PSMGD: Periodic Stochastic Multi-Gradient Descent for Fast Multi-Objective Optimization

Mingjing Xu[1], **Peizhong Ju**[2], **Jia Liu**[3], **Haibo Yang**[1]
[1] Dept. of Computing and Information Sciences Ph.D., Rochester Institute of Technology
[2] Dept. of Computer Science, University of Kentucky
[3] Dept. of Electrical and Computer Engineering, The Ohio State University
{mx6835, hbycis}@rit.edu, peizhong.ju@uky.edu, liu.1736@osu.edu

Abstract

Multi-objective optimization (MOO) lies at the core of many machine learning (ML) applications that involve multiple, potentially conflicting objectives. Despite the long history of MOO, recent years have witnessed a surge in interest within the ML community in the development of gradient manipulation algorithms for MOO, thanks to the availability of gradient information in many ML problems. However, existing gradient manipulation methods for MOO often suffer from long training times, primarily due to the need for computing dynamic weights by solving an additional optimization problem to determine a common descent direction that can decrease all objectives simultaneously. To address this challenge, we propose a new and efficient algorithm called Periodic Stochastic Multi-Gradient Descent (PSMGD) to accelerate MOO. PSMGD is motivated by the key observation that dynamic weights across objectives exhibit small changes under minor updates over short intervals during the optimization process. Consequently, our PSMGD algorithm is designed to periodically compute these dynamic weights and utilizes them repeatedly, thereby effectively reducing the computational overload. Theoretically, we prove that PSMGD can achieve state-of-the-art convergence rates for strongly-convex, general convex, and non-convex functions. Additionally, we introduce a new computational complexity measure, termed backpropagation complexity, and demonstrate that PSMGD could achieve an objective-independent backpropagation complexity. Through extensive experiments, we verify that PSMGD can provide comparable or superior performance to state-of-the-art MOO algorithms while significantly reducing training time.

Code — https://github.com/MarcuXu/PSMG
Extended version — https://arxiv.org/abs/2412.10961

1 Introduction

Just as the human world is full of diverse and potentially conflicting goals, many machine learning paradigms are also multi-objective, such as multi-objective reinforcement learning (Hayes et al. 2022), multi-task learning (Caruana 1997; Sener and Koltun 2018), and learning-to-rank (Mahapatra et al. 2023). At the core of these machine learning paradigms lies multi-objective optimization (MOO), which aims to optimize multiple potentially conflicting objectives simultaneously. Formally, MOO can be mathematically cast as:

$$\min_{\mathbf{x}\in\mathcal{D}} \mathbf{F}(\mathbf{x}) := [f_1(\mathbf{x}), \cdots, f_S(\mathbf{x})], \tag{1}$$

where $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^d$ is the model parameter, and $f_s : \mathbb{R}^d \to \mathbb{R}$, $s \in [S]$ is the objective function.

The most common approach for handling MOO is to optimize a weighted average of the multiple objectives, also known as linear scalarization (Kurin et al. 2022; Xin et al. 2022). This approach transforms MOO into a single-objective problem, for which there are various off-the-shelf methods available. The linear scalarization approach has a fast per-iteration runtime but suffers from slow convergence and poor performance due to potential conflicts among objectives. To address this problem, there has been a surge of interest in recent years in developing gradient manipulation algorithms (Kendall, Gal, and Cipolla 2018; Chen et al. 2018; Yu et al. 2020; Chen et al. 2020; Javaloy and Valera 2021; Liu and Vicente 2021; Chen et al. 2024; Xiao, Ban, and Ji 2024). The key idea is to calculate dynamic weights to avoid conflicts and find a direction $\mathbf{d}(\mathbf{x})$ that decreases all objectives simultaneously. This common descent vector $\mathbf{d}(\mathbf{x})$ is often determined by solving an additional optimization problem that involves all objective gradients. Updating the model with the common descent direction $\mathbf{d}$ ensures a decrease in every objective. While these approaches show improved performance, the extra optimization process requires a substantial amount of time. This is particularly evident when the number of objectives is large and the model parameters are high-dimensional, which could significantly prolong the MOO training process. Existing empirical evaluations consistently demonstrate that gradient manipulation algorithms typically necessitate much longer training times (Kurin et al. 2022; Liu et al. 2024) (also shown in Sec 4). To this end, a natural question arises: *Can we design efficient gradient manipulation algorithms for fast MOO with theoretical guarantees?*

In this paper, we present Periodic Stochastic Multi-Gradient Descent (PSMGD), a simple yet effective algorithm to accelerate the MOO. Our PSMGD is motivated by the key observation that dynamic weights across objectives exhibit small changes under minor updates over short intervals. Thus, it is designed to periodically compute these dynamic weights and utilizes them repeatedly, thereby effectively reducing the

computational overload and achieving a faster MOO training process. Our contributions are summarized as follows:

- Built upon the key observation, we introduce PSMGD, a new and efficient gradient manipulation method. By calculating dynamic weights infrequently, it significantly alleviates computational burden and reduces training time.
- We conduct a theoretical analysis of PSMGD for strongly convex, convex, and non-convex functions, demonstrating that PSMGD achieves state-of-the-art convergence rates comparable to existing MOO methods (Table 1). We also introduce a new computational complexity measure, backpropagation (BP) complexity, to quantify computational workload, and further show that PSMGD can achieve an objective-independent backpropagation complexity.
- Through comprehensive evaluation, PSMGD shows comparable or even superior performance compared to existing MOO methods, with significantly less training time.

2 Related Works

The MOO problem, as stated in Eq. 1, has a long history that dates back to the 1950s. MOO algorithms can be broadly categorized into two main groups. The first category comprises gradient-free methods, such as evolutionary MOO algorithms and Bayesian MOO algorithms (Zhang and Li 2007; Deb et al. 2002; Belakaria et al. 2020; Laumanns and Ocenasek 2002). These methods are more suitable for small-scale problems but are less practical for high-dimensional models, such as deep neural networks. The second category is the gradient-based approach by utilizing (stochastic) gradients (Fliege and Svaiter 2000; Désidéri 2012; Fliege, Vaz, and Vicente 2019; Liu and Vicente 2021), making them more practical for high-dimensional MOO problems. In this work, we primarily focus on gradient-based approaches to solve MOO in high-dimensional deep-learning models. We first provide a brief overview of two typical approaches: linear scalarization (LS) and gradient manipulation methods.

