LEVERAGING VARIABLE SPARSITY TO REFINE PARETO STATIONARITY IN MULTI-OBJECTIVE OPTIMIZATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Gradient-based multi-objective optimization (MOO) is essential in modern machine learning, with applications in e.g., multi-task learning, federated learning, algorithmic fairness and reinforcement learning. In this work, we first reveal some limitations of Pareto stationarity, a widely accepted first-order condition for Pareto optimality, in the presence of sparse function-variable structures. Next, to account for such sparsity, we propose a novel solution concept termed Refined Pareto Stationarity (RPS), which we prove is always sandwiched between Pareto optimality and Pareto stationarity. We give an efficient partitioning algorithm to automatically mine the function-variable dependency and substantially trim non-optimal Pareto stationary solutions. Then, we show that gradient-based descent algorithms in MOO can be enhanced with our refined partitioning. In particular, we propose Multiple Gradient Descent Algorithm with Refined Partition (RP-MGDA) as an example method that converges to RPS, while still enjoying a similar per-step complexity and convergence rate. Lastly, we validate our approach through experiments on both synthetic examples and realistic application scenarios where distinct function-variable dependency structures appear. Our results highlight the importance of exploiting function-variable structure in gradient-based MOO, and provide a seamless enhancement to existing approaches.

1 INTRODUCTION

Multi-objective optimization (MOO) aims to optimize several objective functions simultaneously over shared parameters, seeking solutions where no objective can be improved without worsening another. Many modern machine learning applications naturally involve multiple objectives. For example, in fair machine learning, one needs to balance model utility with various fairness notions, while in multi-task learning, models are trained across different tasks with shared representations. Similarly, federated learning and reinforcement learning often involve optimizing the performance of multiple agents or entities distributed across environments. The solutions to such problems are known as Pareto optimal (PO) solutions, representing a balance across the competing objectives. However, finding PO solutions becomes challenging in complex (*e.g.* high-dimensional) problems.

Among various approaches for solving MOO problems, one that has gained significant attention recently is the Multiple Gradient Descent Algorithm (MGDA) (Mukai 1980; Fliege and Svaiter 2000; Désidéri 2012). MGDA extends classical gradient descent to optimize multiple objectives simultaneously. Unlike traditional linear scalarization, which requires predetermined weights for each objective, MGDA directly optimizes the vector-valued problem without such weighting. At each iteration, MGDA identifies a common descent direction that improves all objectives, a feature not guaranteed by scalarization methods. MGDA's ability to converge to a Pareto stationary solution, which serves as a first-order necessary condition for Pareto optimality, has made it widely applicable in areas like multi-task learning (Sener and Koltun 2018), federated learning (Hu et al. 2022), and generative modeling (Albuquerque et al. 2019). While Pareto stationarity provides a solid theoretical foundation (as shown in Lemma 1 below), it can fall short in complex, especially sparse settings, creating room for a more refined approach, as we illustrate in Section 4.

In this work, we reveal that Pareto stationarity (PS), while necessary for Pareto optimality, is often insufficient. Even in convex settings, PS can frequently lead to sub-optimal PO solutions, especially when sparse function-variable dependencies cause strict convexity to fail. To address this, we propose

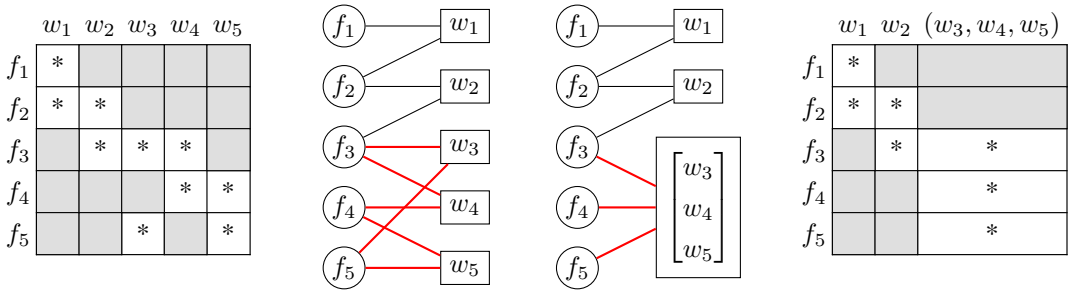

Figure 1: A simple illustration of our main idea. From left to right: adjacency matrix showing the function-variable dependency structure; the underlying bipartite graph with a cycle highlighted in red; merging the variables in the cycle to a subset and contracting the graph; adjacency matrix showing the new dependency structure. The process is iterated until the graph becomes acyclic.

Refined Pareto Stationarity (RPS), which leverages sparsity to identify a valid partition of variables and refines PS with respect to this partition. We prove that RPS is (strictly) sandwiched between PS and Pareto optimality (PO), making it a more useful and practical goal. Additionally, under *partial* strict convexity, we prove RPS reduces exactly to Pareto optimality, whereas the widely-used PS does not. These advantages are illustrated through intuitive examples and validated experimentally.

Since RPS is defined with respect to a refined partition, we propose an efficient partitioning procedure (REFINED_PARTITION) to automatically discover, if needed, the correct refined partition of variables, based on an input adjacency matrix encoding function-variable dependencies. Figure 1 illustrates our main idea through a simple example. More importantly, our results enable practitioners to enhance existing gradient-based descent algorithms for MOO by incorporating this partitioning routine and the refined stationary condition RPS. Specifically, we introduce Multiple Gradient Descent Algorithm with Refined Partition (RP-MGDA) to exploit these advances and we formally prove its convergence.

We summarize our contributions below, with further context provided in the next section:

- We present illustrative examples to reveal some limitations of Pareto stationarity, and motivate the need for a *valid* partitioning of variables when function-variable dependency is sparse.
- We propose RPS, a novel solution concept that refines Pareto stationarity, together with an efficient partitioning procedure REFINED_PARTITION to automatically discover the refined partition.
- We introduce RP-MGDA, an example of enhancing gradient-based descent algorithms in MOO by incorporating refined partitioning. We establish the convergence and complexity of RP-MGDA.
- We validate the effectiveness of our approach through experiments on synthetic and benchmark datasets, showcasing its application to ML scenarios with diverse dependency structures.

