified Tchebycheff function.

5. **Penalty-Based Intersection (PBI):**
$$g_{\boldsymbol{\lambda}}^{\text{PBI}}(\boldsymbol{\theta}) = \underbrace{\frac{1}{\|\boldsymbol{\lambda}\|} \cdot \sum_{i=1}^{m} \lambda_i L_i(\boldsymbol{\theta})}_{d_1} + \mu \underbrace{\left\| \boldsymbol{L}(\boldsymbol{\theta}) - \frac{d_1}{\|\boldsymbol{\lambda}\|} \cdot \boldsymbol{\lambda} \right\|}_{d_2}, \tag{7}$$
where μ is a positive weight factor that encourage a objective to align with a preference vector $\boldsymbol{\lambda}$.

6. p**-norm**:
$$g_{\boldsymbol{\lambda}}^{\text{pnorm}}(\boldsymbol{\theta}) = \|\boldsymbol{\lambda} \odot \boldsymbol{L}(\boldsymbol{\theta}) - \boldsymbol{z}\|_p. \tag{8}$$
The symbol $\odot$ denotes the element-wise product between two vectors.

Table 10: Short name to full name table

Short Name	Full name
MOP	Multiobjective Optimization Problem
SOP	Singleobjective Optimization Problem
MOO	MultiObjective Optimization
MOEA	MultiObjective Evolutionary Algorithm
MOBO	MultiObjective Baysian Optimization
PSL	Pareto Set Learning
PS	Pareto Set
PF	Pareto Front
"exact" Pareto solution	The corresponding Pareto objective aligns with a given preference vector
ES	Evolutionary strategy
BP	Backward propagation
PMTL [13]	Pareto Multa-Task Learning
MOO-SVGD [17]	MultiObjective Optimization Stein Variational Gradient Descent
EPO [16]	Exact Pareto Optimization
PMGDA [40]	Preference based Multiple Gradient Descent Algorithm
Agg-LS [37]	Aggregation function with Linear Scalarization
Agg-PBI [30]	Aggregation function with Penalty Based Intersection
Agg-Tche [30]	Aggregation function with Tchebycheff scalarization
Agg-mTche [42]	Aggregation function with modified Tchebycheff scalarization
Agg-COSMOS [34]	Aggregation function with COSMOS scalarization
RE problems	Realworld problems

Table 11: Notations used in this paper

Notation	Meaning
$\boldsymbol{\theta}$	The decision variable of an MOP.
$\boldsymbol{\phi}$	The decision variable of a Pareto set model.
m	Number of objectives.
n	Number of decision variables.
K	Number of finite Pareto solutions.
α_i	Coefficients of objective functions.
$\boldsymbol{\lambda}$	A preference vector.

7. **Augmented Achievement Scalarization Function (AASF)**:

$$g_{\boldsymbol{\lambda}}^{\text{AASF}}(\boldsymbol{\theta}) = g_{\boldsymbol{\lambda}}^{\text{mTche}}(\boldsymbol{\theta}) + \mu g_{\boldsymbol{\lambda}}^{\text{LS}}(\boldsymbol{\theta}), \tag{9}$$

where μ is small positive coefficient, which is usually set as 0.1. Contour curves for this function for a bi-objective case can be found in the LibMOON Doc[6].

A.3 Metrics

Metrics used in `LibMOON` can be categorized into two groups. The first group evaluates the quality of a *set* of solutions $\mathbb{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(K)}\}$, with specific metrics such as IGD and FD relying on the known Pareto front for accuracy. The second group of metrics assesses the quality of *individual* solutions $\boldsymbol{y}$ when a preference vector $\boldsymbol{\lambda}$ is provided.

Group 1: Metrics for a set of solutions.