1) LS: A straightforward approach is to transform it into a single objective problem by using pre-defined weights $\boldsymbol{\lambda}$: $\min_{\mathbf{x}\in\mathcal{D}} \boldsymbol{\lambda}^\top \mathbf{F}(\mathbf{x})$. Thanks to the LS, one can leverage many existing single-objective methods (e.g., gradient descent), for training at each iteration. However, in LS, it is not uncommon to see conflicts among multiple objectives during the optimization process, i.e., $\langle \nabla f_s(\mathbf{x}), \nabla f_{s'}(\mathbf{x}) \rangle < 0$. This implies that the update using static weights may decrease some objectives, while inevitably increasing others at the same time and leading to slow convergence and poor performance.

2) Gradient Manipulation Methods: A popular alternative is to dynamically weight gradients across objectives to avoid such conflicts and obtain a direction $\mathbf{d}$ to decrease all objectives simultaneously. For example, multiple gradient descent algorithm (MGDA) (Fliege and Svaiter 2000) seeks to find the $\mathbf{d}$ by solving the following optimization problem given gradients: $(\mathbf{d}, \beta) \in \arg\min_{\mathbf{d}\in\mathbb{R}^s, \beta\in\mathbb{R}} \beta + \frac{1}{2}\|\mathbf{d}\|^2, s.t., \nabla f_s(\mathbf{x})^T\mathbf{d} - \beta \leq 0, \forall s \in [S]$. By doing so, if $\mathbf{x}$ represents a first-order Pareto stationary point, then $(\mathbf{d}, \beta) = (\mathbf{0}, 0)$. Otherwise, we can find the $\mathbf{d}$ such that $\nabla f_s(\mathbf{x})^T\mathbf{d} \leq \beta < 0, s \in [S]$ as the solution. Using such a non-conflicting direction $\mathbf{d}$ to update the model, i.e., $\mathbf{x} \leftarrow \mathbf{x} + \eta\mathrm{d}$ where η is the learning rate, has been shown to achieve better performance in practice (Sener and Koltun 2018; Liu et al. 2021a). Following this token, many gradient manipulation methods attempt to obtain a common descent direction through various formulations and solutions. For example, GRADDROP (Chen et al. 2020) randomly dropped out highly conflicted gradients, RotoGrad (Javaloy and Valera 2021) rotated objective gradients to alleviate the conflict, and many other methods exploring similar principles (Kendall, Gal, and Cipolla 2018; Chen et al. 2018; Yu et al. 2020; Liu et al. 2021a; Liu and Vicente 2021; Chen et al. 2024; Xiao, Ban, and Ji 2024). However, due to the additional optimization process required to generate such $\mathbf{d}$, gradient manipulation methods often incur an expensive per-iteration cost, thereby prolonging training time. In this work, we propose a new algorithm, PSMGD, which achieves fast per-iteration and overall convergence, along with enhanced performance.

3 Periodic Stochastic Multi-Gradient Descent

In this section, we first present the basic concept of MOO, followed by a pedagogical example comparing linear scalarization with MGDA, a representative of gradient manipulation methods. Based on one observation of the stable variation of dynamic weights for MGDA, we propose the PSMGD algorithm, followed by its convergence analyses.

3.1 Preliminaries of MOO

Analogous to the stationary and optimal solutions in single-objective optimization, MOO seeks to adopt the notion of Pareto optimality/stationarity:

Definition 1 ((Weak) Pareto Optimality)**.** *For any two solutions* $\mathbf{x}$ *and* $\mathbf{y}$*, we say* $\mathbf{x}$ *dominates* $\mathbf{y}$ *if and only if* $f_s(\mathbf{x}) \leq f_s(\mathbf{y}), \forall s \in [S]$ *and* $f_s(\mathbf{x}) < f_s(\mathbf{y}), \exists s \in [S]$*. A solution* $\mathbf{x}$ *is Pareto optimal if it is not dominated by any other solution. One solution* $\mathbf{x}$ *is weakly Pareto optimal if there does not exist a solution* $\mathbf{y}$ *such that* $f_s(\mathbf{x}) > f_s(\mathbf{y}), \forall s \in [S]$.

Definition 2 (Pareto Stationarity)**.** *A solution* $\mathbf{x}$ *is said to be Pareto stationary if there is no common descent direction* $\mathbf{d} \in \mathbb{R}^d$ *such that* $\nabla f_s(\mathbf{x})^\top \mathbf{d} < 0, \forall s \in [S]$.

Similar to solving single-objective non-convex optimization problems, finding a Pareto-optimal solution in MOO is NP-Hard in general. As a result, it is often of practical interest to find a solution satisfying Pareto-stationarity (a necessary condition for Pareto optimality). Following Definition 2, if $\mathbf{x}$ is not a Pareto stationary point, we can find a common descent direction $\mathbf{d} \in \mathbb{R}^d$ to decrease all objectives simultaneously, i.e., $\nabla f_s(\mathbf{x})^\top \mathbf{d} < 0, \forall s \in [S]$. If no such a common descent direction exists at $\mathbf{x}$, then $\mathbf{x}$ is a Pareto stationary solution. For example, multiple gradient descent algorithm (MGDA) (Désidéri 2012) searches for an optimal weight $\boldsymbol{\lambda}_*$ of gradients $\nabla\mathbf{F}(\mathbf{x}) := \{\nabla f_s(\mathbf{x}), \forall s \in [S]\}$ by solving $\boldsymbol{\lambda}_*(\mathbf{x}) = \mathrm{argmin}_{\boldsymbol{\lambda}} \|\boldsymbol{\lambda}^\top \nabla\mathbf{F}(\mathbf{x})\|^2$, which is the dual of the original problem. Then, a common descent direction can be chosen as: $\mathbf{d} = \boldsymbol{\lambda}_*^\top \nabla\mathbf{F}(\mathbf{x})$. MGDA performs the iterative update rule: $\mathbf{x} \leftarrow \mathbf{x} - \eta\mathbf{d}$ until a Pareto optimal/stationary point is reached, where η is a learning rate. Many gradient