2 RELATED WORKS

Multi-objective optimization (MOO) and Pareto solution have been extensively studied, with traditional approaches such as evolutionary algorithms (Deb et al. 2002). However, these techniques face challenges when applied to modern machine learning problems, which are often better addressed by gradient-based methods. Consequently, in this work we focus on gradient-based MOO.

Gradient-based MOO. Gradient-based MOO involves optimizing multiple objectives using gradient information. A foundational algorithm in this regard is MGDA (Mukai 1980; Fliege and Svaiter 2000; Désidéri 2012), which computes a common descent direction for all objectives by solving for the minimum-norm vector in the convex hull of gradients. Fliege et al. (2019) offered a detailed convergence analysis of MGDA, while subsequent works have also extended classical single-objective methods to the multi-objective setting, including Newton's method (Fliege et al. 2009), (accelerated) proximal gradient (Tanabe et al. 2019; Tanabe et al. 2023), conditional gradient (Assunção et al. 2021), and subgradient method (O. Montonen and Mäkelä 2018), among others. Another important line of research focuses on stochastic variants of MGDA, e.g. Mercier et al. (2018), due to their practical relevance in machine learning, particularly for mini-batch training of deep neural networks.

Liu and Vicente (2021) introduced SMG as a direct stochastic counterpart of MGDA, albeit with a biased estimate of the common descent direction. Subsequent works of Zhou et al. (2022) and Fernando et al. (2023) track historical information to reduce bias, while Chen et al. (2023) and Xiao et al. (2023) propose unbiased estimate of the update direction through double sampling.

MOO in Multi-task Learning (MTL). MTL aims to train a common model that can solve multiple, potentially related tasks. Sener and Koltun (2018) first framed MTL as a multi-objective optimization problem and addressed it using MGDA. Since then, numerous works in MTL have approached this problem within the general MOO framework, treating task losses as objectives and proposing novel gradient aggregation schemes to mitigate task conflicts while addressing various aspects of MTL (e.g., Lin et al. 2019). PCGrad (Yu et al. 2020) projects each gradient onto the normal plane of other conflicting gradients and aggregates them. CAGrad (Liu et al. 2021a) balances minimizing the average task loss and improving the worst-performing objective by constraining the search region for the common direction in MGDA. Nash-MTL (Navon et al. 2022) formulates MTL as a bargaining game to weigh the gradients. I-MTL (Liu et al. 2021b) separately searches for the magnitude and direction of the common gradient. FairGrad (Ban and Ji 2024) proposes a new gradient aggregation scheme to ensure α-fairness in MTL.

MOO in Fair machine learning (FML). FML is inherently a multi-objective problem, due to the trade-off between utility and various notions of fairness that need not be compatible with each other (Kleinberg et al. 2017). A common strategy in FML is to linearly combine the multiple objectives through scalar weights (Kamishima et al. 2011), corresponding to the well-known scalarization technique in MOO, while later work, such as Padh et al. (2021), applies MGDA to handle these competing objectives more directly.

The works mentioned above either 1) adopt the general MOO formulation that treats all learnable parameters as a single variable $\mathbf{w} \in \mathbb{R}^d$, thereby disregarding any underlying function-variable structure, or 2) are confined to the MTL setting presented in Sener and Koltun (2018), whose variable partition scheme can be viewed as a special case of our more general framework. To our best knowledge, our work is the first to systematically investigate and refine Pareto stationary solutions in gradient-based MOO, by taking into account the function-variable structure. Our results are complementary to the above approaches and potentially integrable with many of them.

3 Preliminaries

In this section, we briefly review the concepts of Pareto optimality and Pareto stationarity in multi-objective optimization. We then introduce the widely used MGDA method.

3.1 Multi-Objective Minimization (MOO)

In mathematical terms, a Multi-Objective Minimization (MOO) problem can be written as[1]

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathbf{f}(\mathbf{w}), \quad \text{where } \mathbf{f}(\mathbf{w}) := (f_1(\mathbf{w}), f_2(\mathbf{w}), \ldots, f_m(\mathbf{w})) \tag{1}$$

and the minimum is defined w.r.t. the *partial* ordering[2] :

$$\mathbf{f}(\mathbf{w}) \leq \mathbf{f}(\mathbf{z}) \iff \forall i = 1, \ldots, m,\ f_i(\mathbf{w}) \leq f_i(\mathbf{z}). \tag{2}$$

Unlike single objective optimization, with multiple objectives it is possible that

$$\mathbf{f}(\mathbf{w}) \not\leq \mathbf{f}(\mathbf{z}) \text{ and } \mathbf{f}(\mathbf{z}) \not\leq \mathbf{f}(\mathbf{w}), \tag{3}$$

in which case we say $\mathbf{w}$ and $\mathbf{z}$ are not comparable. As a result, there is usually a set of solutions that are all optimal (*a.k.a. Pareto Optimal*) for a given MOO problem, whose objective values form the *Pareto front*.

[1]We focus on minimization in this work, while maximization can be treated similarly.

[2]We remind that algebraic operations such as $\leq$ and $+$, when applied to a vector with another vector *or scalar*, are always performed component-wise.

3.2 Pareto Optimality and Pareto Stationarity

Definition 1 (Pareto Optimality)**.** *We call* $\mathbf{w}_*$ *a* Pareto optimal *solution of* (1) *if its objective value* $\mathbf{f}(\mathbf{w}_*)$ *is a minimum element w.r.t. the partial ordering in* (2)*; equivalently,*

$$\forall \mathbf{w},\ \mathbf{f}(\mathbf{w}) \leq \mathbf{f}(\mathbf{w}_*) \implies \mathbf{f}(\mathbf{w}) = \mathbf{f}(\mathbf{w}_*). \tag{4}$$

In other words, it is not possible to improve *any* component objective in $\mathbf{f}(\mathbf{w}_*)$ without compromising *some* other objective.

Definition 2 (Weakly Pareto Optimality)**.** *We call* $\mathbf{w}_*$ *a* weakly *Pareto optimal solution if there does not exist any* $\mathbf{w}$ *such that* $\mathbf{f}(\mathbf{w}) < \mathbf{f}(\mathbf{w}_*)$*,* i.e.*, it is not possible to improve* all *component objectives in* $\mathbf{f}(\mathbf{w}_*)$ *simultaneously.*

Clearly, any Pareto optimal solution is also weakly Pareto optimal but the converse may not hold.