1. Hypervolume (HV) (↑) [55]: This metric evaluates both the convergence to the PF and the diversity of solutions. A low HV value indicates poor convergence, while high HV values

[6] https://libmoondocs.readthedocs.io/en/latest/gallery/aggfuns.html

imply better performance. The hypervolume is calculated as the volume dominated by at least one solution in the set $\mathbb{S}$ with respect to a reference point $\boldsymbol{r}$:

$$\mathrm{HV}_{\boldsymbol{r}}(\mathbb{S}) = \mathrm{Vol}(\boldsymbol{y} \mid \exists \boldsymbol{y}' \in \mathbb{S}, \boldsymbol{y}' \preceq \boldsymbol{y} \preceq \boldsymbol{r}).$$

2. Inverted Generational Distance (IGD) [56]: IGD measures the average distance between points in a reference set $\mathbb{Z}$ and the nearest solutions in the set $\mathbb{S}$:

$$\mathrm{IGD}(\mathbb{S}) = \frac{1}{|\mathbb{Z}|} \left(\sum_{i=1}^{|\mathbb{Z}|} \min_{\boldsymbol{y}' \in \mathbb{S}} \rho(\boldsymbol{y}^{(i)}, \boldsymbol{y}')^2 \right)^{1/2}.$$

3. Fill Distance (FD) [57]: This metric calculates the covering radius of a set of solutions $\mathbb{S}$, defined as the maximum minimum distance from any point in the reference set $\mathbb{Z}$ to the nearest solution in $\mathbb{Z}$:

$$\mathrm{FD}(\mathbb{S}) = \max_{\boldsymbol{y}' \in \mathbb{Z}} \min_{\boldsymbol{y} \in \mathbb{S}} \rho(\boldsymbol{y}, \boldsymbol{y}'). \tag{10}$$

4. Minimal Distance (l_{min}): This metric captures the smallest pairwise distance among all objectives:

$$l_{\text{min}} = \min_{1 \le i < j \le K} \rho(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$$

where $\rho()$ denotes the Euclidean distance.

5. Smooth Minimal Distance (sl_{min}): This metric is a “smooth-min" version of the minimal distance function, defined as:

$$sl_{\text{min}} = -\frac{1}{h \cdot K(K-1)} \log \left(\sum_{1 \le i < j \le K} \exp \left(-h\rho \left(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)} \right) \right) \right). \tag{11}$$

6. Spacing: This metric measures the standard deviation of the minimal distances from one solution to others, with lower values indicating a more uniform distribution of objective vectors:

$$\text{spacing} = \frac{1}{K} \sum_{i=1}^{K} (d_i - \bar{d})^2, \qquad \bar{d} = \frac{1}{K} \sum_{i=1}^{K} d_i, \qquad d_i = \min_{1 \le i \ne j \le K} \rho(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}). \tag{12}$$

7. Span: This metric evaluates the range (span) of solutions in their minimal dimension, defined as:

$$\mathrm{Span} = \min_{1 \le i \le m} \max_{1 \le k < l \le K} |y_i^{(k)} - y_i^{(l)}|. \tag{13}$$

Group 2: Metrics for individual solutions.

1. Penalty-based Intersection (PBI): This metric is a weighted sum of two distance functions d_1 and d_2, given by $\mathrm{PBI} = d_1 + \mu d_2$, where

$$d_1 = \frac{\langle \boldsymbol{y} - \boldsymbol{z}, \boldsymbol{\lambda} \rangle}{\|\boldsymbol{\lambda}\|}, \qquad d_2 = \|\boldsymbol{y} - (d_1 \boldsymbol{\lambda} + \boldsymbol{z})\|. \tag{14}$$

2. Inner Product (IP): This metric measures the alignment of the objective vector $\boldsymbol{y}$ with the preference vector $\boldsymbol{\lambda}$:

$$\mathrm{IP} = \langle \boldsymbol{y}, \boldsymbol{\lambda} \rangle. \tag{15}$$

3. Cross Angle (ϑ): For bi-objective problems, this metric measures the angular difference between the objective vector and the preference vector:

$$\vartheta = \|\arctan(y_2/y_1) - \arctan(\lambda_2/\lambda_1)\|. \tag{16}$$

Those metrics are summarized in Table 12 and also can be found in the `LibMOON` document.