Convexity	Algorithm	Assume Lipschitz continuity of $\boldsymbol{\lambda}^*(\mathbf{x})$	Convergence Rate	BP Complexity
Strongly Convex	SMG (Liu and Vicente 2021)	✓	$\mathcal{O}(\frac{1}{T})$	$\mathcal{O}(\frac{S}{\epsilon})$
	MoDo (Chen et al. 2024)	✗	$\mathcal{O}(\frac{1}{T})$	$\mathcal{O}(\frac{S}{\epsilon})$
	CR-MOGM (Zhou et al. 2022)	✗	$\mathcal{O}(\frac{1}{T})$	$\mathcal{O}(\frac{S}{\epsilon})$
	PSMGD	✗	$\mathcal{O}(\frac{1}{T})$	$\mathcal{O}(\frac{S}{\epsilon R}+\frac{(R-1)}{\epsilon R})$
General Convex	SMG (Liu and Vicente 2021)	✓	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\frac{S}{\epsilon^2}$
	CR-MOGM (Zhou et al. 2022)	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\frac{S}{\epsilon^2}$
	PSMGD	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\mathcal{O}(\frac{S}{\epsilon^2 R}+\frac{(R-1)}{\epsilon^2 R})$
Non-Convex	CR-MOGM (Zhou et al. 2022)	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\frac{S}{\epsilon^2}$
	MoCO (Fernando et al. 2022)	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\frac{S}{\epsilon^2}$
	MoDo (Chen et al. 2024)	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\frac{S}{\epsilon^2}$
	SDMGrad (Xiao, Ban, and Ji 2024)	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\frac{S}{\epsilon^2}$
	PSMGD	✗	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\mathcal{O}(\frac{S}{\epsilon^2 R}+\frac{(R-1)}{\epsilon^2 R})$

Table 1: Comparison of different algorithms for MOO problem in stochastic first order oracle.

1. BP Complexity measures the total number of backpropagation to achieve certain metric (ϵ), which is defined in Def 3.
2. Notations: T is the number of iterations, S is the number of objectives, R is the weight calculation hyper-parameter.

manipulation algorithms have been inspired by MGDA. In the next subsection, we compare two typical gradient-based approaches using a pedagogical example, with MGDA representing the gradient manipulation approach.

3.2 A Pedagogical Example

We consider the Fonseca problem(Fonseca and Fleming 1996; Pardalos et al. 2017), which is to solve a two-objective optimization problem, $\min_{\mathbf{x}}[f_1(\mathbf{x}), f_2(\mathbf{x})]$. These two objective functions are defined by $f_1(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{d}\left(x_d - \frac{1}{\sqrt{d}}\right)^2\right), f_2(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{d}\left(x_d + \frac{1}{\sqrt{d}}\right)^2\right)$, where $\mathbf{x} = (x_1, x_2, ..., x_d)$ representing the d-dimensional decision variable. The problem exhibits a non-convex Pareto front in the objective space. We run three algorithms on this pedagogical example: linear scalarization, MGDA, and our proposed PSMGD.

Setting. We run each algorithm 10 times. In Figure 1, we show the trajectory of different methods from random starting points. The blue dashed lines represent the trajectories, with the final solutions highlighted by red dots.

Observation 1: Linear scalarization can only converge to specific points and is unable to explore different trade-offs among objectives. we run the linear scalarization method by using randomly static weights. As shown in Figure 1(a), it is obvious that linear scalarization can only converge to two specific points (see red dots), with either $f_1(\mathbf{x})$ or $f_2(\mathbf{x})$ being under-optimized. In other words, linear scalarization cannot fully explore the Pareto front, which is consistent with previous results (Miettinen 1999; Pardalos et al. 2017). In contrast, MGDA can find trade-off points, and our proposed PSMGD successfully identifies a set of well-distributed Pareto solutions with different trade-offs, shown in Figure 1(b,c). We note that linear scalarization usually runs faster for MOO in each iteration than gradient manipulation methods due to its simplicity, as shown in many existing works (Kurin et al. 2022; Liu et al. 2024) and confirmed by our experimental results (see Sec 4). However, our observation highlights the necessity of gradient manipulation methods and motivates us to develop faster gradient manipulation techniques for MOO.

Observation 2: The dynamic weights calculated by MGDA converge quickly and exhibit stability. As shown in Figure 1(d), we visualize the dynamic weights for the Fonseca problem. In this simple problem, the weights converge within 5 iterations, after which they remain unchanged. In more complex problems, such as multi-task learning using deep learning (see Sec B in Appendix), the dynamic weights calculated by gradient manipulation methods also demonstrate stability. We verified stability by examining two scales, epoch and iteration. The weights show consistent behavior with a stable average value and low variability across both scales, epoch and iteration, as depicted in Figure 6(a) and 6(b) in Appendix. This observation suggests a new approach to develop more efficient MOO algorithms. Specifically, we can calculate and update the dynamic weights periodically in a more lazy and thus efficient manner.

3.3 PSMGD

Based on our observations, we propose a new and efficient algorithm called Periodic Stochastic Multi-Gradient Descent (PSMGD) for MOO, as shown in Algorithm 1. Our PSMGD algorithm is designed to periodically compute these dynamic weights and utilizes them repeatedly, thereby reducing the computational load effectively. Specifically, in each iteration $t \in [T]$, we offer two options: 1) If $t\%R == 0$, we calculate the dynamic weights $\hat{\boldsymbol{\lambda}}_t^*$ based on the stochastic gradient of each objective $\nabla f_s(\mathbf{x}_t, \xi_t), s \in [S]$, by solving optimization problem 2. Following this, we apply momentum to $\boldsymbol{\lambda}$ to further stabilize the weights: $\boldsymbol{\lambda}_t = \alpha_t \boldsymbol{\lambda}_{t-R} + (1-\alpha_t)\hat{\boldsymbol{\lambda}}_t^*$. 2) Otherwise, we reuse the dynamic weights: $\boldsymbol{\lambda}_t = \boldsymbol{\lambda}_{t-1}$. With predefined $\boldsymbol{\lambda}_t$, it is worth pointing out that the MOO