Next, we recall the concept of *Pareto Sationarity* (also referred to as *Pareto Criticality*), which is the first order necessary condition for Pareto optimality.

Definition 3 (Pareto Stationarity)**.** *We call* $\mathbf{w}_*$ *Pareto stationary iff*

$$\mathbf{0} \in \operatorname{conv}\{\nabla f_1(\mathbf{w}_*), \cdots, \nabla f_m(\mathbf{w}_*)\}, \tag{5}$$

i.e.*, there exists some* $\boldsymbol{\lambda} \geq 0$ *such that* $\sum_i \lambda_i = 1$ *and* $\sum_i \lambda_i \nabla f_i(\mathbf{w}^*) = \mathbf{0}$.

The relevance of Pareto stationarity is captured in the following lemma:

Lemma 1 (*e.g.*, Mukai 1980, Theorem 1)**.** *Any Pareto optimal solution is Pareto stationary. Conversely, if all functions are convex (resp., strictly convex), then any Pareto stationary solution is weakly Pareto optimal (resp., Pareto optimal).*

3.3 Multiple Gradient Descent Algorithm (MGDA)

MGDA has been independently proposed by Mukai (1980) and Fliege and Svaiter (2000) and Désidéri (2012) as a gradient-based method to find Pareto stationary solutions of an MOO problem. It has gained notable attention in machine learning in recent years, largely because of its gradient-based nature, in contrast to traditional MOO techniques. Compared with the scalarization technique (*i.e.*, linearly combining different objectives), MGDA removes the need to pre-determine any weight vector. More importantly, MGDA is guaranteed to improve all objectives of any starting solution while scalarization cannot (Mukai 1980).

In each iteration, MGDA seeks a new solution that minimizes the maximum change among all objectives, i.e.,

$$\tilde{\mathbf{w}}^{t+1} = \operatorname*{argmin}_{\mathbf{w}} \max_{\boldsymbol{\lambda} \in \Delta} \boldsymbol{\lambda}^\top (\mathbf{f}(\mathbf{w}) - \mathbf{f}(\mathbf{w}^t)), \tag{6}$$

where Δ is the standard simplex. Next, we upper bound $\mathbf{f}(\mathbf{w})$ by the usual quadratic approximation:

$$\min_{\mathbf{w}} \max_{\boldsymbol{\lambda} \in \Delta} \boldsymbol{\lambda}^\top J_{\mathbf{f}}^\top(\mathbf{w}^t)(\mathbf{w} - \mathbf{w}^t) + \tfrac{1}{2\eta}\|\mathbf{w} - \mathbf{w}^t\|^2. \tag{7}$$

where $J_{\mathbf{f}} = [\nabla f_1, \ldots, \nabla f_m] \in \mathbb{R}^{d \times m}$ is the Jacobian and $\eta > 0$ is the step size. This approximation (7) is (strongly) convex in $\mathbf{w}$ (even for nonconvex $\mathbf{f}$) and concave in $\boldsymbol{\lambda}$, and hence strong duality holds. Swapping $\min$ with $\max$ in (7) and setting the derivative w.r.t. $\mathbf{w}$ to $\mathbf{0}$, we deduce the following "minimum-norm" direction:

$$\mathbf{d}^t = J_{\mathbf{f}}(\mathbf{w}^t)\boldsymbol{\lambda}_*^t, \ \text{ where } \ \boldsymbol{\lambda}_*^t = \operatorname*{argmin}_{\boldsymbol{\lambda} \in \Delta} \|J_{\mathbf{f}}(\mathbf{w}^t)\boldsymbol{\lambda}\|^2. \tag{8}$$

MGDA then performs the update along the direction $\mathbf{d}^t$:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \mathbf{d}^t. \tag{9}$$

Common stopping criteria include $\|\mathbf{d}^t\| \leq \texttt{tol}$ for some tolerance `tol` or by capping at a maximum number of iterations.

When the step size η is sufficiently small (so that (7) is indeed an upper bound on (6)), the MGDA update (9) *simultaneously* decreases all objectives (seen easily by the feasible choice $\mathbf{w} = \mathbf{w}_t$ in (7)). It is clear that the fixed points of MGDA are exactly Pareto stationary solutions. In fact, Fliege et al. (2019) proved that the minimum norm $\|\mathbf{d}^t\|$ converges to 0 at rate $O(1/\sqrt{t})$, matching the well-known result in single objective optimization.

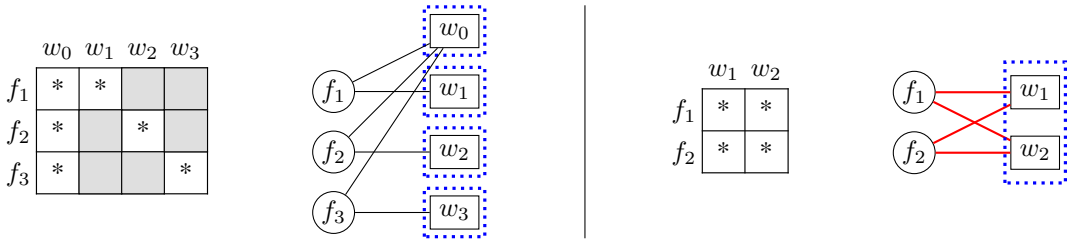

Figure 2: Illustration of Example 1 and Example 2. Left: function-variable dependency structure of Example 1. There is no cycle in the underlying bipartite graph so our method partitions each variable into its own block. Right: function-variable dependency structure of Example 2. There is a cycle (highlighted in red) in the underlying bipartite graph so our method merges both variables into a single block. In both examples, MGDA treats all variables as a single block.

4 Illustrative Examples

We construct two examples that illustrate the various definitions above and reveal some subtle issues about Pareto stationarity and the (coordinate-wise) MGDA update.

