UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Multi-Task Learning: Effective Weighting Algorithms and Model Parameterization

by Feiyang Ye

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Prof. Ivor Tsang and Prof. Yu Zhang

University of Technology Sydney
Faculty of Engineering and Information Technology

October 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Feiyang Ye, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note:
Signature removed prior to publication.

[Feiyang Ye]

DATE: 7th October, 2024

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my principal supervisor at University of Technology Sydney, Professor Ivor W. Tsang, and my principal supervisor at Southern University of Science and Technology, Professor Yu Zhang. Without their patience and encouragement, I would never complete this Ph.D. journey.

I would like to thank many friends at the Southern University of Science and Technology and the University of Technology Sydney who have supported me. I want to thank Dr. Baijiong Lin, and Dr. Zhixiong Yue for discussing both research and life. It was a memorable time to discuss research together at SUSTech. I would like to thank Dr. Yueming Lyu for his support and discussion at A*STAR. I would thank Xuehao Wang, Dr. Weisen Jiang, Dr. Pengxin Guo, and Dr. Jinjing Zhu for the discussion and for working together. I may miss mentioning many others, but I thank them for their help. Their company and help bring me to the end of this journey.

Abstract

Multi-Task Learning (MTL) is a widely used paradigm for training shared models across multiple tasks. It improves data efficiency by jointly training all tasks, allowing the model to leverage shared information. However, directly optimizing the mean of the losses across tasks can lead to imbalanced performance due to conflicts in task objectives. These conflicts arise because different tasks often compete for the same model capacity, which can degrade the performance of certain tasks. While many MTL models attempt to address this issue by employing loss weighting techniques or through model parameterization, the challenge of effectively balancing these competing objectives remains unresolved. Thus, the question of how to design more effective task weighting methods or model parameterization methods for MTL continues to be a key challenge in the field.

This thesis aims to address this challenge by focusing on optimization algorithms and model parameterization specifically tailored for MTL. It seeks to answer three critical research questions that arise in realistic multi-task learning scenarios: 1) How can we effectively learn task weights that ensure balanced training and performance across all tasks; 2) How can we effectively learn the model parameters in black-box settings where explicit task gradient information is not available. 3) How can we exploit task similarities to design parameterized models that reduce competition and conflict between tasks?

To solve problem 1), this thesis introduces a Multi-Objective Bi-Level Optimization framework, which alternates between learning task-specific model parameters and

task weights. This method not only balances the performance across tasks but also dynamically adjusts weights to address conflicts as training progresses. To solve problem 2), this thesis presents a black-box multi-objective learning algorithm, which optimizes model parameters in scenarios where task gradient information is unavailable. To solve problem 3), this thesis presents a neural ODE-based model, which leverages the inherent similarities between tasks to reduce competition for shared parameters. This model enables smoother coordination between tasks, leading to improved overall performance.

In conclusion, this thesis presents a series of novel optimization techniques, task-weighting algorithms, and parameterized models, contributing to the development of more effective multi-task learning systems.

List of Publications

Related to the Thesis :

1. **Feiyang Ye**[*], Baijiong Lin[*], Zhixiong Yue, Pengxin Guo, Qiao Xiao, and Yu Zhang. Multi-Objective Meta Learning, In *Advances in Neural Information Processing Systems* (NeurIPS), 2021.

2. **Feiyang Ye**, Baijiong Lin, Zhixiong Yue, Yu Zhang, and Ivor W. Tsang. Multi-Objective Meta-Learning. In *Artificial Intelligence* (AIJ), 2024.

3. **Feiyang Ye**, Baijiong Lin, Xiaofeng Cao, Yu Zhang, and Ivor W. Tsang. A First-Order Multi-Gradient Algorithm for Multi-Objective Bi-Level Optimization. In *European Conference on Artificial Intelligence* (ECAI), 2024.

4. **Feiyang Ye**[*], Yueming Lyu[*], Xuehao Wang, Yu Zhang, and Ivor W. Tsang. Adaptive Stochastic Gradient Algorithm for Black-box Multi-Objective Learning. In *International Conference on Learning Representations* (ICLR), 2024.

5. **Feiyang Ye**[*], Xuehao Wang[*], Yu Zhang, and Ivor W. Tsang. Multi-Task Learning via Time-aware NODE. In *International Joint Conference on Artificial Intelligence* (IJCAI), 2023.

Others :

6. Baijiong Lin, **Feiyang Ye**, Yu Zhang, and Ivor W. Tsang. Reasonable Effectiveness of Random Weighting: A Litmus Test for Multi-Task Learning, *Transactions on Machine Learning Research* (TMLR), 2022.

7. **Feiyang Ye**[*], Jianghan Bao[*], and Yu Zhang. Partially-Labeled Domain Generalization via Multi-Dimensional Domain Adaptation. In *International Joint Conference on Neural Networks* (IJCNN), 2023.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Background

Humans have the remarkable ability to learn multiple tasks simultaneously and often apply knowledge from one task to assist in learning another. For example, skills developed in playing tennis can aid in learning squash, and vice versa. This natural ability to transfer learning across domains has inspired the development of Multi-Task Learning (MTL) in machine learning. MTL seeks to replicate this human capability by training multiple related tasks together, allowing the knowledge gained from one task to enhance the learning of others. The goal of MTL [10, 146] is to enhance the generalization and performance across all tasks being learned.

MTL was initially motivated by the challenge of data sparsity, where individual tasks often lacked sufficient labeled data for effective training. By jointly training tasks and sharing their labeled data, MTL acts as a form of data augmentation, producing more reliable and accurate models. This approach not only enables the reuse of existing knowledge but also reduces the costs associated with manual labeling and data collection. In the modern era of big data, especially in domains like computer vision and Natural Language Processing (NLP), deep MTL models have consistently outperformed their single-task counterparts. This success is attributed to MTL's ability to harness the data from multiple tasks, learning more generalized and robust representations that improve knowledge sharing between tasks. Moreover, by learning shared representations, MTL

helps to prevent overfitting and enhances task performance, particularly in data-scarce scenarios.

MTL is also closely related to several other machine learning paradigms, including transfer learning [90], multi-label learning, and multi-output regression [144]. Although MTL shares some similarities with transfer learning, there are important distinctions. In MTL, all tasks are treated equally, with the aim of improving the performance of every task involved. In contrast, transfer learning prioritizes a single target task, using knowledge from source tasks to boost the performance of that specific task. Similarly, multi-label learning and multi-output regression, where each data point is associated with multiple labels, can be viewed as special cases of MTL. In these scenarios, each label is treated as a separate task, but the tasks consistently share the same data throughout training and testing.

Over the past decades, MTL has attracted much attention in the artificial intelligence and machine learning communities. Many MTL models have been devised, and many MTL applications in other areas have been exploited. As deep learning models are becoming larger and larger to solve complex problems, MTL becomes attractive since by sharing parameters across all the tasks and training all the tasks jointly.

In this thesis, we focus on advancing MTL by addressing several critical questions, including how to optimize task weights and how to parameterize models effectively in MTL settings. By exploring these challenges, we aim to contribute to the development of more efficient and powerful multi-task learning systems.

1.2 Motivation

MTL has the potential to enhance the quality of the learned representations [118], thereby benefiting each individual task. However, simultaneously learning multiple tasks presents a challenging optimization problem due to the presence of multiple objectives [121]. The most commonly used MTL objective is the average loss across all tasks. However, even when this average loss accurately reflects the true objective (unlike situations where the focus is on a single task, as in the door/doorknob analogy), directly optimizing the average loss can sometimes lead to suboptimal outcomes. For instance, the optimizer may struggle to make meaningful progress, resulting in a significant decline in learning performance. One known explanation of this phenomenon is the conflicting gradients [121]: gradients from different tasks may point in different directions, so directly optimizing the average loss can be quite detrimental to a specific task's performance.

To build an effective MTL model, recent advances in MTL mainly come from three aspects: weighting algorithm, parameterized model design, and task grouping. Each approach tackles the underlying issue of task imbalance in different ways.

Weighting algorithms are based on the idea that suboptimal performance in MTL is often due to gradient conflicts. When some tasks generate gradients with significantly larger magnitudes, they dominate the parameter updates, causing the model to become biased toward those tasks at the expense of others. This imbalance degrades the performance of the underrepresented tasks. To mitigate this issue, recent research has proposed various optimization strategies to balance task losses or gradients more effectively [16, 66, 70, 89, 103, 141]. These methods aim to ensure that no single task disproportionately influences the shared parameters, thereby improving overall model performance.

Model parameterization design focuses on creating or searching for more sophisticated parameter-sharing architectures to address the imbalanced performance between tasks [43, 78, 117]. These approaches often involve dynamically adjusting the amount of shared knowledge between tasks or designing specialized layers for certain tasks. While these methods can be effective at reducing task conflicts, they often come with trade-offs, such as increased model complexity, higher inference times, and a larger number of parameters.

Task grouping methods attempt to group tasks based on their similarities, allowing more effective sharing of information between related tasks while isolating unrelated tasks [56, 131]. These methods typically rely on pre-computed task similarity measures, which can be computationally expensive and may also increase the total model size, further complicating deployment and scalability.

In this thesis, we mainly focus on the first two categories. We try to propose more effective weighting algorithms and parameterization models to overcome the imbalance performance in MTL.

1.3 Research Questions and Objectives

The main research objectives of this thesis is to develop a set of effective weighting algorithms and model parameterization methods for multi-task learning. Our research aims to answer the following three research questions:

Research Question 1: *How to design an effective weighting method for multi-task learning?*

Task weighting, which assigns weights on the including tasks during training, significantly matters the performance of MTL. As a result, there has been an explosive interest in it. Existing task weighting methods compute the task weights only based on training losses or corresponding gradients. Therefore, in optimizing these task weights, these methods do not consider the generalization performance of the learned task weights. Therefore, it motivates us to design an algorithm to learn the task weights more effectively by considering its generalization performance.