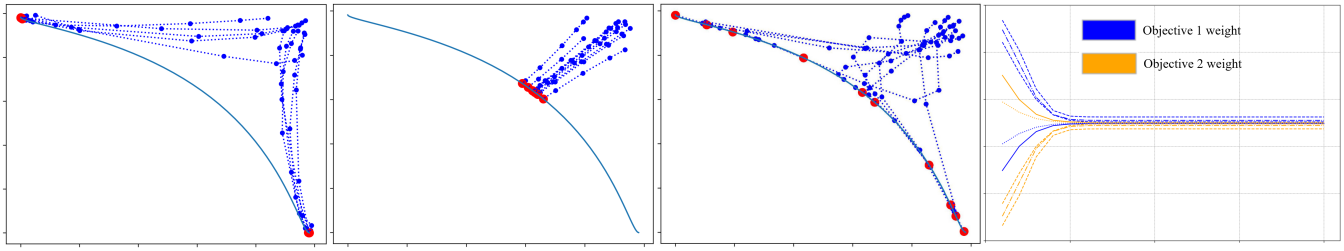

Figure 1: The convergence behaviors on a pedagogical example with 10 runs for each algorithm from left to right: (a) Solutions obtained from random linear scalarization. (b) Solutions obtained from the MGDA method. (c) Solutions obtained from the PSMGD method proposed in this paper, with weights updated every 4 iterations ($R = 4$) through the training. (d) We visualize weights changing curves over iterations in 5 runs. This method successfully generates a set of widely distributed Pareto solutions with different trade-offs. Details of the pedagogical example can be found in Section 3.2.

Algorithm 1: Periodic Stochastic Multi-Gradient Descent (PSMGD)

1: Initialize model parameter $\mathbf{x}_0$, learning rate η, and hyper-parameter R.
2: **for** $t = 0, \ldots, T-1$ **do**
3: If $t\%R == 0$: ▶ Weights Calculation
4: Compute $\hat{\boldsymbol{\lambda}}_t^* \in [0,1]^S$ by solving

$$\min_{\boldsymbol{\lambda}} \left\| \sum\nolimits_{s\in[S]} \lambda_s \nabla f_s(\mathbf{x}_t, \xi_t) \right\|^2, \quad \text{s.t.} \quad \sum\nolimits_{s\in[S]} \lambda_s = 1. \tag{2}$$

5: Update: $\boldsymbol{\lambda}_t = \alpha_t \boldsymbol{\lambda}_{t-R} + (1-\alpha_t)\hat{\boldsymbol{\lambda}}_t^*$.
6: Otherwise: $\boldsymbol{\lambda}_t = \boldsymbol{\lambda}_{t-1}$. ▶ Reuse the weights
7: Update the model: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{d}_t$, where $\mathbf{d}_t = \sum_{s\in[S]} \lambda_{t,s} \nabla f_s(\mathbf{x}_t, \xi_t)$.
8: **end for**

can be naturally transformed into a single-objective problem by using pre-defined weighted sum of the objectives, thereby requiring only one backpropagation step. Subsequently, after obtaining the weights across objectives, we can approximate the common descent direction by $\mathbf{d}_t = \boldsymbol{\lambda}_t^T \nabla \mathbf{F}(\mathbf{x}_t, \xi_t)$. Then the model can be updated by $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{d}_t$.

3.4 Convergence Analysis

Assumption 3.1 (L-Lipschitz continuous)**.** $\|\nabla f_s(\mathbf{x}) - \nabla f_s(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, s \in [S]$.

Assumption 3.2 (Bounded variance)**.** We assume the stochastic gradient estimation is unbiased with bounded variance.

$$\mathbb{E}\left[\nabla f_s(\mathbf{x}, \xi)\right] = \nabla f_s(\mathbf{x}), \tag{3}$$

$$\mathbb{E}\left[\|\nabla f_s(\mathbf{x}, \xi) - \nabla f_s(\mathbf{x})\|^2\right] \leq \sigma^2, s \in [S]. \tag{4}$$

Assumption 3.3 (Bounded weights)**.** There exists a constant B s.t., $0 \leq \lambda_{t,s} \leq B, \sum_{s\in[S]} \lambda_{t,s} \geq 1, \forall s \in [S], t \in [T]$.

Assumption 3.4 (Bounded Gradient)**.** The gradient of each objective is bounded, i.e., there exists a constant $H > 0$ such that $\|\nabla f_s(\mathbf{x})\| \leq H, \forall s \in [S]$.

These four assumptions are widely-used assumptions in MOO (Liu and Vicente 2021; Zhou et al. 2022). For notation clarity, we have the following definition: $G(\mathbf{x}_t, \boldsymbol{\lambda}_t) = \sum_{s\in[S]} \lambda_{t,s} f_s(\mathbf{x}_t), \nabla \mathbf{G}(\mathbf{x}_t, \boldsymbol{\lambda}_t) = \sum_{s\in[S]} \lambda_{t,s} \nabla f_s(\mathbf{x}_t)$. With these assumptions and definitions, we have the following convergence rates:

Theorem 3.5 (Non-Convex Functions)**.** *Under Assumptions 3.1- 3.4, when each objective is bounded by F ($f_s(\mathbf{x}) \leq F, s \in [S]$), the sequence of iterates generated by the PSMGD Algorithm in non-convex functions satisfies:*

$$\frac{1}{T}\sum_{t\in[T]} \mathbb{E}[\|\nabla \mathbf{G}(\mathbf{x}_t, \boldsymbol{\lambda}_t)\|^2] \tag{5}$$

$$\leq \frac{4SBF}{T\eta_T} + \frac{2F}{T} \sum_{\substack{t\%R==0 \\ t\neq 0}} \frac{\mathbb{E}\left[\sum_{s\in[S]}(1-\alpha_t)|\hat{\lambda}_t^s - \lambda_{t-R}^s|\right]}{\eta_t}$$

$$+ 2LS^2B^2\sigma^2 \frac{1}{T}\sum_{t\in[T]} \eta_t + \frac{4S^{3/2}B^2H\sigma}{T} \sum_{t\%R==0} (1-\alpha_t).$$

Setting $1-\alpha_t = \min\{\frac{\eta_t}{\eta_1}, \frac{\eta_t}{\eta_1\sqrt{t}\max_{s\in[S]}|\hat{\lambda}_t^s - \lambda_{t-R}^s|}\}, \eta_t = \mathcal{O}(\frac{1}{\sqrt{t}})$*, the convergence rate is* $\mathcal{O}(\frac{1}{\sqrt{T}})$.