Example 1. *Let* $f_1(\mathbf{w}) = w_0^2 + w_1^2$, $f_2(\mathbf{w}) = w_0^2 + w_2^2$ *and* $f_3(\mathbf{w}) = w_0^2 + w_3^2$. *Clearly, all component functions are convex. There is a unique Pareto optimal solution* $\mathbf{w}_\star = (0, 0, 0, 0)$, *but there are infinitely many Pareto stationary (in fact, weakly Pareto optimal) solutions:*

$$\{(0, 0, w, z), (0, w, 0, z), (0, w, z, 0) : w, z \in \mathbb{R}\}. \tag{10}$$

It is clear the MGDA update (8)-(9) *could get trapped at any of the Pareto stationary solutions. However, if we update the variables separately, the coordinate-wise MGDA converges to the unique Pareto optimal solution. See Figure 2 for a pictorial illustration.*

We note that Example 1 does not contradict Lemma 1, since none of the component functions is *strictly* convex. In general, strict convexity, or even convexity, is rarely satisfied in deep learning.

It is then tempting to *always* partition the variables and update them separately using MGDA (*i.e.*, a coordinate-wise MGDA). However, the following example shows this idea may also lead us astray.

Example 2. *Let* $f_1(\mathbf{w}) = (w_1 - w_2)^2 + w_1^2$ *and* $f_2(\mathbf{w}) = (w_1 - w_2)^2 + w_2^2$. *Clearly, both functions are strictly convex. There is a unique Pareto stationary (and hence Pareto optimal) solution* $\mathbf{w}_\star = (0, 0)$. *However, if we apply MGDA to each variable separately, it may converge to any point on the diagonal* $\{(w, w) : w \in \mathbb{R}\}$. *On the other hand, the vanilla MGDA (treating both variables as a whole) converges to the unique Pareto stationary solution. See Figure 2 for a pictorial illustration.*

In the next section, by leveraging on the (sparse) function-variable dependency structure, we prove that it is possible to identify a *valid* partition of variables and define refined Pareto stationary solutions that are sandwiched by Pareto optimal solutions and Pareto stationary solutions.

5 Refined Pareto Stationary Solutions

Motivated by the aforementioned examples, we now consider partitioning the variables in the MOO problem (1). Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a partition of $[d] := \{1, \ldots, d\}$. For a subset P of variables, we use $\mathbf{f}^P$ to denote the set of functions f_i that depend on some variable in P.

Definition 4 (Generalized Pareto Stationarity)**.** *We call* $\mathbf{w}_*$ *Pareto stationary w.r.t. the partition* $\mathcal{P}$ *iff*

$$\mathbf{0} \in \prod_{j=1}^{k} \operatorname{conv}\{\nabla_{\mathbf{w}_{P_j}} \mathbf{f}^{P_j}(\mathbf{w}_*)\} \text{ (the Cartesian product)}, \tag{11}$$

i.e., *for all* $j \in [k]$, *there exists* $\boldsymbol{\lambda}^j \in \Delta$ *such that* $\sum_i \lambda_i^j \nabla_{\mathbf{w}_{P_j}} f_i(\mathbf{w}^*) = \mathbf{0}$ *and* $\lambda_i^j = 0$ *if* f_i *does not depend on the variables in* P_j.

When $\mathcal{P} = [d]$ is the trivial partition, our definition reduces to Pareto stationarity in Definition 3. The following lemma slightly generalizes one direction of Lemma 1.

Lemma 2. *Any Pareto optimal solution is Pareto stationary w.r.t. any partition.*

(All proofs can be found in Appendix A.) However, the converse part of Lemma 2 under (strict) convexity is harder to extend: we need to find the right partition first. Indeed, Example 2 already showed Pareto stationarity w.r.t. the finest partition is not equal to Pareto optimality, or put it slightly differently, Pareto stationary w.r.t. an arbitrary *finer* partition does not necessarily imply Pareto stationary w.r.t. a *coarser* partition[3].

Consider the adjacency matrix $A \in \{0,1\}^{m \times d}$ of a bipartite graph, with

$$A_{ij} = \begin{cases} 1, & \text{if } f_i \text{ depends on } w_j \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

Our next goal is to find the (finest) partition $\mathcal{P}$ of variables (*i.e.*, columns of A) so that Pareto stationary w.r.t. $\mathcal{P}$ is also Pareto stationary (w.r.t. the trivial partition). This is particularly useful when the dependency matrix A is sparse, since it enables us to largely reduce the set of Pareto stationary solutions (without missing any Pareto optimal ones); see Example 1.

The main idea is to find cycles in the underlying bipartite graph, merge and contract the resulting variables, and repeat until the graph becomes acyclic; see Figure 1 for an illustration. We recall from Cormen et al. (2009, Exercise 22.4-3, p. 615) that detecting a cycle in an undirected graph using depth-first search costs $O(m+d)$ (the number of nodes) for our bipartite graph. Since merging the variables in a cycle reduces the graph size by at least 1, the process can repeat at most $O(d)$ times. Therefore, the entire process, which we denote as REFINED_PARTITION (A), costs $O(d(m+d))$ to output a partition $\mathcal{P} := \{P_1, \ldots, P_d\}$ of the variables.

We formally justify the above partition procedure in:

Theorem 1. *Let* $\mathcal{P} =$ REFINED_PARTITION(A) *where* A *is defined in* (12). *Then, any* $\mathbf{w}_*$ *that is Pareto stationary w.r.t.* $\mathcal{P}$*, which we call refined Pareto stationary from now, is Pareto stationary.*

We are finally ready to provide a converse of Lemma 2, under (partial) strict convexity.

Theorem 2. *Let* $\mathcal{P} = \{P_1, \ldots, P_k\}$ *be returned by* REFINED_PARTITION. *Suppose each* f_i *is strictly convex in* $\mathbf{w}_{I_i}$ *where* $I_i \subseteq \mathcal{P}$ *is the set of blocks of variables that* f_i *depends on. Then, any refined Pareto stationary solution (w.r.t.* $\mathcal{P}$*) is Pareto optimal.*

To summarize, refined Pareto stationarity (based on our partition procedure) is sandwiched between Pareto optimality and Pareto stationarity; Figure 3 illustrates the relationships between these concepts. In particular, we have identified a *valid* partition to sharpen Lemma 1 in its full extent. Importantly, applying our refined Pareto stationarity to Example 1 and Example 2 successfully identifies the unique Pareto optimal solution while eliminating infinitely many spurious Pareto stationary solutions.