Table 12: Supported metrics.

Metrics	Full name	Descriptions
HV	Hypervolume	Hypervolume value of the dominated region.
IGD	Inverted general distance	The average general distance between a set and the true PF.
FD	Fill distance	The radius of a set to cover the true PF.
$\mathrm{l}_{\min}$	Minimal distance	The minimal pairwise distance of a set.
$\mathrm{sl}_{\min}$	Smooth minimal distance	The smooth minimal pairwise distance of a set.
Spacing	-	The standard deviation of the minimal distances to other solutions.
Span	-	The range of a set.
PBI	Penalty-based intersection	The weighted sum distance between IP and distance to a preference vector.
IP	Inner product	The inner product between a preference and an objective vector.
ϑ	Cross product	The cross angle between a preference and an objective vector.

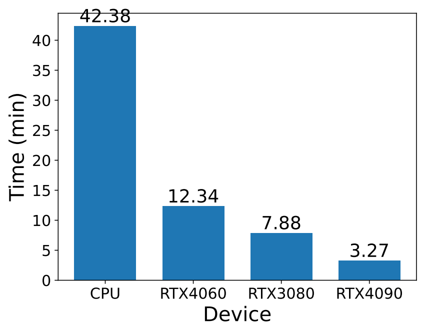


Figure 8: Running time for Pareto set learning on the MO-MNIST problem using different devices using 3M parameters.

A.4 GPU acceleration

We evaluate `LibMOON` performance on Pareto set learning for the MO-MNIST problem across various platforms (CPU, RTX 3080, 4060, 4090). Running times are detailed in Table 13 and visualized in Figure 8. The table and figure show a significant reduction in time (about one-third) when using a personal GPU compared to a CPU. The RTX 4090 further reduces time by approximately 25% compared to the RTX 4060.

A.5 License, usage, and code dependence

The license used for Adult/Compas/Credit follows Creative Commons Attribution 4.0 International (CC BY 4.0), Database Contents License (DbCL) v1.0, and CC0: Public Domain, respectively. For academic use of `LibMOON`, please cite our paper or GitHub. Commercial use requires author permission.

Table 13: Running time (minutes) for different platforms on MO-MNIST problem.

Platform	**MO-MNIST**	**MO-Fashion**	**Fashion-MNIST**
CPU	43.43	44.45	46.45
RTX 4060 (8G)	14.72	13.21	12.43
RTX 3080 (10G)	7.88	7.16	7.17
RTX 4090 (24G)	3.27	3.27	3.27

We run all datasets for 100 epochs using 3M parameters on a 13th Gen Intel(R) Core(TM) i9-13900HX CPU.

Some part codes of `LibMOON` follows (1) COSMOS [7], (2) PHN [8], and (3) HVGrad [9].

[7] https://github.com/ruchtem/cosmos

[8] https://github.com/AvivNavon/pareto-hypernetworks

[9] https://github.com/timodeist/multi_objective_learning

References

[1] Han Zhao and Geoffrey J Gordon. Inherent tradeoffs in learning fair representations. *The Journal of Machine Learning Research*, 23(1):2527–2552, 2022.

[2] Ruicheng Xian, Lang Yin, and Han Zhao. Fair and optimal classification via post-processing. In *International Conference on Machine Learning*, pages 37977–38012. PMLR, 2023.

[3] Martim Brandao, Maurice Fallon, and Ioannis Havoutis. Multi-controller multi-objective locomotion planning for legged robots. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4714–4721. IEEE, 2019.

[4] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International conference on machine learning*, pages 10607–10616. PMLR, 2020.

[5] Dietmar Jannach. Multi-objective recommender systems: Survey and challenges. *arXiv preprint arXiv:2210.10309*, 2022.