Theorem 3.6 (General Convex Functions)**.** *Under Assumptions 3.1- 3.4, when the distance from sequence to Pareto set is bounded ($\|\mathbf{x}_t - \mathbf{x}_*\| \leq D$), the sequence of iterates generated by the PSMGD in general convex functions satisfies:*

$$\frac{1}{T}\sum_{t\in[T]} \mathbb{E}[G(\mathbf{x}_t, \boldsymbol{\lambda}_t) - G(\mathbf{x}_*, \boldsymbol{\lambda}_t)] \tag{6}$$

$$\leq \frac{\|\mathbf{x}_1 - \mathbf{x}_*\|^2}{T\eta_1} + 2D\sigma S^{3/2}B\frac{1}{T}\sum_{t\%R==0}(1-\alpha_t)$$

$$+ (2S^2B^2\sigma^2 + 2S^2B^2H^2)\frac{1}{T}\sum_{t\in[T]}\eta_t.$$

Setting $\eta_t = \frac{1}{\sqrt{t}}$ *and* $(1-\alpha_t) = \eta_t$*, the convergence rate is* $\mathcal{O}(1/\sqrt{T})$.

Theorem 3.7 (μ-Strongly Convex Functions)**.** *Under Assumptions 3.1- 3.4 and we further assume each objective function $f_s(\mathbf{x}), s \in [S]$ is $\mu-$strongly convex and each objective is bounded by F ($f_s(\mathbf{x}) \leq F, s \in [S]$), the sequence of iterates generated by the PSMGD satisfies:*

$$\begin{aligned}&\mathbb{E}[G(\mathbf{x}_{t+1}, \boldsymbol{\lambda}_{t+1}) - G(\mathbf{x}^*_{t+1}, \boldsymbol{\lambda}_{t+1})] \\ &\leq (1 - 2\mu\eta_t)\mathbb{E}[G(\mathbf{x}_t, \boldsymbol{\lambda}_t) - G(\mathbf{x}^*_t, \boldsymbol{\lambda}_t)] + \eta_t^2\Phi, \quad (7)\end{aligned}$$

where $\Phi = 2LS^2B^2\sigma^2 + 4FSB\mathbf{1}\{(t+1)\%R = 0\} + 4S^{3/2}B^2H\sigma\mathbf{1}\{t\%R = 0\}$ and $\mathbf{1}$ is the indicator function. Set $\eta = \frac{c}{T}$ with $c > \frac{1}{\mu}$, the convergence rate is

$$\begin{aligned}&\mathbb{E}[G(\mathbf{x}_T, \boldsymbol{\lambda}_T) - G(\mathbf{x}^*_T, \boldsymbol{\lambda}_T)] \\ &\leq \frac{\max\{2c^2\Phi^{'}(2\mu c - 1)^{-1}, G(\mathbf{x}_0, \boldsymbol{\lambda}_0) - G(\mathbf{x}^*_0, \boldsymbol{\lambda}_0)\}}{T} \\ &= \mathcal{O}(1/T), \quad (8)\end{aligned}$$

where $\Phi^{'} = 2LS^2B^2\sigma^2 + 4FSB/R + 4S^{3/2}B^2H\sigma/R$.

Remark 3.8. 1) Convergence Metrics. Different convergence metrics are used in existing studies, and we follow convergence conditions outlined in (Tanabe, Fukuda, and Yamashita 2019; Zhou et al. 2022). In the appendix, we provide an explanation of convergence metrics and a comparison with other works. 2) Convergence Rates. With proper hyperparameters, our algorithm, PSMGD , can achieve $\mathcal{O}(1/T)$ for strongly convex functions and $\mathcal{O}(1/\sqrt{T})$ for general convex and non-convex functions. These convergence rates match the state-of-the-art rates in existing MOO algorithms.

The convergence rate reflects the training speed in terms of iterations but does not fully capture the total computational complexity. Similar to how sample complexity is used in single-objective learning, we propose a new metric, Backpropagation (BP) complexity, to quantify the computational workload tailored to MOO in first-order oracle.

Definition 3 (Backpropogation complexity)**.** *We define Backpropagation Complexity as the total number of backpropagation operations required by an algorithm to achieve a specified performance threshold, denoted as ϵ.*

Remark 3.9. Our PSMGD algorithm can achieve BP complexity of $\mathcal{O}(\frac{S}{\epsilon R} + \frac{(R-1)}{\epsilon R})$ for stongly-convex functions and $\mathcal{O}(\frac{S}{\epsilon^2 R} + \frac{(R-1)}{\epsilon^2 R})$ for general convex and non-convex functions. Compared to existing MOO methods, PSMGD can achieve a linear speedup in terms of R.

Remark 3.10. If $R = \Omega(S)$, PSMGD exhibits an *objective-independent BP complexity*, i.e., $\mathcal{O}(\frac{1}{\epsilon})$ for stongly-convex functions and $\mathcal{O}(\frac{1}{\epsilon^2})$ for general convex and non-convex functions. These rates indicate that PSMGD requires the same order of computations for MOO as classic SGD does for single-objective learning. We provide a comparison of these rates and complexities in Table 1, which are also validated by extensive experiments in Sec 4.

4 Experiment

In this section, we conduct a comprehensive empirical evaluation of the proposed PSMGD algorithm, focusing primarily on multi-task learning with deep neural networks. The primary goal is to verify the following key points:

- Improved Performance: Can PSMGD provide improved model performance?
- Fast Training: Can PSMGD have a fast training process?
- Ablation Study: How does the hyper-parameter impact the training? (Results In Appendix)