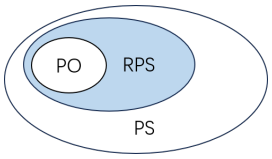

Figure 3: Euler diagram for Pareto Optimal (PO), Refined Pareto Stationary (RPS) and Pareto Stationary (PS).

6 IMPROVING EXISTING ALGORITHMS WITH THE REFINED PARTITION

The refined Pareto stationarity introduced in the previous section is a solution concept. Broadly, it can be used to enhance existing gradient-based *descent* algorithms for MOO. Indeed, we can simply cycle through or randomly pick the blocks of variables identified by REFINED_PARTITION, and update each block with a chosen gradient-based descent algorithm in turn. The resulting algorithm is guaranteed to be descending, *i.e.*, it improves all objectives in each step.

Interestingly, for certain MOO algorithms we can even afford to update all blocks *simultaneously*. We demonstrate this possibility and its practical value by modifying and analyzing the MGDA algorithm from Section 3.3, enabling it to converge to a refined Pareto stationary solution. The full algorithm, which we call RP-MGDA, is summarized in Algorithm 1. The algorithm again terminates when $\|\mathbf{d}\|_2 < \texttt{tol}$, or when a maximum number of iterations is achieved.

[3]The subtlety lies on the requirement that each $\boldsymbol{\lambda}^j \in \Delta$, *i.e.*, we cannot set any of them to $\mathbf{0}$.

Algorithm 1: Multiple Gradient Descent Algorithm with Refined Partition (RP-MGDA)

Input: function-variable dependency structure A, functions $\mathbf{f}$, initializer $\mathbf{w} \in \mathbb{R}^d$, learning rate $\boldsymbol{\eta}$

1 $(P_1, \ldots, P_k) = \text{REFINED_PARTITION}(A)$ `// partition variables based on A`

2 **for** $t = 1, 2, \ldots$ **do**

3 **parfor** $j = 1, 2, \ldots, k$ **do** `// in parallel`

4   $J^j(\mathbf{w}) \leftarrow \nabla_{\mathbf{w}_{P_j}} \mathbf{f}^{P_j}(\mathbf{w})$ `// compute gradients`

5   $\boldsymbol{\lambda}^j \leftarrow \operatorname{argmin}_{\boldsymbol{\lambda} \in \Delta} \|J^j(\mathbf{w})\boldsymbol{\lambda}\|^2$ `// solve the dual subproblem`

6   $\mathbf{d}^t_{P_j} \leftarrow J^j(\mathbf{w})\boldsymbol{\lambda}^j$ `// minimum-norm direction in` P_j

7   $\mathbf{w}_{P_j} \leftarrow \mathbf{w}_{P_j} - \eta_j \cdot \mathbf{d}^t_{P_j}$ `// update`

Output: final weights $\mathbf{w}$

Following Fliege et al. (2019) we prove that the direction $\mathbf{d}$ in RP-MGDA (Algorithm 1) converges to $\mathbf{0}$ at the rate $O(1/\sqrt{t})$, and hence matching the usual result for single objective optimization.

Theorem 3. *Suppose the functions f_i are bounded from below and Lipschitz smooth with constant L_i. Set $\eta_j \equiv \eta \leq \min_i \frac{1}{L_i}$. Then, $\min_{1 \leq t \leq T} \|\mathbf{d}^t\| \leq O(\frac{1}{\sqrt{T}})$.*

Combining Theorem 1 and Theorem 3 we immediately have the following result:

Corollary 1. *Assuming RP-MGDA converges, then it converges to a Pareto stationary solution at the rate $O(1/\sqrt{t})$.*

A few remarks regarding RP-MGDA are in order: **1)** When the sublevel set $\{\mathbf{w} : \mathbf{f}(\mathbf{w}) \leq \mathbf{f}(\mathbf{w}^1)\}$ is compact, RP-MGDA admits a limit point that is refined Pareto stationary and hence Pareto stationary, thanks to its descending property. **2)** In practice we do not need to know the Lipschitz constants L_i: a standard line search procedure on η to test the quadratic upper bound (7) suffices to achieve the same convergence rate. **3)** Theorem 1 and hence Corollary 1 continue to hold if we replace the partition $\mathcal{P}$ with any coarser partition (such as the trivial partition $[d]$). Thus, our results strictly generalize those about MGDA (*e.g.*, Fliege et al. 2019).

Complexity. We emphasize the inner loop of Algorithm 1 is executed in parallel, as they correspond to updating different blocks of $\mathbf{w}$ *simultaneously*. We have analyzed the complexity of the subroutine REFINED_PARTITION in the previous section, and we note that this one-time cost can often be avoided if one knows the domain application well, see our experiments in Section 7 for examples. Barring the one-time overhead in line 1, Algorithm 1 is theoretically cheaper than the vanilla MGDA, since the most costly step in line 5 (solving the dual subproblem) often has superlinear dependence on m (the number of objectives) and hence dividing it into blocks of smaller sizes reduces the runtime.

7 EXPERIMENTS

In this section, we present experiments comparing RP-MGDA and MGDA across both synthetic examples and realistic benchmark datasets. These experiments explore varying variable dependency structures and convexity assumptions, demonstrating RP-MGDA's effectiveness and versatility. More experimental details and discussions can be found in Appendix B.

7.1 SYNTHETIC EXAMPLES

We examine two settings. First, quadratic MOO problems with randomly generated adjacency matrices, focusing on the behaviour of REFINED_PARTITION and the effectiveness of RP-MGDA. Second, in Appendix B.1, we consider a two-objective convex problem, where we visualize and compare solution fronts, observing how often MGDA converges to sub-optimal PS solutions.