Research Question 2: *How to design an effective balancing method for multi-task learning in black-box scenarios?*

Recently, many large models such as large language models (LLMs) [23, 96, 139] are released in the service and are only allowed for access with APIs [8]. In such scenarios, users can only query the large models without accessing gradients to accomplish tasks of interest [114, 116]. For Multi-Task Learning (MTL) in these black-box settings, traditional approaches that rely on gradients or stochastic gradients with respect to model parameters are no longer feasible. As a result, existing task-weighting methods in MTL are not applicable. This challenge motivates us to develop a new algorithm specifically designed for black-box MTL that can effectively balance tasks without requiring gradient information, while still achieving strong performance across tasks.

Research Question 3: *How to use task similarity to design a parameterized model to improve performance in multi-task learning?*

Among all the architectures for deep MTL, the Hard Parameter Sharing (HPS) architecture, which typically shares one feature extractor among tasks and, after that, has a task-specific output layer or head for each task, is the earliest and the most used one. Though it is simple, several works [6, 85, 118, 143] point out that the HPS architecture is effective in improving the performance of each task. However, due to the use of a shared feature extractor to obtain a shared feature representation among tasks, the HPS architecture often faces the problem of competing for shared parameters among tasks during the training process, specifically in the form of gradient conflict which often leads to performance degradation for some tasks [70, 141]. Previous studies commonly use the weighting approach [16, 66, 70, 89, 103, 141] to directly address this problem

and did not study the similarity between all tasks. This motivates us to model the task similarity directly and use it to help train the model.

1.4 Research Contributions

Contribution 1. We proposed a multi-objective bi-level optimization framework for task weighting for training the MTL model. Two reliable algorithms [132–134] with comprehensive theoretical results were proposed to solve the proposed framework.

- This study provides a Multi-Objective Bi-Level Optimization (MOBLO) framework for multi-task learning, which alternately learns model parameters and task weights to solve the MTL problem.
- This study first introduces a nested gradient-based MOBLO method called MOML and provides asymptotic and non-asymptotic convergent results for the proposed MOML method.
- This study introduces a first-order gradient-based MOBLO method called FORUM, which can solve the MOBLO problem more efficiently. A non-asymptotic convergent result for the proposed FORUM method is provided.
- Extensive experiments demonstrate the effectiveness and efficiency of the proposed framework and methods. In particular, the proposed FORUM method achieves state-of-the-art performance on multiple benchmark datasets under the setting of multi-task learning.

Contribution 2. An effective weighting approach for black-box MTL problem [135].

- This study provides a novel ASMG algorithm for black-box multi-objective optimization, which can be used to solve the black-box multi-task learning problem. To the best of our knowledge, this work is the first to design a stochastic gradient algorithm for black-box MOO with a theoretical convergence guarantee.
- This study explicitly provides the connection of the Pareto optimal and stationary conditions between the original MOO and the corresponding Gaussian-smoothed MOO. Moreover, we prove the convergence rate for the proposed ASMG algorithm in both the convex case and the non-convex case.

- Empirically, the proposed ASMG algorithm achieves state-of-the-art performance on black-box multi-task learning problems.

Contribution 3. An effective Neural Neural Ordinary Differential Equation (NODE) based parameterized model [136] for MTL training.

- This study is the first to model feature transformations in MTL from the perspective of dynamic flow and propose the NORMAL method to learn task positions that represent the task-specific feature transformations in the dynamic flow.
- The NORMAL method outperforms multiple methods on four MTL benchmark datasets, including the Office-31, Office-Home, NYUv2, and CelebA datasets.
- The task positions learned by the NORMAL method can be used to evaluate the relevance of different tasks, which could improve the interpretability of the proposed NORMAL method

Contribution Clarification. The contributions listed above are based on some publications where the first two authors share equal contributions. We clarify the authors' specific contributions to these works. In [133], Feiyang Ye's main contributions are methodology, theoretical analysis, writing the original draft, and experiments. Baijiong Lin's main contributions are writing the original draft and experiments. In [135], Feiyang Ye's main contributions are methodology, theoretical analysis, and writing the original draft. Yueming Lyu's main contributions are methodology and theoretical analysis. In [136], Feiyang Ye's main contributions are methodology, and writing the original draft. Xuehao Wang's main contributions are experiments, and writing the original draft.

1.5 Thesis Outline

The six chapters of this thesis are summarized as follows:

- CHAPTER 2 presents the preliminary and literature review of multi-task learning. In particular, we first review some concepts of multi-objective optimization since gradient multi-objective optimization algorithms are widely used in multi-task learning. Then, we introduce the previous work on loss balancing and gradient balancing methods for multi-task learning.

- CHAPTER 3 presents presents a Multi-Objective Bi-Level Optimization (MOBLO) framework for MTL. Two reliable algorithms with comprehensive theoretical results were proposed to solve the proposed framework and achieve our first research objective.
- CHAPTER 4 introduces an effective weighting algorithm for black-box multi-task learning, termed ASMG, designed to address our second research objective. Through extensive experiments and theoretical analysis, we demonstrate the effectiveness of the proposed ASMG method.
- CHAPTER 5 presents an effective model parameterization approach for multi-task learning, named NORMAL, which is designed to achieve our third research objective. Experimental results demonstrate that the proposed NORMAL method can explicitly represent task relationships and significantly enhance performance in multi-task learning scenarios.
- CHAPTER 6 summarises the main contributions of this thesis and discusses future work.

CHAPTER 2

PRELIMINARY AND LITERATURE REVIEW

In this chapter, to provide a comprehensive understanding of multi-task learning, we begin with an introduction to the preliminary concepts in MTL. We provide the problem statement of MTL in Section 2.1. We provide the review of a representative MTL method called the Multiple Gradient Descent Algorithm in Section 2.2. Then, we provide a literature review for weighting algorithms and model parameterization of MTL in Section 2.3 and Section 2.4, respectively.

2.1 Problem Statement

We first recall the definition of MTL [146].

Definition 2.1. Given m learning tasks $\{\mathcal{T}\}_{i=1}^{m}$ where some or all tasks are related, the goal of multi-task learning (MTL) is to jointly learn these tasks. By leveraging knowledge from other tasks, MTL aims to enhance the performance of each individual task $\mathcal{T}_i$.

Based on the definition of MTL, we now provide a detailed problem statement. Given m learning tasks $\{\mathcal{T}_i\}_{i=1}^{m}$, task i has its corresponding dataset $\mathcal{D}_i$. Then the MTL model usually contains two parts of parameters: task-shared parameters θ and task-specific parameters $\{\phi_i\}_{i=1}^{m}$. The feature extractor $f_\theta(x) : \mathcal{X} \to \mathbb{R}^q$, which maps a sample $x \in \mathcal{X}$ into a q-dimensional feature space, is parameterized by task-shared parameters θ. Then, the i-th task-specific output module parameterized by task-specific parameters ϕ_i outputs the prediction as $h_{\phi_i}(f_\theta(x))$. Let $\mathcal{L}_i(\cdot,\cdot)$ denote the loss function for task i (e.g., the cross-

entropy loss for classification tasks). Then for the standard MTL problem with **Equal Weights (EW)** method, MTL aims to learn all the parameters (i.e., $\theta, \phi_1, \phi_2, ..., \phi_m$) by minimizing the total loss as

$$\mathscr{L} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}\mathscr{L}_i(y_i^j, h_{\phi_i}(f_\theta(x_i^j))), \tag{2.1}$$

where n_i denotes the number of samples for task i, x_i^j denotes the jth sample in task i, and y_i^j denotes the label of x_i^j.

2.2 Multiple Gradient Descent Algorithm

Directly optimizing the average loss can sometimes result in suboptimal performance, as the optimizer may struggle to make meaningful progress. This can cause a significant decline in learning efficiency and overall model performance. Therefore, an alternative view is to consider MTL as a Multi-Objective Optimization (MOO) problem [103]. By considering each task as one optimization objective. A m-tasks MTL problem can be viewed as the following MOO problem

$$\min_{\{\theta,\phi_1,\phi_2,...,\phi_m\}} \big(F_1(\theta,\phi_1),...,F_m(\theta,\phi_m)\big)^\top, \tag{2.2}$$

where $F_i(\theta,\phi_i) = \frac{1}{n_i}\sum_{j=1}^{n_i}\mathscr{L}_i(y_i^j, h_{\phi_i}(f_\theta(x_i^j)))$. Since most algorithms proposed in this thesis and many MTL algorithms are designed based on this view. We first review the MOO problem.

MOO aims to solve multiple objectives simultaneously and its goal is to find the Pareto-optimal solutions. MOO algorithms can be broadly divided into three categories: population-based [1], evolutionary-based [12, 147], and gradient-based [22, 83]. In MTL, we mainly focus on the gradient-based category, because this approach can easily be integrated into gradient-based machine learning models. One notable gradient-based MOO method is the Multiple Gradient Descent Algorithm (MGDA) [22] algorithm, which serves as a representative approach in this field. The MGDA algorithm employs a quadratic programming problem to determine the optimal direction for gradient updates during each training iteration. By doing so, it ensures that all objectives decrease simultaneously. Compared with the widely-used linear scalarization approach which linearly combines multiple objectives to a single objective, MGDA and its variants [31, 149] have shown their superiority in many learning problems such as multi-task

learning [103] and reinforcement learning [141], especially when some objectives are conflicting.

We introduce the detailed approach for MGDA to solve an unconstrained multi-objective optimization problem (i.e. $\min_z g(z)$). MGDA finds the minimum-norm point in the convex hull composed by the gradients of multiple objectives. Specifically, it performs the following two steps alternately:

Step 1. Compute the gradients $\nabla_z g_i(z)$ for $i = 1,\ldots,m$, and solve the following quadratic programming problem

$$\min_{\gamma} \left\| \sum_{i=1}^{m} \gamma_i \nabla_z g_i(z) \right\|^2 \quad \text{s.t. } \gamma_i \geq 0, \ \sum_{i=1}^{m} \gamma_i = 1, \tag{2.3}$$

to determine the weights γ_i in the current iteration. γ_i can be viewed as a weight for the i-th objective. To solve the problem (2.3), we can use the Frank-Wolfe algorithm [103]. Then, the descent direction searched is computed as $d = \sum_{i=1}^{m} \gamma_i \nabla_z g_i(z)$.