Baseline Algorithms. To conduct an extensive comparison, we use 14 algorithms as the baselines, including a single objective learning baseline, linear scalarization, and 12 advanced MOO algorithms. They are: **(1)** Single task learning (STL), training independent models $f_S(\mathbf{x})$ for all objectives; **(2)** Linear scalarization (LS) baseline; **(3)** Scale-invariant (SI) that minimizes $\sum_{s\in[S]} \log f_s(\mathbf{x})$; **(4)** Dynamic Weight Average (DWA) (Liu, Johns, and Davison 2019) adaptively updates weights based on the comparative rate of loss reduction for each objective; **(5)** Uncertainty Weighting (UW) (Kendall, Gal, and Cipolla 2018) leverages estimated task uncertainty to guide weight assignments; **(6)** Random Loss Weighting (RLW) (Lin, Feiyang, and Zhang 2021) samples objective weighting with log-probabilities by normal distribution; **(7)** MGDA (Sener and Koltun 2018) identifies a joint descent direction that concurrently decreases all objectives; **(8)** PCGRAD (Yu et al. 2020) projects each objective gradient onto the normal plan of the gradients to avoid conflicts; **(9)** CAGRAD (Liu et al. 2021a) balances the average loss while guaranteeing a controlled minimum improvement for each objective; **(10)** IMTL-G (Liu et al. 2021b) determines the update direction by having equal projections on objective gradients; **(11)** GRADDROP (Chen et al. 2020) randomly drops out certain dimensions of the objective gradients for their level of conflict; **(12)** NASHMTL (Navon et al. 2022) determines the game's solution that is advantageous for all goals by a bargaining game; **(13)** FAMO (Liu et al. 2024) is a dynamic weighting method that reduces objective losses in space and time in a balanced manner; **(14)** FAIRGRAD (Maheshwari and Perrot 2022) aims to achieve group fairness in MOO by dynamically re-weighting; **(15)** SDMGRAD (Xiao, Ban, and Ji 2024) utilizes direction-oriented regularization.

Datasets. We evaluate on 5 multi-task learning datasets with objective/task numbers ranging from 2 to 40, covering scenarios in regression, classification, and dense prediction. For regression, we choose QM-9 dataset (Blum and Reymond 2009) (11 tasks), a widely used benchmark in graph neural network learning. For image classification, we use Multi-MNIST (Sener and Koltun 2018) (2 tasks) and CelebA dataset (Liu et al. 2015) (40 tasks). For dense prediction, CityScapes (Cordts et al. 2016) (2 tasks) and NYU-v2 (Silberman et al. 2012) (3 tasks) are used in our experiments. NYU-v2 is a dataset of indoor scenes with 1449 RGBD images and dense per-pixel labeling across 13 classes. CityScapes is similar to NYU-v2 but boasts 5K street-view RGBD images with per-pixel annotations. We present the results for QM-9 and NYU-v2 in this section, with remaining results and experimental settings detailed in Appendix B.

Metrics. We have two types of metrics to measure MOO algorithms. *I. Model performance metrics.* We consider two widely-used metrics to represent the overall performance of one MOO method m: **(1)** $\boldsymbol{\Delta m}\%$, the average per-task performance drop of a method m relative to the STL baseline de-

Method	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	**MR** ↓	$\boldsymbol{\Delta m}\%$ ↓
	MAE ↓												
STL	0.07	0.18	60.6	53.9	0.50	4.53	58.8	64.2	63.8	66.2	0.07		
LS	0.11	0.33	**73.6**	89.7	5.20	14.06	143.4	144.2	144.6	140.3	0.13	9.00	177.6
SI	0.31	0.35	149.8	135.7	**1.00**	**4.51**	**55.3**	**55.8**	**55.8**	**55.3**	0.11	5.36	77.8
RLW	0.11	0.34	76.9	92.8	5.87	15.47	156.3	157.1	157.6	153.0	0.14	10.36	203.8
DWA	0.11	0.33	74.1	90.6	5.09	13.99	142.3	143.0	143.4	139.3	0.13	8.64	175.3
UW	0.39	0.43	166.2	155.8	1.07	4.99	66.4	66.8	66.8	66.2	0.12	6.64	108.0
MGDA	0.22	0.37	126.8	104.6	3.23	5.69	88.4	89.4	89.3	88.0	0.12	8.36	120.5
PCGRAD	0.11	0.29	75.9	88.3	3.94	9.15	116.4	116.8	117.2	114.5	0.11	7.18	125.7
CAGRAD	0.12	0.32	83.5	94.8	3.22	6.93	114.0	114.3	114.5	112.3	0.12	8.18	112.8
IMTL-G	0.14	0.29	98.3	93.9	1.75	5.70	101.4	102.4	102.0	100.1	0.10	6.64	77.2
NASHMTL	**0.10**	**0.25**	82.9	81.9	2.43	5.38	74.5	75.0	75.1	74.2	**0.09**	**3.82**	62.0
FAMO	0.15	0.30	94.0	95.2	1.63	4.95	70.82	71.2	71.2	70.3	0.10	5.09	58.5
FAIRGRAD	0.12	**0.25**	87.57	84.00	2.15	5.07	70.89	71.17	71.21	70.88	0.10	4.09	**57.9**
PSMGD	0.12	**0.25**	77.2	**74.4**	3.01	6.61	103.0	103.5	103.7	101.6	**0.09**	5.55	92.4

Table 2: Results obtained on the QM-9 dataset. Each experiment is conducted using 3 random seeds, and the mean value is presented. The best average result is highlighted in bold.

noted as B: $\boldsymbol{\Delta m}\% = \frac{1}{N}\sum_{n=1}^{N}(-1)^{\delta_n}\frac{(M_{m,n}-M_{B,n})}{M_{B,n}} \times 100$, where $M_{B,n}$ and $M_{m,n}$ represent the STL and m's value for metric M_n, respectively. Here, δ_n equals 1 if a higher value of M_n is better or 0 if a lower value of M_n is better. **(2) Mean Rank (MR)**, the average rank of each method in the tasks. For example, if a method ranks first in every task (whether higher or lower is better), **MR** will be 1. Note that in practice, lower values of $\boldsymbol{\Delta m}\%$ and **MR** indicate better overall performance in the final result. *II. Convergence metrics.* We calculate the training time per epoch and total training time to evaluate the convergence performance of MOO algorithms. Based on various datasets, we also calculate the test loss for a visualization comparison through the training process.

4.1 Regression: QM-9

QM-9 dataset (Blum and Reymond 2009) is a crucial benchmark widely used in the field of graph neural network learning, especially in chemical informatics applications. With over 13K molecules, each molecule is intricately depicted as a graph, with nodes representing atoms and edges symbolizing the chemical bonds between them. These graphs feature detailed node and edge characteristics such as atomic number, atom type, bond type, and bond order. The goal is to predict 11 molecule properties. Utilizing 110K molecules from QM9 in PyTorch Geometric (Fey and Lenssen 2019) for training, 10K for validation, and the remaining 10K for testing.