Randomly generated sparse adjacency matrix. We randomly generate $m \times d$ adjacency matrices with varying degrees of sparsity to set up the function-variable dependency structures of quadratic MOO problems with m objectives and d input variables. The sparsity of function-variable dependencies is controlled by a 'density' parameter, which determines the percentage of non-zero entries in the adjacency matrix. We use our proposed REFINED_PARTITION procedure to automatically

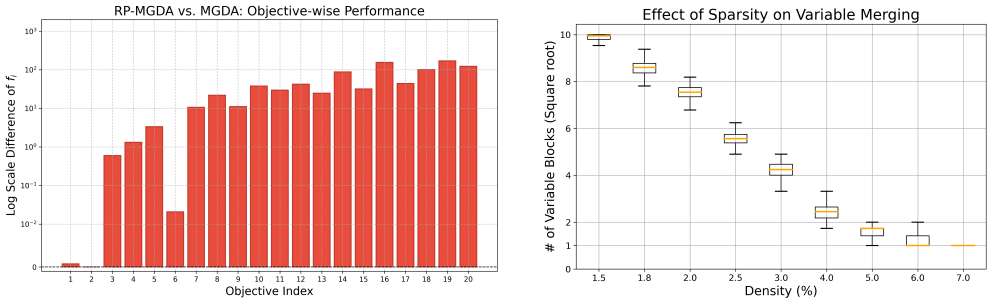

Figure 4: Experiments on randomly generated sparse adjacency matrices. **Left**: [20×100 matrix]. The bar chart compares the solution from MGDA with that post-processed by RP-MGDA. Red bars indicate the magnitude by which RP-MGDA outperforms MGDA in objective value, while green bars (if any) indicate the opposite. **Right**: [100×100 matrices]. The impact of adjacency matrix density on variable merging in REFINED_PARTITION, where y-axis shows the final number of variable blocks returned. A value of 1 indicates all variables are merged into a single block, and a value of 100 (represented as 10 on the square root scale) indicates that all variables are fully separated.

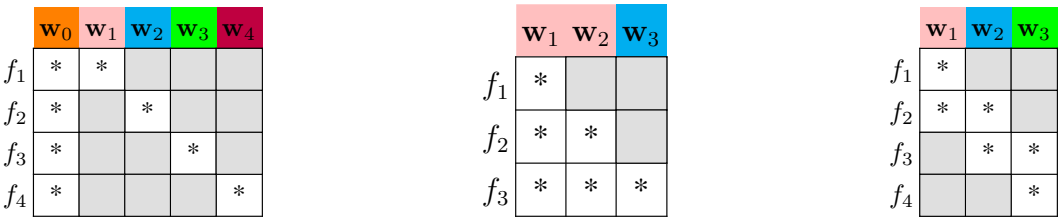

Figure 5: Different function-variable dependency structures for experiments on benchmark datasets. Left: Personalized federated learning; Middle: Hierarchical classification; Right: Multi-objective learning with partial information.

generate refined partitions, especially for larger adjacency matrices (*e.g.* 100×100) where manual partitioning becomes cumbersome. Details of the setup can be found in Appendix B.2.

Results. From Figure 4 (Left) we observe that by post-processing the outputs of MGDA, *i.e.*, using them to initialize RP-MGDA, we can improve the solutions to Pareto dominate the original ones. Conversely, we point out that MGDA cannot improve the stationary solutions of RP-MGDA, due to Theorem 1. In the demonstration with 100×100 matrix (Figure 4, Right), we note that the number of final blocks returned by REFINED_PARTITION decreases as the *density* increases, starting at 1.5%, where all variables are separated, and eventually merging all variables into a single block when the density reaches 7%. We also empirically verified Theorem 3 in Appendix B.2.

7.2 ML BENCHMARK DATASETS

We examine three problems with different function-variable dependency structures derived from real-world application scenarios. These examples also serve as a 'proof of concept' for applying refined partitioning to realistic ML problems. The structures and their corresponding refined partitions are illustrated in Figure 5.

7.2.1 PERSONALIZED FEDERATED LEARNING ('MTL' STRUCTURE)

Setup. We consider a personalized federated learning setting with $m = 4$ clients, each holding distinct non-i.i.d. data D_i sampled from the training dataset (MNIST/CIFAR-10). The objectives f_i are defined as empirical cross-entropy losses over the clients' data distributions. As in standard federated learning, we have a shared global model with learnable parameters $\mathbf{w}_0$. To handle data heterogeneity and enable personalization, instead of directly feeding inputs (e.g., images) into the global model $\mathbf{w}_0$, each client i has a local personalized network, parameterized by $\mathbf{w}_i$, which extracts

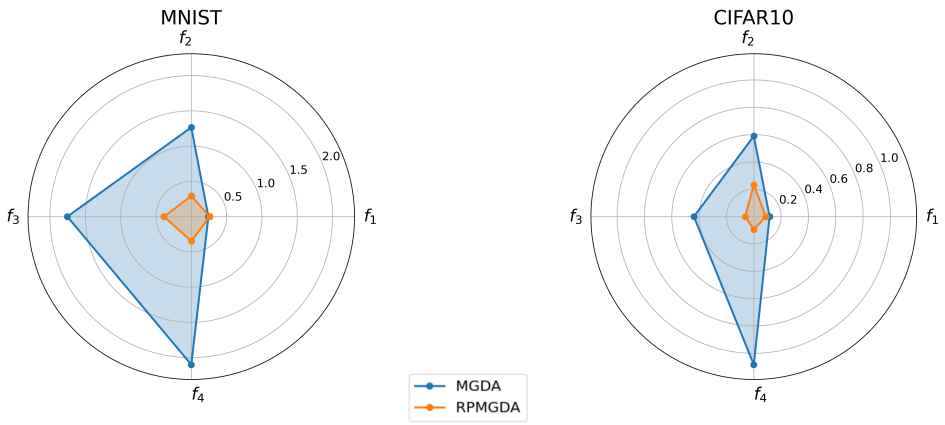

Figure 6: Radar graph of final cross-entropy losses f_i in the personalized federated learning setting, using MGDA and RP-MGDA respectively. Lower is better.

client-specific representations. These representations are then forwarded to the global model for prediction (*e.g.*, Liang et al. 2020). Formally, $f_i(\mathbf{w}_0, \mathbf{w}_i) = \mathbb{E}_{\mathbf{x}\sim D_i}\left[\ell(\mathrm{NN}(\mathrm{NN}(\mathbf{x};\mathbf{w}_i);\mathbf{w}_0))\right]$, thus having the dependency structure illustrated in Figure 5 (Left), which is also commonly seen in multi-task learning (MTL), see *e.g.*, Sener and Koltun (2018).