Step 2. If $d = 0$, the MGDA stops. Otherwise, a line step is determined as the largest positive scalar v, with which all objectives are decreasing. Then we update z as $z - vd$ and go to Step 1.

Remark 2.1. *The original MGDA searches the step size to ensure that all the objectives decrease in each iteration. However, this will result in significant computational complexity for learning models with many parameters such as deep neural networks. Therefore, in MTL, similar to [86, 103], a common approach is to use a fixed and small step size in MGDA to reduce the computational complexity.*

2.3 Weighting Algorithms for Multi-task Learning

Loss Balancing Methods. This approach aims to balance different tasks by dynamically adjusting loss weights based on various factors such as learning speed, relative loss values, validation performance, and uncertainty. For instance, Dynamic Weight Averaging (**DWA**) [74] sets the weight for each task by computing the ratio of two consecutive loss values. Uncertainty Weighting (**UW**) [53] uses homoscedastic uncertainty as the basis for task weights, which are updated dynamically through backpropagation. Another method, **IMTL-L** [70], learns task-specific loss weights by ensuring that the scaled loss values across tasks remain similar, thereby achieving a balanced optimization process.

Gradient Balancing Methods. This class of methods focuses on finding an aggregated gradient that balances different tasks in Multi-Task Learning (MTL). For instance, MGDA-UB [103] frames MTL as a multi-objective optimization problem, solving for optimal task weights at each iteration using MGDA [22]. This approach finds a common descent direction by solving a quadratic programming problem. GradNorm [16] aims to learn loss weights by ensuring that the gradient magnitudes across tasks are balanced. GradDrop [17] addresses gradient conflicts by identifying inconsistencies in the signs of gradient values across tasks and resolves these conflicts by masking out conflicting gradient components. PCGrad [141] handles gradient conflicts by projecting each task's gradient onto the normal plane of another task's gradient, particularly when a conflict is detected through negative cosine similarities. GradVac [126] builds upon PCGrad, projecting gradients more adaptively where cosine similarities between task gradients can vary. IMTL-G [70] finds an aggregated gradient that maintains equal projection lengths across all tasks' gradients, ensuring balance in updates. Similarly, CAGrad [66] solves an optimization problem at every iteration to find a gradient that minimizes all task losses. RotoGrad [49] takes a different approach by homogenizing both gradient magnitudes and directions, using a learnable rotation matrix to adjust the gradient direction for each task and computing weights that equalize gradient magnitudes. Finally, Nash-MTL [89] formulates the gradient aggregation problem as a Nash bargaining game, allowing for a fair resolution of conflicts between task gradients.

While many weighting algorithms exist for multi-task learning, most of these methods determine task weights solely based on training losses or their corresponding gradients, overlooking the critical gap between training loss and generalization loss. To tackle this issue, we propose a meta-learning approach that leverages the division of the entire training dataset to effectively learn task weights using the validation dataset. Additionally, prior research in multi-task learning has predominantly focused on white-box scenarios. However, with the rise of large language models (LLMs) and similar large-scale models, many are now deployed as services accessible only through APIs, causing black-box scenarios. Consequently, developing effective weighting algorithms for black-box multi-task learning has become increasingly important.

2.4 Model Parameterization in Multi-task Learning

Hard and Soft Parameter Sharing Methods. In hard parameter sharing methods, one of the most widely used architectures is the multi-head hard sharing model [10]. In

this approach, the initial layers of the network are shared across all tasks, while the later layers are task-specific. Although task weighting methods are typically applied in this hard parameter sharing setting, the fixed structure of the shared network can limit the model's capacity, potentially leading to suboptimal solutions due to insufficient flexibility.

To better capture the relationships between tasks, soft parameter sharing methods have been introduced. For instance, the cross-stitch network [87] linearly combines hidden representations from different tasks, allowing for more adaptive feature sharing. Similarly, the multi-task attention network [74] uses a shared base network with task-specific attention modules, enabling the model to learn both shared and task-specific representations through the attention mechanism. Another approach is the neural discriminative dimensionality reduction layer [38], which automatically fuses features from different tasks at each layer, enhancing the model's ability to learn task relationships dynamically. While soft parameter sharing methods provide greater flexibility, they often involve maintaining multiple full-size networks, which can lead to overfitting on smaller datasets and result in large model sizes that are impractical for deployment.

Task Routing. Rather than relying solely on either soft sharing architecture or hard sharing architecture, task routing methods and other adaptive network models have been introduced to offer greater flexibility. For example, Multi-Agent Reinforcement Learning [99] enables the network to dynamically self-organize based on the input, while the Task Routing Layer [112] allows a single model to handle multiple tasks by using task-specific mask matrices. However, these routing-based methods typically operate within a single network, which can limit their expressive power.

To address this limitation, several more adaptive methods have been proposed. Adaptive feature aggregation layers [18] introduce a dynamic mechanism where each task can determine how much knowledge to share with other tasks. Another approach is an adaptive sharing method [117], which learns a task-specific policy that selectively determines which layers should be executed for each task. Similarly, branched multi-task networks [120] use task affinity scores to dynamically construct network branches.

These approaches dynamically adjust feature or parameter sharing between tasks, resulting in better generalization compared to manually designed architectures. However, they also introduce additional computational overhead, as determining task relatedness often requires extra processing.

Architecture Learning. Instead of manually designing deep neural network architectures, Neural Architecture Search (NAS) [69, 93] offers an automated approach to discovering high-performing architectures. Several works have applied NAS to automatically search for optimal architectures in Multi-Task Learning (MTL), aiming to enhance overall performance across tasks.

For example, [78] dynamically expands a multi-layer network to form a tree-like structure, where tasks with similar characteristics are grouped into the same branches. [62] leverages an evolutionary architecture search algorithm to identify blueprints and modules, which are then assembled into a cohesive multi-task learning (MTL) network. Additionally, [37] focuses on searching for inter-task layers that facilitate more effective feature fusion across tasks. A differentiable architecture search algorithm has also been introduced in [43] to learn branching blocks, creating tree-structured networks tailored specifically for MTL. Moreover, NAS techniques have been employed to optimize branching architectures automatically [9] for the encoder in MTL networks, particularly under resource constraints.

Unlike traditional hard and soft parameter-sharing methods, which rely on designing different models to implicitly capture task relationships, we introduce a novel NODE-based multi-task learning model that explicitly models the relationships between tasks.

2.5 Summary

In this chapter, we begin by introducing the problem statement of multi-task learning (MTL), followed by a review of the representative MTL method, MGDA. We also provide a comprehensive literature review of weighting algorithms and model parameterization techniques in MTL. Weighting algorithms for multi-task learning can be categorized into two main types: loss balancing methods and gradient balancing methods. In contrast to prior studies, our work focuses on weighting algorithms based on a meta-learning approach and addresses black-box scenarios. Regarding model parameterization, MTL methods can be grouped into three categories: hard and soft parameter-sharing methods, task routing, and architecture learning. Unlike traditional hard and soft parameter-sharing approaches, our work explicitly models task relationships to mitigate competition among tasks for shared parameters.

CHAPTER 3

A MULTI-OBJECTIVE BI-LEVEL OPTIMIZATION FRAMEWORK FOR MTL

3.1 Chapter Abstract

This chapter introduces a novel Multi-Objective Bi-Level Optimization (MOBLO) framework for Multi-Task Learning (MTL). In this framework, we optimize the model parameters for the MTL model in the lower-level subproblem, while in the upper-level subproblem, we optimize the ideal task weights. To address the MOBLO problem, we propose an alternating optimization algorithm called MOML, and a more efficient first-order method, FORUM. We provide comprehensive theoretical analysis for both methods and conduct extensive experiments on various MTL benchmark datasets. The results demonstrate the effectiveness of the proposed algorithms in improving MTL performance.

3.2 Preliminary

Bi-Level Optimization. Bi-level optimization is a special kind of optimization, where one problem is nested within another one. It has been recognized as a powerful optimization tool for meta-learning methods [48]. The generic Bi-Level Optimization problem (BLO) is formulated as

$$\min_{\alpha \in \mathcal{A}, \omega \in \mathbb{R}^p} F(\omega, \alpha) \ \text{ s.t. } \ \omega \in \mathcal{S}(\alpha), \tag{3.1}$$

where the function F is called the Upper-Level (UL) objective and $\mathscr{S}(\alpha)$ is the solution set of the Lower-Level (LL) subproblem, and α, ω denote the UL and LL variables, respectively.

One representative category of the BLO method is the ITD-based methods [34, 35, 42] that use approximated hypergradient to optimize the UL variable, which is computed by the automatic differentiation based on the optimization trajectory of the LL variable. Some value-function-based algorithms [67, 72, 109] have been proposed recently to solve BLO by reformulating the original BLO to an equivalent optimization problem with a simpler structure. The value-function-based reformulation strategy naturally yields a first-order algorithm, hence it has high computational efficiency.

Existing BLO methods mostly assume that the UL subproblem is a single-objective optimization problem (i.e. $F : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$) and aim to propose optimization algorithms to maintain convergence properties under different assumptions such as non-convexity [71]. There are only a handful of works for solving Multi-Objective Bi-Level Optimization (MOBLO) problems [21, 51, 107]. However, all those works solve MOBLO via evolutionary algorithms and analyze it from a game theoretic point of view. The MOML [133] method introduced in this thesis is proposed as the first gradient-based MOBLO algorithm. However, MOML needs to calculate the complex Hessian matrix to obtain the hypergradient, causing the computationally inefficient problem. MoCo [31] also employs the ITD-based approach like MOML for hypergradient calculation. It uses a momentum-like gradient approximation approach for hypergradient and a one-step approximation method to update the weights. It has the same inefficiency problem as the MOML method. [140] propose a mini-batch approach to optimize the UL subproblem in the MOBLO. However, it aims to generate weights for a huge number of UL objectives and is different from what we focus on.