[6] Ee Yeo Keat, Nurfadhlina Mohd Sharef, Razali Yaakob, Khairul Azhar Kasmiran, Erzam Marlisah, Norwati Mustapha, and Maslina Zolkepli. Multiobjective deep reinforcement learning for recommendation systems. *IEEE Access*, 10:65011–65027, 2022.

[7] Fatima Ezzahra Zaizi, Sara Qassimi, and Said Rakrak. Multi-objective optimization with recommender systems: A systematic review. *Information Systems*, 117:102233, 2023.

[8] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[9] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017.

[10] Francesco Biscani and Dario Izzo. A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53):2338, 2020.

[11] Julian Blank, Kalyanmoy Deb, Yashesh Dhebar, Sunith Bandaru, and Haitham Seada. Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):48–60, 2020.

[12] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[13] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

[14] Sebastian Peitz and Michael Dellnitz. Gradient-based multiobjective optimization with uncertainties. In *NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico*, pages 159–182. Springer, 2018.

[15] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021.

[16] Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*, pages 6597–6607. PMLR, 2020.

[17] Xingchao Liu, Xin Tong, and Qiang Liu. Profiling pareto front with multi-objective stein variational gradient descent. *Advances in Neural Information Processing Systems*, 34:14721–14733, 2021.

[18] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*, 2020.

[19] Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. Controllable pareto multi-task learning. *arXiv preprint arXiv:2010.06313*, 2020.

[20] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. Panacea: Pareto alignment via preference adaptation for llms. *arXiv preprint arXiv:2402.02030*, 2024.

[22] Xi Lin, Xiaoyuan Zhang, Zhiyuan Yang, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Smooth tchebycheff scalarization for multi-objective optimization. In *International conference on machine learning*. PMLR, 2024.

[23] Xi Lin, Zhiyuan Yang, Xiaoyuan Zhang, and Qingfu Zhang. Pareto set learning for expensive multi-objective optimization. *Advances in Neural Information Processing Systems*, 35:19231–19247, 2022.

[24] Xiaoyuan Zhang, Xi Lin, Bo Xue, Yifan Chen, and Qingfu Zhang. Hypervolume maximization: A geometric view of pareto set learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[25] Baijiong Lin and Yu Zhang. Libmtl: A python library for deep multi-task learning. *The Journal of Machine Learning Research*, 24(1):9999–10005, 2023.

[26] Juan J. Durillo, Antonio J. Nebro, and Enrique Alba. The jmetal framework for multi-objective optimization: Design and architecture. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[27] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[28] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[29] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2014.

[30] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[31] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[32] Nihat Engin Toklu, Timothy Atkinson, Vojtěch Micka, Paweł Liskowski, and Rupesh Kumar Srivastava. Evotorch: Scalable evolutionary computation in python. *arXiv preprint arXiv:2302.12600*, 2023.

[33] Beichen Huang, Ran Cheng, Zhuozhao Li, Yaochu Jin, and Kay Chen Tan. Evox: A distributed gpu-accelerated framework for scalable evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 2024.

[34] Michael Ruchte and Josif Grabocka. Scalable pareto front approximation for deep multi-objective learning. In *2021 IEEE international conference on data mining (ICDM)*, pages 1306–1311. IEEE, 2021.

[35] Yuzheng Hu, Ruicheng Xian, Qilong Wu, Qiuling Fan, Lang Yin, and Han Zhao. Revisiting scalarization in multi-task learning: A theoretical perspective. *arXiv preprint arXiv:2308.13985*, 2023.

[36] Kirtan Padh, Diego Antognini, Emma Lejal-Glaude, Boi Faltings, and Claudiu Musat. Addressing fairness in classification with a model-agnostic multi-objective algorithm. In *Uncertainty in artificial intelligence*, pages 600–609. PMLR, 2021.