Refined Partition. For this 'MTL' dependency structure, the optimal refined partition is $\{\{\mathbf{w}_0\}, \{\mathbf{w}_1\}, \ldots, \{\mathbf{w}_m\}\}$. Consequently, at each iteration, RP-MGDA updates $\mathbf{w}_0$ by descending along the minimum-norm direction, while $\mathbf{w}_i$ (for $i \in [m]$) is updated with gradient descent *simultaneously*. This partitioning (a) simplifies the dual sub-problem by reducing its dimensionality and (b) allows for more flexible updates to the personalized networks.

Results. As shown in Figure 6 and Appendix B.3.2, we observe that RP-MGDA converges faster and reaches better solutions than MGDA, particularly for f_2, f_3 and f_4, while for f_1, the performances are similar. Further details are provided in Appendix B.3.

7.2.2 Hierarchical classification ('Ladder' Structure)

Setup. We consider a multi-objective hierarchical classification problem with $m = 3$ levels of objectives that predict image categories, from coarse to fine labels, on the CIFAR-10 dataset, using a network structure similar to Branch-CNN (Zhu and Bain 2017). Here, $\mathbf{w}_1, \mathbf{w}_2$ and $\mathbf{w}_3$ represent the network weights of feature extraction layers $1, 2$ and 3, respectively. The three objectives, f_1, f_2 and f_3, are all cross-entropy losses but defined for different prediction tasks. The first task is binary classification, categorizing images as 'living' or 'non-living,' using only the representations from feature extraction layer 1, and thus depending solely on $\mathbf{w}_1$.[4] The second task classifies images into four categories: 'ground vehicle,' 'non-ground vehicle,' 'land animal,' or 'non-land animal,' using representations from layers 1 and 2, and thus depending on $\mathbf{w}_1$ and $\mathbf{w}_2$. The third task classifies images into the original CIFAR-10 categories, using representations from all three layers, and thus depending on $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$. This hierarchical structure allows for progressively more complex predictions as we move from coarse to fine labels.

Refined Partition. The corresponding dependency structure in Figure 5 (Middle) features an optimal refined partition of $\{\{\mathbf{w}_1\ \mathbf{w}_2\}, \{\mathbf{w}_3\}\}$, where RP-MGDA optimizes $\mathbf{w}_3$ using gradient descent and the block $\{\mathbf{w}_1\ \mathbf{w}_2\}$ using MGDA. This partitioning should be advantageous for Objective 3, as it allows for more flexible updates to $\mathbf{w}_3$ compared to MGDA.

Results. Indeed, we observe that the training loss curves for Objectives 1 and 2 are nearly indistinguishable; however, RP-MGDA demonstrates faster and better convergence for Objective 3, as illustrated in Figure 7.

[4] After 'branching out,' an additional linear classification layer is used for the final prediction, which can either be fixed or updated separately. We omit these 'off-ramps' from the discussion, as they do not affect the dependency structure of the feature extraction layers, which is the focus here.

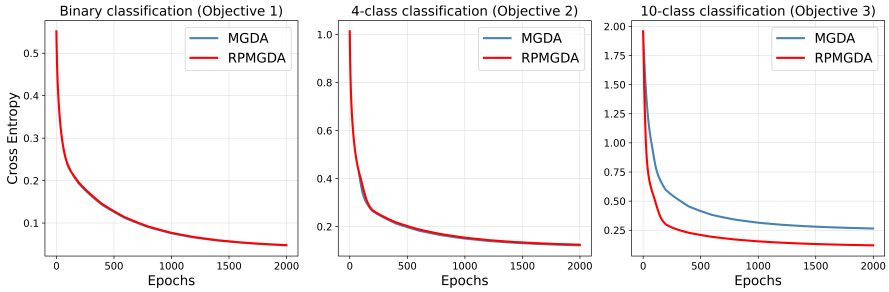

Figure 7: Loss curves of MGDA and RP-MGDA for the three objectives in hierarchical classification.

7.2.3 Multi-objective learning with partial information ('Chain' Structure)

Setup. We simulate a scenario inspired by vertical federated learning (Liu et al. 2024) where each party holds different feature sets for the same instances, akin to multiple institutions possessing various attributes for the same users. In particular, we explore the setting where each party only has access to consecutive variables, *e.g.*, a company in a supply chain having only adjacent upstream and downstream data. In this context, clients with heterogeneous data collaborate to learn a stronger model without directly sharing sensitive user data. Specifically, we consider an $m = 4$-objective learning problem on the California housing dataset (Pace and Barry 1997), where we employ a linear model to predict the median house value. The original 8 features are grouped into 3 sets[5], each associated with linear regression weights $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$. The objectives f_i represent the mean-squared error (MSE) for different parties, all trained on the same dataset but with each objective depending on different subsets of features, as illustrated in Figure 5 (Right).

Refined Partition. For this 'chain' dependency structure, the optimal refined partition is $\{\{\mathbf{w}_1\}, \{\mathbf{w}_2\}, \{\mathbf{w}_3\}\}$, allowing RP-MGDA to apply block-wise MGDA to $\mathbf{w}_1, \mathbf{w}_2$ and $\mathbf{w}_3$ separately. This results in simpler dual sub-problems and improved convergence.

Results. As shown in Figure 8, RP-MGDA demonstrates greater stability and lower average losses across the 20 trials. We also observe that the stationary solutions of MGDA can be improved by post-processing with RP-MGDA, in 18 out of 20 trials. Additionally, the conditions of Theorem 2 are met in this setup, ensuring that RP-MGDA achieves Pareto optimality.

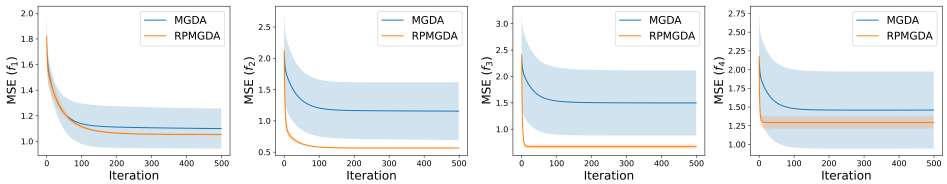

Figure 8: We calculate the mean and standard deviation of the 4 losses across 20 ramdom trials (different seeds) for both MGDA and RP-MGDA, and plot the means, with shaded regions representing the standard deviation.