3.3 A Multi-Objective Bi-Level Optimization Framework

In MTL, it is common for the including tasks to be competing. If we cannot properly balance these tasks, some tasks might dominate the training process and hurt the performance of other tasks, a phenomenon known as task imbalance. To address the task imbalance, the most widely used method is task weighting, which adaptively assigns weights on the tasks during training to balance their impacts. Existing task weighting methods compute the task weights only based on training losses or corresponding

gradients. They ignore the gap between training loss and generalization loss. To address this problem, based on the split of the entire training dataset, we use a meta-learning approach to effectively learn the task weights in the validation dataset.

Specifically, suppose there are m tasks. The i-th task has a dataset $\mathscr{D}_i$ for model training. Here each $\mathscr{D}_i$ is partitioned into two subsets: the training dataset $\mathscr{D}_i^{tr}$ and the validation dataset $\mathscr{D}_i^{val}$, where $\mathscr{D}_i^{tr}$ is used to train a multi-task model and $\mathscr{D}_i^{val}$ is to measure the performance of the multi-task model on the i-th task. $f(\cdot;\omega)$, the learning function of the multi-task model parameterized by ω, receives data points from the m tasks and outputs predictions. Let $\alpha_i \in [0,1]$ denote the loss weight for the i-th task. The goal is to learn the simplex weight vector $\alpha = (\alpha_1,\ldots,\alpha_m)^\top \in \Delta^{m-1}$ and the model parameter ω. The objective function of the proposed method is formulated as

$$\begin{aligned} &\min_{\alpha\in\Delta^{m-1},\omega} \left(\mathscr{L}_{MTL}(\omega^*(\alpha),\mathscr{D}_1^{val}),\ldots,\mathscr{L}_{MTL}(\omega^*(\alpha),\mathscr{D}_m^{val})\right) \\ &\text{s.t.} \quad \omega^*(\alpha) = \arg\min_{\omega} \sum_{i=1}^{m} \alpha_i \mathscr{L}_{MTL}(\omega,\mathscr{D}_i^{tr}), \end{aligned} \tag{3.2}$$

where $\mathscr{L}_{MTL}(\omega,\mathscr{D}) = \frac{1}{|\mathscr{D}|}\sum_{(\mathbf{x},y)\in\mathscr{D}} \ell(f(\mathbf{x};\omega),y)$ denotes the average loss of $f(\cdot;\omega)$ on a dataset $\mathscr{D}$ with $|\mathscr{D}|$ denoting the size of $\mathscr{D}$ and $\ell(\cdot,\cdot)$ denoting a loss function. In the inner problem of the problem (3.2), when given weights in α, we aim to learn an MTL model to get optimal parameters ω^* on the training dataset and in the outer problem, we expect to update α via minimizing the loss of the trained MTL model with parameters ω^* on the validation dataset of each task.

The problem (3.2) is not a standard BLO problem since it has multiple objectives to optimize in the outer problem. There exist several multi-objective evolutionary algorithms [21, 100, 106] which can be used to solve problem (3.2). However, such methods have a high complexity without convergence guarantee and are not easy to be integrated with gradient-based models such as deep neural networks. Therefore, we need to propose an effective gradient-based method to solve such an optimization problem.

To design effective gradient-based algorithms for solving problem (3.2), we first abstract the problem (3.2) as the following optimization problem:

$$\min_{\alpha\in\mathscr{A},\omega\in\mathbb{R}^p} F(\omega,\alpha) = (F_1(\omega,\alpha),F_2(\omega,\alpha),\ldots,F_m(\omega,\alpha))^T \quad \text{s.t.} \ \ \omega\in\mathscr{S}(\alpha), \tag{3.3}$$

where function $F:\mathbb{R}^p\times\mathbb{R}^n\to\mathbb{R}^m$ is a vector-valued jointly continuous function for the m desired meta-objectives and $\mathscr{A}$ is a nonempty compact subset of $\mathbb{R}^n$. Problem (3.3) is a Multi-Objective Bi-Level Optimization (MOBLO) problem. The goal of solving problem (3.3) is to find the Pareto-optimal solution, which is defined in Appendix A.1.1. In problem

(3.3), $\mathscr{S}(\alpha)$ is defined as the set of optimal solutions to minimize $f(\omega, \alpha)$ w.r.t. ω, i.e.

$$\mathscr{S}(\alpha) = \arg\min_{\omega} f(\omega, \alpha). \tag{3.4}$$

When m equals 1, problem (3.3) reduces to a standard BLO, and hence from this perspective, the proposed MOBLO framework is a generalization of BLO. In problems (3.3) and (3.4), F is the UL subproblem and $f : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$ is the LL subproblem. The LL subproblem can be considered as a constraint for the UL subproblem.

Note that in our work [133, 134], we also called this framework a Multi-Objective Meta-Learning (MOML) framework since we view this framework as a learning-to-learn strategy (a.k.a meta-learning). In MTL, F contains multiple meta-objectives to be achieved for the meta-learner, and f defines the objective function for the training losses of current weighted tasks.

In Section 3.8, we will see the application of the MOBLO framework in different learning problems, including few-shot learning, NAS, and RL.

3.4 A Nested Gradient-Based MOBLO Method

In this section, we devise a general nested gradient-based algorithm to solve the objective function in the proposed MOBLO framework (i.e. problem (3.3)).

3.4.1 Reformulation of MOBLO

We now discuss an important assumption for problem (3.3). Due to the complicated dependency between UL and LL variables, solving it is challenging, especially when optimal solutions of the LL subproblem are not unique.

For a standard BLO with a single objective in the UL subproblem, many studies [25, 35, 104] potentially require that the LL subproblem only admits a unique minimizer ω for every $\alpha \in \mathscr{A}$, which is formally introduced as the Lower-Level Singleton (LLS) condition in [73]. If the LLS condition does not hold, then for a fixed point α_0, it is unclear which $\omega \in S(\alpha_0)$ should be evaluated in the UL subproblem. Thus, this condition can simplify both the optimization process and convergence analyses.

To solve problem (3.3) with multiple objectives in the UL subproblem, the LLS condition is necessary. If not, the MOBLO is even ill-defined [27]. Moreover, since the UL objective F is vector-valued, it is more challenging than standard BLO with a scale-valued F to determine which ω should be evaluated in the UL subproblem. Although we

can select one specific ω by adding some constraints (e.g. choosing the minimum-norm solution), it is not a general solution and would complicate the problem. Thus, in this work, we use the LLS condition to simplify the analyses.

With the LLS condition, the optimal solution for a given α in the LL subproblem is denoted by ω^*, and then problem (3.3) can be reformulated as

$$\min_{\alpha \in \mathcal{A}} \quad \varphi(\alpha) = F\left(\omega^*(\alpha), \alpha\right) \quad \text{s.t.} \quad \omega^*(\alpha) = \arg\min_{\omega} f(\omega, \alpha). \tag{3.5}$$

3.4.2 MOML Algorithm

Here we design a general nested gradient-based optimization algorithm to solve problem (3.5) called MOML. Usually, there is no closed form for the solution $\omega^*(\alpha)$ of the LL subproblem and so it is difficult to optimize the UL subproblem directly. Another approach is to use the optimality condition of the LL subproblem (i.e. $\nabla_\omega f(\omega, \alpha) = 0$) as equality constraints for the UL subproblem in a way similar to [92]. However, this approach only works for LL subproblems with simple forms and cannot work for general learning models.

To solve problem (3.5), we take a strategy similar to the alternating optimization. In the first part of each iteration (corresponding to steps 3-6 in Algorithm 1), we solve the LL subproblem via gradient descent methods. Specifically, with an initialization ω_0 for the LL variable where the iteration index k is omitted for notation simplicity, the solution of the LL subproblem can be updated for T steps as $\omega_{t+1}(\alpha) = \mathcal{T}_t(\omega_t(\alpha), \alpha), t = 1, \ldots, T-1$, where $\mathcal{T}_t$ represents an operator to update ω. Here we consider a first-order gradient descent method for $\mathcal{T}_t$ such as the Stochastic Gradient Descent (SGD) method and $\mathcal{T}_t$ can be formulated explicitly as $\mathcal{T}_t(\omega_t(\alpha), \alpha) = \omega_t(\alpha) - \mu \nabla_\omega f(\omega_t(\alpha), \alpha)$, where $\mu > 0$ denotes the step size and $\nabla_\omega f(\omega_t(\alpha), \alpha)$ denotes the derivative of f w.r.t. ω at $\omega = \omega_t(\alpha)$. In the second part of each iteration (corresponding to steps 7-9 in Algorithm 1), by fixing the value of ω as the current solution $\omega_T(\alpha)$ obtained in the first part, we solve the UL subproblem as

$$\min_{\alpha} \varphi_T(\alpha) = F(\omega_T(\alpha), \alpha). \tag{3.6}$$

Problem (3.6) is an unconstrained Multi-Objective Optimization (MOO) problem and we can use any multi-objective optimization algorithms to solve it. Here we choose gradient-based multi-objective optimization methods as they can be seamlessly integrated into any gradient-based learning framework. There are several gradient-based multi-objective optimization algorithms [22, 94, 119] and they commonly find an appropriate descent direction d for all the objectives in F by aggregating their gradients w.r.t.