Performance. We assess two common metrics, **MR** and $\boldsymbol{\Delta m}\%$, for various MOO algorithms to ensure effective evaluations. Table 2 shows that PSMGD achieves comparable or even better performance in both **MR** and $\boldsymbol{\Delta m}\%$ when compared with existing baselines. Specifically, it has the best performance in 3 out of the 11 total tasks, namely α, ϵ_{LUMO}, and c_v. When we take a closer look at the training process, we observe that PSMGD has a faster and more smooth process as illustrated in Figure 2. Specifically, PSMGD demonstrates a smoother descent towards lower test loss at least in the early stages. In addition, PSMGD exhibits resistance to overfitting, distinguishing itself from IMTL-G and NashMTL due to its straightforward yet efficient structure.

Convergence measurements. In Table 3, we show the training time per epoch and the total training time required to achieve a certain loss. In Figure 2, we use the average Mean Absolute Error (MAE) across all molecular property prediction tasks as test loss, calculated after scaling both the predictions and the ground truth values by the standard deviation. PSMGD ranks third in training speed per epoch but achieves the average test loss (<100, <75) in the shortest overall time. The visualization results can be found in Appendix B.1.

Method	Training time ↓ (mean per epoch)	Avg loss ↓ < 100	Avg loss ↓ < 75
LS	**1.70**	75.68	170.28
MGDA	12.94	436.92	754.68
PCGRAD	6.15	259.56	630.36
CAGRAD	5.29	233.64	451.35
IMTL-G	5.38	275.40	604.80
FAMO	1.96	72.52	131.32
NASHMTL	8.45	287.30	371.80
FAIRGRAD	5.40	162.60	314.36
PSMGD	2.23	**69.13**	**109.27**

Table 3: Results obtained on the QM-9 dataset. Each experiment is conducted using 3 random seeds, and the mean value is presented. The best average result is highlighted in bold.

4.2 Dense Prediction: NYU-v2

NYU-v2 (Silberman et al. 2012) is an indoor scene dataset containing 1449 RGBD images, each annotated with dense per-pixel labeling across 13 distinct classes. The dataset supports 3 tasks for MOO experiments: image segmentation, depth prediction, and surface normal prediction.

Method	Segmentation		Depth		Surface Normal					MR ↓	$\Delta m\%$ ↓
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Dist ↓		Within t° ↑				
					Mean	Median	11.25	22.5	30		
STL	38.30	63.76	0.6754	0.2780	25.01	19.21	30.14	57.20	69.15		
LS	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08	11.44	5.59
SI	38.45	64.27	0.5354	0.2201	27.60	23.37	22.53	48.57	62.32	10.11	4.39
RLW	37.17	63.77	0.5759	0.2410	28.27	24.18	22.26	47.05	60.62	14.11	7.78
DWA	39.11	65.31	0.5510	0.2285	27.61	23.18	24.17	50.18	62.39	10.44	3.57
UW	36.87	63.17	0.5446	0.2260	27.04	22.61	23.54	49.05	63.65	10.11	4.05
MGDA	30.47	59.90	0.6070	0.2555	24.88	19.45	29.18	56.88	69.36	8.11	1.38
PCGRAD	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14	10.67	3.97
GRADDROP	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87	9.56	3.58
CAGRAD	39.79	65.49	0.5486	0.2250	26.31	21.58	25.61	52.36	65.58	7.00	0.20
IMTL-G	39.35	65.60	0.5426	0.2256	26.02	21.19	26.20	53.13	66.24	6.33	-0.76
NASHMTL	40.13	65.93	0.5261	0.2171	25.26	20.08	28.40	55.47	68.15	4.22	-4.04
FAMO	38.88	64.90	0.5474	0.2194	25.06	19.57	29.21	56.61	68.98	5.11	-4.10
FAIRGRAD	39.74	**66.01**	0.5377	0.2236	24.84	19.60	29.26	56.58	69.16	**3.22**	-4.66
SDMGRAD	**40.47**	65.90	**0.5225**	**0.2084**	25.07	19.99	28.54	55.74	68.53	3.44	**-4.84**
PSMGD	35.44	63.78	0.5494	0.2369	**24.83**	**18.89**	**30.68**	**58.00**	**69.84**	6.11	-3.62

Table 4: Results obtained on NYU-v2. Each experiment is conducted using 3 random seeds, and the mean value is presented. The best average result is highlighted in bold.

Method	Training time ↓ (mean per epoch)	Semantic loss ↓ < 1.40	Semantic loss ↓ < 1.20	Depth loss ↓ < 0.65	Depth loss ↓ < 0.55	Normal loss ↓ < 0.20	Normal loss ↓ < 0.16
LS	**1.36**	32.14	66.28	46.18	159.36	**30.84**	152.76
MGDA	3.34	166.43	384.32	218.88	590.96	56.44	249.77
PCGRAD	3.31	72.38	154.63	98.79	299.39	75.67	332.29
CAGRAD	3.53	63.18	137.89	94.77	351.06	66.69	213.11
IMTL-G	4.63	101.42	239.72	147.52	470.22	87.59	318.09
FAMO	1.65	32.69	68.46	73.35	**156.26**	34.23	114.13
NASHMTL	3.49	76.34	159.62	79.81	277.60	65.93	201.26
FAIRGRAD	2.93	141.82	328.86	148.47	334.53	48.20	177.41
SDMGRAD	1.98	31.52	72.89	63.04	200.94	35.46	128.05
PSMGD	1.85	**29.28**	**64.54**	**43.07**	176.66	31.11	**113.46**

Table 5: Training time per epoch [Min.] and convergence process (averaged over 3 random seeds) in all tasks on NYU-v2.

Performance. As shown in Table 4, PSMGD achieves better performance in both **MR** and $\Delta m\%$ compared to existing MOO algorithms. It performs the best in 5 out of the 9 total tasks. In Figure 3, PSMGD achieves consistently low test losses across all tasks compared to other methods in the initial stage, indicating superior performance across all 3 tasks throughout the 200 training epochs.

Convergence measurements. In Figure 3, PSMGD outperforms other MOO methods in reducing test loss for all three tasks on the NYU-v2 dataset, particularly in the early training stages. Table 5 further highlights PSMGD's efficiency with its third lowest training time per epoch (1.85), significantly outperforming other classical gradient manipulation methods like IMTL-G (Liu et al. 2021b) and NASHMTL (Navon et al. 2022). PSMGD achieves the fastest training times by reaching semantic loss in 29.28 and 64.54 minutes, depth loss in 43.07 minutes, and normal loss in 113.46 minutes, when considering the following thresholds: <1.40 and <1.20 for semantic loss, <0.65 for depth loss, and <0.16 for normal loss. The visualization results can be found in Appendix B.1.