8 Conclusion and Discussion

In this work, we introduced Refined Pareto Stationarity (RPS) with an efficient partitioning procedure (Refined_Partition) to address limitations of Pareto stationarity in multi-objective optimization (MOO). We proposed RP-MGDA, a novel MOO algorithm based on refined partitioning that converges to RPS, supported by both theoretical and empirical justification. Future work may explore applying our idea to other MOO algorithms and demonstrate the superior performance on other real problems with (sparse) dependency structures.

[5]Details on feature grouping are provided in Appendix B.5.

REFERENCES

Albuquerque, I., J. Monteiro, T. Doan, B. Considine, T. Falk, and I. Mitliagkas (2019). "Multi-objective training of Generative Adversarial Networks with multiple discriminators". In: *Proceedings of the 36th International Conference on Machine Learning*.

Assunção, P., O. P. Ferreira, and L. Prudente (2021). "Conditional gradient method for multiobjective optimization". *Computational Optimization and Applications*, vol. 78, no. 3, pp. 741–768.

Ban, H. and K. Ji (2024). "Fair Resource Allocation in Multi-Task Learning". In: *Proceedings of the 41st International Conference on Machine Learning*, pp. 2715–2731.

Chen, L., H. Fernando, Y. Ying, and T. Chen (2023). "Three-Way Trade-Off in Multi-Objective Learning: Optimization, Generalization and Conflict-Avoidance". In: *Advances in Neural Information Processing Systems*, pp. 70045–70093.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2009). "Introduction to Algorithms". 3rd. The MIT Press.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197.

Désidéri, J.-A. (2012). "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization". *Comptes Rendus Mathematique*, vol. 350, no. 5, pp. 313–318.

Fernando, H. D., H. Shen, M. Liu, S. Chaudhury, K. Murugesan, and T. Chen (2023). "Mitigating Gradient Bias in Multi-objective Learning: A Provably Convergent Approach". In: *International Conference on Learning Representations*.

Fliege, J., L. G. Drummond, and B. F. Svaiter (2009). "Newton's method for multiobjective optimization". *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 602–626.

Fliege, J. and B. F. Svaiter (2000). "Steepest descent methods for multicriteria optimization". *Mathematical Methods of Operations Research*, vol. 51, no. 3, pp. 479–494.

Fliege, J., A. I. F. Vaz, and L. N. Vicente (2019). "Complexity of gradient descent for multiobjective optimization". *Optimization Methods and Software*, vol. 34, no. 5, pp. 949–959.

Hu, Z., K. Shaloudegi, G. Zhang, and Y. Yu (2022). "Federated learning meets multi-objective optimization". *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2039–2051.

Kamishima, T., S. Akaho, and J. Sakuma (2011). "Fairness-aware Learning through Regularization Approach". In: *IEEE 11th International Conference on Data Mining Workshops*, pp. 643–650.

Kleinberg, J., S. Mullainathan, and M. Raghavan (2017). "Inherent trade-offs in the fair determination of risk scores". In: *Innovations in Theoretical Computer Science (ITCS)*, 43:1–43:23.

Liang, P. P., T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency (2020). "Think locally, act globally: Federated learning with local and global representations". In: *NeurIPS Workshop on Federated Learning*.

Lin, X., H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong (2019). "Pareto Multi-Task Learning". In: *Advances in Neural Information Processing Systems*.

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

Liu, L., Y. Li, Z. Kuang, J.-H. Xue, Y. Chen, W. Yang, Q. Liao, and W. Zhang (2021b). "Towards Impartial Multi-task Learning". In: *International Conference on Learning Representations*.

Liu, S. and L. N. Vicente (2021). "The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning". *Annals of Operations Research*, vol. 339, pp. 1119–1148.

Liu, Y., Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang (2024). "Vertical federated learning: Concepts, advances, and challenges". *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3615–3634.

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017). "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282.

Mercier, Q., F. Poirion, and J.-A. Désidéri (2018). "A stochastic multiple gradient descent algorithm". *European Journal of Operational Research*, vol. 271, no. 3, pp. 808–817.

Mukai, H. (1980). "Algorithms for multicriterion optimization". *IEEE Transactions on Automatic Control*, vol. 25, no. 2, pp. 177–186.

Navon, A., A. Shamsian, I. Achituve, H. Maron, K. Kawaguchi, G. Chechik, and E. Fetaya (2022). "Multi-Task Learning as a Bargaining Game". In: *Proceedings of the 39th International Conference on Machine Learning*.

Nesterov, Y. (2018). "Lectures on Convex Optimization". 2nd. Springer.

O. Montonen, N. K. and M. M. Mäkelä (2018). "Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization". *Optimization*, vol. 67, no. 1, pp. 139–158.

Pace, R. K. and R. Barry (1997). "California Housing Dataset". Accessed through scikit-learn library.

Padh, K., D. Antognini, E. Lejal-Glaude, B. Faltings, and C. Musat (2021). "Addressing fairness in classification with a model-agnostic multi-objective algorithm". In: *Uncertainty in Artificial Intelligence*, pp. 600–609.

Sener, O. and V. Koltun (2018). "Multi-Task Learning as Multi-Objective Optimization". In: *Advances in Neural Information Processing Systems*.

Tanabe, H., E. H. Fukuda, and N. Yamashita (2019). "Proximal gradient methods for multiobjective optimization and their applications". *Computational Optimization and Applications*, vol. 72, pp. 339–361.

– (2023). "An accelerated proximal gradient method for multiobjective optimization". *Computational Optimization and Applications*, vol. 86, pp. 421–455.

Xiao, P., H. Ban, and K. Ji (2023). "Direction-oriented Multi-objective Learning: Simple and Provable Stochastic Algorithms". In: *Advances in Neural Information Processing Systems*, pp. 4509–4533.

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

Zhou, S., W. Zhang, J. Jiang, W. Zhong, J. GU, and W. Zhu (2022). "On the Convergence of Stochastic Multi-Objective Gradient Manipulation and Beyond". In: *Advances in Neural Information Processing Systems*.

Zhu, X. and M. Bain (2017). "B-CNN: branch convolutional neural network for hierarchical classification".