α. So such process is denoted by $d = \text{MOOSolver}(\{\nabla_\alpha F_i(\omega_T(\alpha), \alpha)\}_{i=1}^m)$ in Algorithm 1, where the derivative of $F_i(\omega_T(\alpha), \alpha)$ (w.r.t. α) can be computed by automatic differentiation techniques. In this work, we mainly adopt a simple gradient-based MOO method called Multiple Gradient Descent Algorithm (MGDA) [22], whose details are introduced in Section 2.2. In MGDA, the descent direction d for multiple objectives can be found in the convex hull of the gradients of each objective. Therefore, the descent direction can be determined by $d = \sum_{i=1}^m \gamma_{i,k} \nabla_\alpha F_i(\omega_T(\alpha), \alpha)$, where $\sum_{i=1}^m \gamma_{i,k} = 1$ and $\gamma_{i,k} \geq 0$. Here $\gamma_{i,k}$ could be viewed as the weight of the i-th objective in the k-th iteration, but different from the weighted sum algorithm that uses a fixed weight for each objective, MGDA determines the weights $\{\gamma_{i,k}\}$ by minimizing the ℓ_2 norm of the convex hull of the gradients of each objective in each iteration.

Algorithm 1 The MOML Method

Input: numbers of iterations (T, K), step size (μ, ν)

Randomly initialized α_0;
for $t = 1$ **to** K **do**
  Initialize $\omega_0^k(\alpha_k)$;
  for $j = 1$ **to** T **do**
    $\omega_j^k(\alpha_k) \leftarrow \omega_{j-1}^k(\alpha_k) - \mu \nabla_\omega f(\omega_{j-1}^k(\alpha_k), \alpha_k)$;
  end for
  Compute gradients $\nabla_\alpha F_i(\omega_T^k(\alpha_k), \alpha_k)$ for all the i's;
  Compute the gradient as $d(\omega_T^k(\alpha_k), \alpha_k) = \text{MOOSolver}(\{\nabla_\alpha F_i(\omega_T^k(\alpha_k), \alpha_k)\})$;
  $\alpha_{k+1} = \alpha_k - \nu_k d(\omega_T^k(\alpha_k), \alpha_k)$ with a step size ν_k;
end for

The entire algorithm to solve problem (3.5) is shown in Algorithm 1, which, to the best of our knowledge, is the first gradient-based optimization algorithm for solving MOBLO problems.

3.4.3 Convergence Analysis

Algorithm 1 is simple and intuitive, but its convergence properties are unclear. In this section, we provide convergence analyses for the approximated problem (3.6) and Algorithm 1 under certain assumptions.

As an MOBLO, problem (3.5) cannot be reduced to a BLO with a scalar-valued objective function in the upper-level subproblem when using Algorithm 1 to solve it. Therefore, it has different theoretical properties from BLO as we need to focus on the convergence properties of a minimal point set instead of a minimum scalar in the UL subproblem.

We first recall some notions about vector-valued functions. Consider a vector-valued function $g(z):\mathbb{R}^n \to \mathbb{R}^m$ $(m,n \in \mathbb{N}, m \geq 2)$. We denote by Min $g(z)$ the set of all the minimal points of the function $g(z)$. Min $g(z)$ is also called the Pareto frontier or Pareto-optimal set. The corresponding efficient solution or Pareto-optimal solution set of $g(z)$ is denoted by Eff(g(z)). The convexity of vector-valued functions is called the P-convex. The details of these definitions can be found in Appendix A.1.1.

To analyze the convergence properties of problem (3.6), we first make the following assumptions.

Assumption 3.1. *We assume that* (i) *the set* $\mathscr{A}$ *is a nonempty compact subset of* $\mathbb{R}^n$*;* (ii) *the functions* $f(\omega,\alpha)$ *and* $F(\omega,\alpha)$ *are both jointly continuous functions;* (iii) $\arg\min_\omega f(\omega,\alpha)$ *is a singleton for every* $\alpha \in \mathscr{A}$*;* (iv) $\omega^*(\alpha)$ *is uniformly bounded on* $\mathscr{A}$*.*

Note that the third assumption in Assumption 3.1 is the LLS condition introduced in Section 3.4.1 and is widely adopted in BLOs [29, 104]. With Assumption 3.1, we can obtain the following result.

Theorem 3.1. *Suppose Assumption 3.1 holds, then the function* $F(\omega^*(\alpha),\alpha)$ *is continuous w.r.t.* α*.*

The proof of Theorem 3.1 is provided in Appendix A.1.2.1. Because $\mathscr{A}$ is a compact set, Theorem 3.1 implies the existence of solutions. Theorem 3.1 and the uniform convergence of $\omega_T(\alpha)$ can further imply the convergence for the solution of the LL subproblem, which is similar to that of the standard BLO problem [35].

For the convergence of Algorithm 1, we need to analyze minimal point sets of the images of perturbed functions $\varphi_T(\alpha)$ and $\varphi(\alpha)$ which are defined in problems (3.6) and (3.5), respectively. We consider the most natural set convergence under this setting, i.e. the Kuratowski-Painlevé set-convergence. Please refer to these definitions in Appendix A.1.1. Under certain assumptions that are used in analyses of BLO and MOO [35, 80], we have the following convergence results.

Theorem 3.2. *In addition to Assumption 3.1, it is assumed that* (i) *The iterative sequence* $\{\omega_T(\alpha)\}_{t=1}^T$ *converges uniformly to* $\omega^*(\alpha)$ *on* $\mathscr{A}$ *as* $T \to +\infty$*;* (ii) $\mathscr{A}$ *is a convex set;* (iii) φ_T *is P-convex and* φ *is strictly P-convex. Then, the Kuratowski-Painlevé set-convergence of both the minimal point set and efficient solution set in Algorithm 1 holds, i.e.*

$$\text{Min } \varphi_T(\alpha) \to \text{Min } \varphi(\alpha),\ \text{Eff } \varphi_T(\alpha) \to \text{Eff } \varphi(\alpha).$$

The proof of Theorem 3.2 is provided in Appendix A.1.2.2. Theorem 3.2 implies that, under some specific assumptions, both the minimal point set and solution set of the objective $\varphi_T(\alpha)$ will converge to that of the objective $\varphi(\alpha)$ as $T \to +\infty$. Thus, Theorem 3.2 provides a theoretical justification our proposed approximation procedure (3.6).

3.4.3.1 Finite-Step Convergence

In Theorem 3.2, we have proved the convergence of Algorithm 1 when $T \to +\infty$. In practice, T only takes a finite value to reduce the computational cost. To further analyze how T and K affect the convergence of Algorithm 1, in this section, we provide a finite-step convergence analysis in a concrete case, where gradient descent is used to update ω and MGDA is used as a MOOSolver to update α.

To analyze the finite-step convergence of Algorithm 1, we first make the following assumptions.

Assumption 3.2. *We assume that* (i) *the function* $f(\omega, \alpha)$ *is* ϑ*-strongly-convex w.r.t.* ω*;* (ii) *the functions* ∇F_i *are* $\hat{c}_i$*-Lipschitz continuous;* (iii) *the gradient functions* ∇f *and* ∇F_i *are Lipschitz function;* (iv) *the functions* $\nabla_\alpha \nabla_\omega f$ *and* $\nabla^2_\omega f$ *are Lipschitz continuous.*

Note that the strongly convexity assumption in Assumption 3.2 can ensure that the LL subproblem satisfies the LLS condition and is commonly used in the analysis for the BLOs [34, 35]. The Lipschitz continuous assumption for the second order derivative is commonly used in the finite-step convergence analysis for BLOs [50].

In Algorithm 1, it first runs T steps of gradient decent to find an approximation point ω_T, and then calculates the gradient of φ w.r.t the UL variable α. For the i-th entry of φ, Algorithm 1 computes the gradient $\frac{\partial F_i(\omega_T(\alpha),\alpha)}{\partial \alpha}$ as an approximation of the true hyper-gradient that is computed as

$$\nabla \varphi_i(\alpha) = \frac{\partial F_i(\omega^*(\alpha),\alpha)}{\partial \alpha} = \nabla_\alpha F_i\left(\omega^*(\alpha),\alpha\right) + \frac{\partial \omega^*(\alpha)}{\partial \alpha} \nabla_\omega F_i\left(\omega^*(\alpha),\alpha\right). \tag{3.7}$$

Such approximation causes an estimation error in each iteration of the outer loop. With Assumption 3.2, we provide the following theorem to analyze this estimation error.

Theorem 3.3. *Suppose Assumption 3.2 holds, then if the step-size* $\mu \le 1/\vartheta$ *in Algorithm 1, we have*

$$\left\|\nabla_\alpha F_i(\omega_T(\alpha),\alpha) - \nabla_\alpha \varphi_i(\alpha)\right\| \le \left(c_1(1-\mu\vartheta)^{\frac{T}{2}} + c_2(1-\mu\vartheta)^{\frac{T-1}{2}}\right)\|\omega^0 - \omega^*(\alpha)\| + c_3(1-\mu\vartheta)^T, \tag{3.8}$$

where $\|\cdot\|$ *denotes the* ℓ_2 *norm of a vector,* ω^0 *is the initialization of* ω *in the inner loop, and* c_1, c_2, c_3 *are constants, which rely on the Lipschitz constants.*

The proof of Theorem 3.3 is provided in Appendix A.1.2.3. Theorem 3.3 shows that the gradient estimation error $\left\|\nabla_\alpha F_i(\omega_T(\alpha),\alpha)-\nabla_\alpha\varphi_i(\alpha)\right\|$ decays exponentially w.r.t. the number of steps in the inner loop. For notation simplicity, we denote by $\Gamma(T)$ the right-hand side of the inequality (3.8). Therefore, according to Theorem 3.3, we have $\Gamma(T)\to 0$ as $T\to+\infty$ if the step size of the inner loop satisfies $\mu\le 1/\vartheta$.

In each iteration of the outer loop, the convex combination coefficients are determined by the MGDA method. We denote by $\widetilde{\gamma}_k$ the convex combination vector determined by the estimated gradients $\nabla_\alpha F_i(\omega_T(\alpha),\alpha)$ in the k-th iteration. We denote by γ_k the corresponding convex combination vector calculated by the true gradients $\nabla\varphi_i(\alpha_k)$ in the k-th iteration. Then, the solution sequence $\{\alpha_k\}$ generated by MGDA method in Algorithm 1 can be formulated as $\alpha_{k+1}=\alpha_k-\nu\Lambda(\nabla_\alpha F_i(\omega_T(\alpha),\alpha),\widetilde{\gamma}_k)$, where the function $\Lambda(\cdot,\cdot)$ denotes a combination of the first argument weighted by the second argument, i.e. $\Lambda(\varphi,\gamma)=\sum_{i=1}^m\gamma_i\varphi_i$. Then we have the following theoretical result for the sequence $\{\alpha_k\}$.