[37] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

[38] Timo M Deist, Monika Grewal, Frank JWM Dankers, Tanja Alderliesten, and Peter AN Bosman. Multi-objective learning to predict pareto fronts using hypervolume maximization. *arXiv preprint arXiv:2102.04523*, 2021.

[39] Yifei He, Shiji Zhou, Guojun Zhang, Hyokun Yun, Yi Xu, Belinda Zeng, Trishul Chilimbi, and Han Zhao. Robust multi-task learning with excess risks. *International conference on machine learning*, 2024.

[40] Xiaoyuan Zhang, Xi Lin, and Qingfu Zhang. Pmgda: A preference-based multiple gradient descent algorithm. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024.

[41] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021.

[42] Xiaoliang Ma, Qingfu Zhang, Guangdong Tian, Junshan Yang, and Zexuan Zhu. On tchebycheff decomposition approaches for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 22(2):226–244, 2017.

[43] Michael Emmerich and André Deutz. Time complexity and zeros of the hypervolume indicator gradient field. In *EVOLVE-a bridge between probability, set oriented numerics, and evolutionary computation III*, pages 169–193. Springer, 2014.

[44] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1:3–52, 2002.

[45] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[46] Weiyu Chen and James T Kwok. Efficient pareto manifold learning with low-rank structure. *arXiv preprint arXiv:2407.20734*, 2024.

[47] Nikolaos Dimitriadis, Pascal Frossard, and Francois Fleuret. Pareto low-rank adapters: Efficient multi-task learning with preferences. *arXiv preprint arXiv:2407.08056*, 2024.

[48] Xi Lin, Xiaoyuan Zhang, Zhiyuan Yang, and Qingfu Zhang. Evolutionary pareto set learning with structure constraints. *arXiv preprint arXiv:2310.20426*, 2023.

[49] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[50] Alexander Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, Alexandre Maravel, Jianye Hao, Jun Wang, Jan Peters, and Haitham Bou Ammar. HEBO: Pushing the limits of sample-efficient hyperparameter optimisation. *Journal of Artificial Intelligence Research*, 74, 07 2022.

[51] Liang Zhao and Qingfu Zhang. Hypervolume-guided decomposition for parallel expensive multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 28(2):432–444, 2024.

[52] Xi Lin, Qingfu Zhang, and Sam Kwong. An efficient batch expensive multi-objective evolutionary algorithm based on decomposition. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1343–1349, 2017.

[53] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pages 766–776. PMLR, 2020.

[54] Zhenkun Wang, Jingda Deng, Qingfu Zhang, and Qite Yang. On the parameter setting of the penalty-based boundary intersection method in moea/d. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 413–423. Springer, 2021.

[55] Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. The hypervolume indicator: Problems and algorithms. *arXiv preprint arXiv:2005.00515*, 2020.

[56] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In *Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29–April 1, 2015. Proceedings, Part II 8*, pages 110–125. Springer, 2015.

[57] Xiaoyuan Zhang, Genghui Li, Xi Lin, Yichi Zhang, Yifan Chen, and Qingfu Zhang. Gliding over the pareto front with uniform designs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] .
   (b) Did you describe the limitations of your work? [Yes] , see Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] . `LibMOON` is a basic optimization library and we do not see direct societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [N/A] .
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] . The assumption is in the theorem itself, e.g., in Theorem 1.
   (b) Did you include complete proofs of all theoretical results? [No] . This theorem is a restatement of previous results. We have give both the thereom and its proof proper citations.
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] . These instruments are provided in the `LibMOON` Github page: `https://github.com/xzhang2523/libmoon`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . See source code of `LibMOON`.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] . We have reported standard derivation results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . See first paragraph of Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] . See Appendix A.5.
   (b) Did you mention the license of the assets? [Yes] . See Appendix A.5.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] . Code are provided in `https://github.com/xzhang2523/libmoon`.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] . `LibMOON` uses public data such as multiobjective classification and fairness classification.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] .
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] .
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]