5 Conclusion

In this paper, we propose a novel and efficient algorithm, Periodic Stochastic Multi-Gradient Descent (PSMGD), to accelerate MOO. Our PSMGD algorithm periodically computes dynamic weights and reuses them, significantly reducing the computational load and speeding up MOO training. We establish that PSMGD achieves state-of-the-art convergence rates for strongly convex, general convex, and non-convex functions. Moreover, we demonstrate the superior backpropagation (BP) complexity of our PSMGD algorithm. Extensive experiments confirm that PSMGD delivers performance comparable to or better than existing MOO algorithms, with a substantial reduction in training time. We believe future work could explore preference-based solutions or the entire Pareto set using our efficient PSMGD algorithm.

Acknowledgments

JL acknowledges the funding from NSF grants CAREER CNS-2110259, CNS-2112471, IIS-2324052, DARPA YFA D24AP00265, ONR grant N00014-24-1-2729, and AFRL grant PGSC-SC-111374-19s. HY acknowledges the funding support from AI Seed Funding and GWBC Award at RIT.

References

Belakaria, S.; Deshwal, A.; Jayakodi, N. K.; and Doppa, J. R. 2020. Uncertainty-aware search framework for multi-objective Bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 10044–10052.

Blum, L. C.; and Reymond, J.-L. 2009. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, 131(25): 8732–8733.

Caruana, R. 1997. Multitask learning. *Machine learning*, 28: 41–75.

Chen, L.; Fernando, H.; Ying, Y.; and Chen, T. 2024. Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance. *Advances in Neural Information Processing Systems*, 36.

Chen, Z.; Badrinarayanan, V.; Lee, C.-Y.; and Rabinovich, A. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, 794–803. PMLR.

Chen, Z.; Ngiam, J.; Huang, Y.; Luong, T.; Kretzschmar, H.; Chai, Y.; and Anguelov, D. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33: 2039–2050.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

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

Désidéri, J.-A. 2012. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6): 313–318.

Fernando, H. D.; Shen, H.; Liu, M.; Chaudhury, S.; Murugesan, K.; and Chen, T. 2022. Mitigating gradient bias in multi-objective learning: A provably convergent approach. In *The Eleventh International Conference on Learning Representations*.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

Fliege, J.; and Svaiter, B. F. 2000. Steepest descent methods for multicriteria optimization. *Mathematical methods of operations research*, 51: 479–494.

Fliege, J.; Vaz, A. I. F.; and Vicente, L. N. 2019. Complexity of gradient descent for multiobjective optimization. *Optimization Methods and Software*, 34(5): 949–959.

Fonseca, C. M.; and Fleming, P. J. 1996. On the performance assessment and comparison of stochastic multiobjective optimizers. In *International conference on parallel problem solving from nature*, 584–593. Springer.

Hayes, C. F.; Rădulescu, R.; Bargiacchi, E.; Källström, J.; Macfarlane, M.; Reymond, M.; Verstraeten, T.; Zintgraf, L. M.; Dazeley, R.; Heintz, F.; et al. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1): 26.

Javaloy, A.; and Valera, I. 2021. Rotograd: Gradient homogenization in multitask learning. *arXiv preprint arXiv:2103.02631*.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7482–7491.

Kurin, V.; De Palma, A.; Kostrikov, I.; Whiteson, S.; and Mudigonda, P. K. 2022. In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35: 12169–12183.

Laumanns, M.; and Ocenasek, J. 2002. Bayesian optimization algorithms for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, 298–307. Springer.

Lin, B.; Feiyang, Y.; and Zhang, Y. 2021. A closer look at loss weighting in multi-task learning.

Liu, B.; Feng, Y.; Stone, P.; and Liu, Q. 2024. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36.

Liu, B.; Liu, X.; Jin, X.; Stone, P.; and Liu, Q. 2021a. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34: 18878–18890.

Liu, L.; Li, Y.; Kuang, Z.; Xue, J.; Chen, Y.; Yang, W.; Liao, Q.; and Zhang, W. 2021b. Towards impartial multi-task learning. iclr.

Liu, S.; Johns, E.; and Davison, A. J. 2019. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1871–1880.

Liu, S.; and Vicente, L. N. 2021. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. *Annals of Operations Research*, 1–30.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, 3730–3738.

Mahapatra, D.; Dong, C.; Chen, Y.; and Momma, M. 2023. Multi-label learning to rank through multi-objective optimization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4605–4616.

Maheshwari, G.; and Perrot, M. 2022. Fairgrad: Fairness aware gradient descent. *arXiv preprint arXiv:2206.10923*.

Miettinen, K. 1999. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

Navon, A.; Shamsian, A.; Achituve, I.; Maron, H.; Kawaguchi, K.; Chechik, G.; and Fetaya, E. 2022. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*.

Pardalos, P. M.; Žilinskas, A.; Žilinskas, J.; et al. 2017. *Non-convex multi-objective optimization*. Springer.

Sener, O.; and Koltun, V. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, 746–760. Springer.

Tanabe, H.; Fukuda, E. H.; and Yamashita, N. 2019. Proximal gradient methods for multiobjective optimization and their applications. *Computational Optimization and Applications*, 72: 339–361.

Xiao, P.; Ban, H.; and Ji, K. 2024. Direction-oriented multi-objective learning: Simple and provable stochastic algorithms. *Advances in Neural Information Processing Systems*, 36.

Xin, D.; Ghorbani, B.; Gilmer, J.; Garg, A.; and Firat, O. 2022. Do current multi-task optimization methods in deep learning even help? *Advances in neural information processing systems*, 35: 13597–13609.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33: 5824–5836.

Zhang, Q.; and Li, H. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6): 712–731.

Zhou, S.; Zhang, W.; Jiang, J.; Zhong, W.; Gu, J.; and Zhu, W. 2022. On the convergence of stochastic multi-objective gradient manipulation and beyond. *Advances in Neural Information Processing Systems*, 35: 38103–38115.