Theorem 3.4. *Suppose Assumption 3.2 holds, then for $1\le i\le m$, if the i-th entry of the function φ_T is c_i-strongly-convex, the i-th entry of the function φ is L_i-Lipschitz continuous, the step size ν_k in the outer loop of Algorithm 1 equals $\frac{2}{c(k+1)}$, and $\mu\le 1/\vartheta$, then for any point $\alpha^*\in\mathscr{A}$, we have*

$$\min_{t=1,\dots,K}\Lambda\left(\varphi_T(\alpha_k),\widetilde{\gamma}_k\right)-\Lambda\left(\varphi_T(\alpha^*),\bar{\gamma}_k\right)\le\frac{4L^2+4\Gamma(T)^2}{c(K+1)},\tag{3.9}$$

holds for the sequence $\{\alpha_k\}_{k=1}^K$, where $\bar{\gamma}_k=\frac{\sum_{k=1}^K k\widetilde{\gamma}_k}{\sum_{k=1}^K k}$, $L=\max_{1\le i\le m}L_i$, and $c=\min_{1\le i\le m}c_i$.

The proof of Theorem 3.4 is provided in Appendix A.1.2.4. For any initialization, in Algorithm 1 we have a sequence of simplex vectors $\{\widetilde{\gamma}_k\}_{k=1}^K$ by the MGDA method as the MOO solver. This sequence is bounded and it has one limit point denoted by $\widetilde{\gamma}^*$. Since each entry of the function φ_T is strongly-convex, the weighted objective $\Lambda(\varphi_T(\alpha),\gamma)$ has only one minimizer for any $\gamma\in\Delta^{m-1}$, where Δ^{m-1} denotes an $(m-1)$-dimensional simplex. Let α^* be the unique solution of the objective $\Lambda(\varphi_T(\alpha),\widetilde{\gamma}^*)$. Then α^* is a Pareto-optimal solution associated with the vector $\widetilde{\gamma}^*$. Since the sequence $\bar{\gamma}_k$ converges to $\widetilde{\gamma}^*$, Theorem 3.4 implies that $\min_{k=1,\dots,K}\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k)$ converges to the Pareto-optimal solution $\Lambda(\varphi_T(\alpha^*),\widetilde{\gamma}^*)$ with the convergence rate as $1/K$.

Theorem 3.4 only compares $\{\alpha_k\}$ with the Pareto-optimal solution of the approximation objective function φ_T. To fully analyze the relation between $\{\alpha_k\}$ and the Pareto-optimal solution of the original objective function φ, we have the following theorem.

Theorem 3.5. *Let α^* be the Pareto-optimal solution of $\varphi_T(\alpha)$ corresponding to the limit point $\widetilde{\gamma}^*$ of the sequence $\{\widetilde{\gamma}_k\}$. With the assumptions in Assumption 3.2 and Theorems 3.4, if the i-th entry of the function φ is $\hat{c}_i$-strongly-convex and $\mu \leq 1/\vartheta$, we have*

$$\left\|\Lambda\left(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*\right)-\Lambda\left(\varphi_T(\alpha^*),\widetilde{\gamma}^*\right)\right\| \leq \frac{\delta L}{\hat{c}}\Gamma(T)+(1-\mu\vartheta)^T\hat{L}\|\omega^0-\omega^*(\alpha_k)\|, \tag{3.10}$$

where $\bar{\alpha}^*$ is the Pareto-optimal solution of $\varphi(\alpha)$ corresponding to the weighted vector $\widetilde{\gamma}^*$, δ is the diameter of the bounded set $\mathscr{A}$, $\hat{L}=\max_{1\leq i\leq m}\hat{L}_i$, and $\hat{c}=\min_{1\leq i\leq m}\hat{c}_i$.

The proof of Theorem 3.5 is provided in Appendix A.1.2.5. Theorem 3.5 implies that for the limit point of the sequence $\{\widetilde{\gamma}_k\}$, the distance between the corresponding Pareto-optimal points in $\mathrm{Min}(\varphi(\alpha))$ and $\mathrm{Min}(\varphi_T(\alpha))$ decays exponentially w.r.t. T.

To analyze the convergence of $\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k)$ to the optimal minimal point $\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)$, based on Theorem 3.5 as well as an assumption that the generated sequence $\{\widetilde{\gamma}_k\}$ approximates well the limit point $\widetilde{\gamma}^*$ in the Pareto-optimal set $\mathrm{Min}(\varphi_T(\alpha))$, we have the following theorem.

Theorem 3.6. *Let α^* be the Pareto-optimal solution of $\varphi_T(\alpha)$ corresponding to the limit point $\widetilde{\gamma}^*$ of the sequence $\{\widetilde{\gamma}_k\}$. With the assumptions in Assumption 3.2 and Theorem 3.4-3.5, we also assume that $\Lambda(\nabla\varphi_T(\alpha^*),\widetilde{\gamma}_k)^\top(\alpha_k-\alpha^*)\geq 0$ and the i-th entry of the function φ_T is M_i-Lipschitz. Then if the step size ν_k in the outer loop of Algorithm 1 satisfies $\nu_k=\xi/k$ where $\xi\geq 1/2c$, and $\mu\leq 1/\vartheta$, we have*

$$\begin{aligned}&\left\|\Lambda\left(\varphi_T(\alpha_k),\widetilde{\gamma}^*\right)-\Lambda\left(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*\right)\right\|\\ &\leq \frac{\delta L}{\hat{c}}\Gamma(T)+(1-\mu\vartheta)^T\hat{L}\|\omega^0-\omega^*(\alpha_k)\|+\frac{\max\{2\xi LM(2c\xi-1)^{-1},M\|\alpha^0-\alpha^*\|^2\}}{k},\end{aligned} \tag{3.11}$$

where α^0 is the initialization of α in the inner loop, $\bar{\alpha}^*$ is the Pareto-optimal solution of $\varphi(\alpha)$ corresponding to the weighted vector $\widetilde{\gamma}^*$, and $M=\max_{1\leq i\leq m}M_i$.

The proof of Theorem 3.6 is provided in Appendix A.1.2.6. Theorem 3.6 gives the finite-step convergence result, which depends on both numbers of steps in the inner and outer loops (i.e. T and K). It demonstrates the efficacy of the proposed algorithm. When $T\to+\infty$, the error bound has a $\mathrm{O}(1/k)$ convergence rate that matches the original MGDA algorithm [33].

3.5 A First-Order Gradient-Based MOBLO Method

MOML [133, 134] and MoCo [31] are proposed as effective gradient-based MOBLO algorithms, which hierarchically optimize the UL and LL variables based on ITerative

Differentiation (ITD) based Bi-Level Optimization (BLO) approach [34, 35, 42]. Specifically, given α, both MOML and MoCo first compute the LL solution $\omega^*(\alpha)$ by solving LL subproblem with T iterations and then update α via the combination of the hypergradients $\{\nabla_\alpha F_i(\alpha, \omega^*(\alpha))\}_{i=1}^m$. Note that they need to calculate the complex gradient $\nabla_\alpha \omega^*(\alpha)$, which requires to compute many Hessian-vector products via the chain rule. Besides, their time and memory costs grow significantly fast with respect to the dimension of ω and T. Therefore, existing gradient-based methods to solve MOBLO problems could suffer from the inefficiency problem, especially in deep neural networks.

To address this limitation, we further propose an efficient **F**irst-**O**rde**R** m**U**lti-gradient method [132] for **M**OBLO (**FORUM**). Specifically, we reformulate MOBLO as an equivalent constrained multi-objective optimization (MOO) problem by the value-function-based approach [67, 72, 109]. Then, we propose a multi-gradient aggregation method to solve the challenging constrained MOO problem. Different from existing MOBLO methods such as MOML and MoCo, FORUM is a fully first-order algorithm and does not need to calculate the high-order Hessian matrix. The complexity analysis shows that FORUM is more efficient than MOML and MoCo in both time and memory costs, as summarised in Table 3.1. In this section, we introduce the details of the proposed FORUM method.

3.5.1 Reformulation of MOBLO

Based on the value-function-based approach [58, 67, 72, 109], we reformulate MOBLO problem (3.3) as an equivalent single-level *constrained multi-objective optimization* problem:

$$\min_{\alpha\in\mathscr{A},\omega\in\mathbb{R}^p} F(\alpha,\omega) \quad \text{s.t.} \quad f(\alpha,\omega)\le f^*(\alpha), \tag{3.12}$$

where $f^*(\alpha) = \min_\omega f(\alpha,\omega) = f(\alpha,\omega^*(\alpha))$ is the *value function*, which represents the lower bound of $f(\alpha,\omega)$ w.r.t. ω. To simplify the notation, we define $z \equiv (\alpha,\omega) \in \mathbb{R}^{n+p}$ and $\mathscr{Z} \equiv \mathscr{A}\times\mathbb{R}^p$. Then, we have $F(z)\equiv F(\alpha,\omega)$ and $f(z)\equiv f(\alpha,\omega)$. Thus, problem (3.12) can be rewritten as

$$\min_{z\in\mathscr{Z}} F(z) \quad \text{s.t.} \quad q(z)\le 0, \tag{3.13}$$

where $q(z) = f(z) - f^*(\alpha)$ is the *constraint function*. Since the gradient of the value function $f^*(\alpha)$ is

$$\nabla_\alpha f^*(\alpha) = \nabla_\alpha f(\alpha,\omega^*(\alpha)) = \nabla_\alpha f(\alpha,\omega^*), \tag{3.14}$$

where the second equality is due to the chain rule and $\nabla_\omega f(\alpha,\omega)|_{\omega=\omega^*(\alpha)}=0$, we do not need to compute the complex Hessian matrix $\nabla_\alpha \omega^*(\alpha)$ like MOML and MoCo.

However, solving problem (3.13) is challenging for two reasons. One reason is that the Slater's condition [13], which is required for duality-based optimization methods, does not hold for problem (3.13), since the constraint $q(z)\le 0$ is ill-posed [52, 72] and does not have an interior point. To see this, we assume $z_0=(\alpha_0,\omega_0)\in\mathcal{Z}$ and $q(z_0)\le 0$. Then the constraint $q(z)\le 0$ is hard to be satisfied at the neighborhood of α_0, unless $f^*(\alpha)$ is a constant function around α_0, which rarely happens. Therefore, problem (3.13) cannot be treated as classic constrained optimization and we propose a novel gradient method to solve it in Section 3.5.2. Another reason is that for given α, the computation of $\omega^*(\alpha)$ is intractable. Thus, we approximate it by $\tilde{\omega}^T$ computed by T steps of gradient descent. Specifically, given α and an initialization $\tilde{\omega}^0$ of ω, we have

$$\tilde{\omega}^{t+1}=\tilde{\omega}^t-\eta\nabla_\omega f(\alpha,\tilde{\omega}^t),\quad t=0,\cdots,T-1, \tag{3.15}$$

where η represents the step size. Then, the constraint function $q(z)$ is approximated by $\widetilde{q}(z)=f(z)-f(\alpha,\tilde{\omega}^T)$ and its gradient $\nabla_z q(z)$ is approximated by $\nabla_z\widetilde{q}(z)$. The approximation error of the gradient $\nabla_z\widetilde{q}(z)$ exponentially decays w.r.t. the LL iterations T [132]. Hence, problem (3.13) is modified to

$$\min_{z\in\mathcal{Z}} F(z)\quad \text{s.t.}\ \ \widetilde{q}(z)=f(z)-f(\alpha,\tilde{\omega}^T)\le 0. \tag{3.16}$$

3.5.2 FORUM Algorithm

We now introduce the proposed FORUM method for solving problem (3.16). Specifically, at k-th iteration, assume z_k is updated by

$$z_{k+1}=z_k+\mu d_k, \tag{3.17}$$

where μ is the step size and d_k is the update direction for z_k. Then, we expect d_k can simultaneously minimize the UL objective $F(z)$ and the constraint function $\widetilde{q}(z)$. Note that the minimum of the approximated constraint function $\widetilde{q}(z)$ converges to the minimum of $q(z)$, i.e. 0, as $T\to+\infty$. Thus, we expect d_k to decrease $\widetilde{q}(z)$ consistently such that the constraint $\widetilde{q}(z)\le 0$ is satisfied.

Note that there are multiple potentially conflicting objectives $\{F_i\}_{i=1}^m$ in the UL subproblem. Hence, we expect d_k can decrease every objective F_i, which can be formulated as the following problem to find d_k to maximize the minimum decrease across all objectives

as

$$\max_{d} \min_{i\in\{1,\dots,m\}} (F_i(z_k) - F_i(z_k+\mu d)) \approx -\mu \min_{d} \max_{i\in\{1,\dots,m\}} \langle \nabla F_i(z_k), d\rangle. \tag{3.18}$$

To regularize the update direction, we add a regularization term $\frac{1}{2}\|d\|^2$ to problem (3.18) and compute d_k by solving

$$\min_{d} \max_{i\in[m]} \langle \nabla F_i(z_k), d\rangle + \frac{1}{2}\|d\|^2. \tag{3.19}$$

To decrease the constraint function $\widetilde{q}(z)$, we expect the inner product of $-d$ and $\nabla\widetilde{q}(z_k)$ to hold positive during the optimization process, i.e., $\langle \nabla\widetilde{q}(z_k), -d\rangle \geq \phi$, where ϕ is a non-negative constant.

To further guarantee that $\widetilde{q}(z)$ can be optimized such that the constraint $\widetilde{q}(z)\leq 0$ can be satisfied, we introduce a dynamic ϕ_k here. Specifically, inspired by [40], we set $\phi_k = \frac{\rho}{2}\|\nabla\widetilde{q}(z_k)\|^2$, where ρ is a positive constant. When $\phi_k > 0$, it means that $\|\nabla\widetilde{q}(z)\| \neq 0$ and $\widetilde{q}(z)$ should be further optimized, and $\langle \nabla\widetilde{q}(z_k), -d\rangle \geq \phi_k > 0$ can enforce $\widetilde{q}(z)$ to decrease. When ϕ_k equals 0, it indicates that the optimum of $\widetilde{q}(z)$ is reached and $\langle \nabla\widetilde{q}(z_k), -d\rangle \geq \phi_k = 0$ also holds. Thus, the dynamic ϕ_k can ensure d_k to iteratively decrease $\widetilde{q}(z)$ such that the constraint $\widetilde{q}(z)\leq 0$ is satisfied.

Therefore, at k-th iteration, we can find d_k by solving the problem:

$$\begin{aligned} &\min_{d} \max_{i\in\{1,\dots,m\}} \langle \nabla F_i(z_k), d\rangle + \frac{1}{2}\|d\|^2, \\ &\text{s.t. } \langle \nabla\widetilde{q}(z_k), d\rangle \leq -\phi_k. \end{aligned} \tag{3.20}$$

This problem can be rewritten equivalently as the following differentiable quadratic optimization

$$d,\mu = \arg\min_{d,\mu}\left(\frac{1}{2}\|d\|^2 + \mu\right) \text{ s.t. } \langle \nabla\widetilde{q}(z_k), d\rangle \leq -\phi, \ \langle \nabla F_i(z_k), d\rangle \leq \mu. \tag{3.21}$$

Based on the Lagrangian multiplier method, we have

$$L = \frac{1}{2}\|d\|^2 + \mu + \sum_{i=1}^{m}\lambda_i(\langle \nabla F_i(z_k), d\rangle - \mu) + \nu(\langle \nabla\widetilde{q}(z_k), d\rangle + \phi), \ \text{ s.t. } \sum_{i=1}^{m}\lambda_i = 1. \tag{3.22}$$

Differentiate with respect to d, and let $\nabla_d L = 0$ we have

$$d + \sum_{i=1}^{m}\lambda\nabla F_i(z_k) + \nu\nabla\widetilde{q}(z_k) = 0. \tag{3.23}$$

Therefore, the gradient $d = -(\sum_{i=1}^{m}\lambda\nabla F_i(z_k) + \nu\nabla\widetilde{q}(z_k))$. Substitute it to problem (3.21), we obtain that λ and ν are the solution of

$$\min_{\lambda\in\Delta^{m-1},\nu\geq 0} \frac{1}{2}\left\|\sum_{i=1}^{m}\lambda\nabla F_i(z_k) + \nu\nabla\widetilde{q}(z_k)\right\|^2 - \nu\phi. \tag{3.24}$$

For given λ, the above equation has a colsed form solution for ν,

$$\nu(\lambda) = \max\left(\sum_{i=1}^{m} \lambda_i \pi_i(z), 0\right), \text{ s.t. } \pi_i(z) = \frac{2\phi - \langle \nabla \widetilde{q}(z), \nabla F_i(z)\rangle}{\|\nabla \widetilde{q}(z)\|^2}. \tag{3.25}$$

Therefore, problem (3.20) has a solution as

$$d_k = -\left(\textstyle\sum_{i=1}^{m} \lambda_i^k \nabla F_i(z_k) + \nu(\lambda^k)\nabla \widetilde{q}(z_k)\right), \tag{3.26}$$

where Lagrangian multipliers $\lambda^k = (\lambda_1^k, \ldots, \lambda_m^k) \in \Delta^{m-1}$ (i.e., $\sum_{i=1}^{m} \lambda_i^k = 1$ and $\lambda_i^k \geq 0$) and $\nu(\lambda)$ is a function of λ as

$$\begin{aligned} &\nu(\lambda) = \max\left(\textstyle\sum_{i=1}^{m} \lambda_i \pi_i(z_k), 0\right), \\ &\text{with } \pi_i(z_k) = \frac{2\phi_k - \langle \nabla \widetilde{q}(z_k), \nabla F_i(z_k)\rangle}{\|\nabla \widetilde{q}(z_k)\|^2}, \end{aligned} \tag{3.27}$$

and λ_i^k can be obtained by solving the following dual problem as

$$\lambda^k = \arg\min_{\lambda \in \Delta^{m-1}} \frac{1}{2} \left\| \textstyle\sum_{i=1}^{m} \lambda_i \nabla F_i(z_k) + \nu(\lambda)\nabla \widetilde{q}(z_k) \right\|^2 - \nu(\lambda)\phi_k. \tag{3.28}$$

Problem (3.28) can be reformulated as

$$\begin{aligned} &\min_{\lambda \in \Delta^{m-1}, \gamma} \frac{1}{2} \left\| \textstyle\sum_{i=1}^{m} \lambda_i \nabla F_i(z_k) + \gamma \nabla \widetilde{q}(z_k) \right\|^2 - \gamma\phi_k \\ &\text{s.t. } \gamma \geq 0, \gamma \geq \textstyle\sum_{i=1}^{m} \lambda_i \pi_i(z_k). \end{aligned} \tag{3.29}$$

The first term of the objective function in problem (3.29) can be simplified to $R^\top \Lambda^\top \Lambda R$, where $R = (\lambda_1, \ldots, \lambda_m, \gamma)^\top$ and $\Lambda = (\nabla F_1, \ldots, \nabla F_m, \nabla \widetilde{q})$. Note that the dimension of the matrix $\Lambda^\top \Lambda$ is $(m+1) \times (m+1)$, which is independent with the dimension of z. As the number of UL objectives m is usually small compared with the dimension of z, solving problem (3.29) does not incur too much computational cost. In practice, we can use the open-source `CVXPY` library [24] to solve problem (3.29).

To ensure convergence, the sequence of $\{\lambda^k\}_{k=1}^K$ should be a convergent sequence (refer to the discussion in [132]). However, $\{\lambda^k\}_{k=1}^K$ obtained by directly solving the problem (3.29) in each iteration cannot ensure such properties. Therefore, we apply a momentum strategy [135, 149] to λ to generate a stable sequence and further guarantee the convergence. Specifically, in k-th iteration, we first solve the problem (3.29) to obtain λ^k, then update the weights by

$$\tilde{\lambda}^k = (1 - \beta_k)\tilde{\lambda}^{k-1} + \beta_k \lambda^k, \tag{3.30}$$

Algorithm 2 The FORUM Method

Input: number of iterations (K, T), step size (μ, η), coefficient β_k, constant ρ

Randomly initialize $z_0 = (\alpha_0, \omega_0)$;
Initialize $\tilde{\lambda}_i^{-1} = 1/m$, $i = 1, ..., m$;
for $k = 0$ **to** $K - 1$ **do**
    Set $\tilde{\omega}^0 = \omega_0$ or $\tilde{\omega}^0 = \omega_k$;
    for $t = 0$ **to** $T - 1$ **do**
        Update $\tilde{\omega}$ as $\tilde{\omega}^{t+1} = \tilde{\omega}^t - \eta \nabla_\omega f(\alpha_k, \tilde{\omega}^t)$;
    end for
    Set $\widetilde{q}(z_k) = f(z_k) - f(\alpha_k, \tilde{\omega}^T)$;
    Compute gradient $\nabla_z \widetilde{q}(z_k) = \nabla_z f(z_k) - \nabla_\alpha f(\alpha_k, \tilde{\omega}^T)$;
    Compute gradients $\nabla_z F_i(z_k)$, $i = 1, \ldots, m$;
    Compute λ^k by solving problem (3.29);
    Update $\tilde{\lambda}^k$ by $\tilde{\lambda}^k = (1 - \beta_k)\tilde{\lambda}^{k-1} + \beta_k \lambda^k$;
    Compute $v(\tilde{\lambda}^k)$ via Eq. (3.27);
    Compute d_k via Eq. (3.26);
    Update z as $z_{k+1} = z_k + \mu d_k$;
end for
return z_K.

where $\beta_k \in (0, 1]$ is set to 1 at the beginning and asymptotically convergent to 0 as $k \to +\infty$.

After obtaining $\tilde{\lambda}^k$ with the momentum update in k-th iteration, we can compute the corresponding $v(\tilde{\lambda}^k)$ via Eq. (3.27). Then we obtain the update direction d_k by Eq. (3.26) and update the variable z_k as $z_{k+1} = z_k + \mu d_k$. The entire FORUM algorithm is shown in Algorithm 2.

3.5.3 Convergence Analysis

In this section, we provide convergence analysis for the FORUM method.

Consider a general constrained MOO problem with one constraint such as problem (3.13). The corresponding first-order Karush-Kuhn-Tucker (KKT) condition [30] is said to hold at a feasible point $z^* \in \mathcal{Z}$ if there exist a vector $\lambda \in \Delta^{m-1}$ and $v \in \mathbb{R}_+$ such that the following three conditions hold,

$$\sum_{i=1}^{m} \lambda_i \nabla F_i(z^*) + v \nabla q(z^*) = 0, \quad q(z^*) \leq 0, \text{and} \quad v q(z^*) = 0. \tag{3.31}$$

Then z^* is a local optimal point. The first condition is the stationarity condition, the second condition is the primal feasibility condition and the last condition is the complementary slackness condition. However, as we discussed in Section 3.5.1, since the

constraint function $q(z)$ is ill-posed, the complementary slackness condition can not be satisfied [58, 67]. To ensure our algorithm converges to a weak stationarity point, we measure the convergence by the stationarity condition and the feasibility condition.

Discussion on Pareto stationary. The Pareto stationary is a first-order KKT stationary condition for the unconstrained MOO optimization problem. However, in this work, we reformulate MOBLO to an equivalent constrained MOO problem. Hence, the Pareto stationary cannot be used as a convergence criterion in our method. We measure the convergence by the local optimality condition of the constrained MOO problem, i.e., KKT stationary and feasibility conditions.

To analyze the convergence property of FORUM. Firstly, we make an assumption for the UL subproblem.

Assumption 3.3. *For $i = 1,\dots,m$, it is assumed that the gradient $\nabla F_i(\alpha,\omega)$ is L_F-Lipschitz continuous with respect to $z := (\alpha,\omega)$. The ℓ_2 norm of $\nabla F_i(z)$ and $|F_i(z)|$ are upper-bounded by a positive constant M.*

The smoothness and the boundedness assumptions in Assumption 3.3 are widely adopted in non-convex multi-objective optimization [31, 149]. Then we make an assumption for the LL subproblem.

Assumption 3.4. *The function $f(\alpha,\omega)$ is c-strongly convex with respect to ω, and the gradient $\nabla f(\alpha,\omega)$ is L_f-Lipschitz continuous with respect to $z := (\alpha,\omega)$.*

The strongly convexity assumption in Assumption 3.4 is commonly used in the analysis for the BLO [34, 35] and MOBLO problems [31, 133]. The proposed FORUM algorithm focuses on generating one Karush-Kuhn-Tucker (KKT) stationary point of the original constrained multi-objective optimization problem (3.13). Following [40, 67], we measure the convergence of problem (3.13) by both its KKT stationary condition and the feasibility condition. Specifically, we denote by $\mathscr{K}(z_k) = \left\|\sum_{i=1}^m \tilde{\lambda}_i^k \nabla F_i(z_k) + \nu_k \nabla q(z_k)\right\|^2$ the measure of KKT stationary condition in the k-th iteration, where $\nu_k = \nu(\tilde{\lambda}^k)$. To satisfy the feasibility condition of problem (3.13), the non-negative function $q(z_k)$ should decrease to 0. Then, with a non-convex multi-objective UL subproblem, we have the following convergence result.

Theorem 3.7. *Suppose that Assumptions 3.3 and 3.4 hold, and the sequence $\{z_k\}_{k=0}^K$ generated by Algorithm 2 satisifes $q(z_k) \le B$, where B is a positive constant. Then if*

$\eta \leq 1/L_f$, $\mu = \mathcal{O}(K^{-1/2})$, and $\beta = \mathcal{O}(K^{-3/4})$, there exists a constant $C > 0$ such that when $T \geq C$, for any $K > 0$, we have

$$\max\left\{\min_{k<K} \mathcal{K}(z_k), q(z_k)\right\} = \mathcal{O}(K^{-1/4} + \Gamma(T)), \tag{3.32}$$

where $\Gamma(T)$ represents exponential decays with respect to T.

The proof is put in Appendix A.1.3.2. Theorem 3.7 gives a non-asymptotic convergence result for Algorithm 2 based on the KKT stationary condition and the feasibility condition of the problem (3.13). The proposed FORUM method achieves a $\mathcal{O}(K^{-1/4} + \Gamma(T))$ convergent rate, which depends on both numbers of steps in the UL and LL subproblems (i.e., K and T).

3.6 Complexity Analysis

In this section, we analyze the proposed MOML method [133, 134], FORUM method [132], and MoCo [31]

For the proposed FORUM method, it takes time $\mathcal{O}(pT)$ and space $\mathcal{O}(p)$ to obtain the approximated constraint function $\widetilde{q}(z)$, and then the computations of all the gradients including $\nabla_z F_i(z)$ and $\nabla_z \widetilde{q}(z)$ require time $\mathcal{O}((n+p)(m+1))$ and space $\mathcal{O}((n+p)(m+1))$. When the number of UL objectives m satisfies $m \ll n+p$, the time and space costs of solving the quadratic programming problem (3.29), which only depends on m, can be negligible. Therefore, FORUM runs in time $\mathcal{O}(mn + p(m+T))$ and space $\mathcal{O}(mn+mp)$ in total for each UL iteration.

For the MOML method, it takes $\mathcal{O}(pT)$ time and $\mathcal{O}(p)$ space to do the T-iteration update for the LL subproblem. Then calculating the Hessian-matrix product via backward propagation in each UL iteration can be evaluated in time $\mathcal{O}(p(n+p)T)$ and space $\mathcal{O}(n+pT)$. Similar to the FORUM method, the cost of solving the quadratic programming problem in MOML is also negligible. Therefore, for each UL iteration, MOML require $\mathcal{O}(mp(n+p)T)$ time and $\mathcal{O}(mn+mpT)$ space in total. For the MoCo method, it uses a similar approach to MOML to calculate the Hessian-matrix product via backward propagation in each UL iteration. Note that MoCo applies a momentum update to the UL variables α, which causes an additional $\mathcal{O}(mn)$ space cost. Thus, for each UL iteration, MoCo require $\mathcal{O}(mp(n+p)T)$ time and $\mathcal{O}(2mn+mpT)$ space in total.

In summary, the above analysis indicates that the proposed FORUM method is more efficient than MOML and MoCo in terms of both time and space complexity. The results are summarised in Table 3.1.

Table 3.1: Comparison of convergence result and complexity analysis per UL iteration for different MOBLO methods. m, n, p, and T denote the number of UL objectives, the dimension of the UL variables, the dimension of the LL variables, and the number of LL iterations, respectively.

Method	**Convergence analysis**	**Computational cost**	**Space cost**
MOML [133, 134]	asymptotic / non-asymptotic	$\mathcal{O}(mp(n+p)T)$	$\mathcal{O}(mn+mpT)$
MoCo [31]	non-asymptotic	$\mathcal{O}(mp(n+p)T)$	$\mathcal{O}(2mn+mpT)$
FORUM [132]	non-asymptotic	$\mathcal{O}(mn+p(m+T))$	$\mathcal{O}(mn+mp)$

3.7 Experiment Result

In this section, we empirically evaluate the proposed methods on different learning problems. All experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU.

3.7.1 Multi-Task Data Hyper-Cleaning

Setup. Data hyper-cleaning [4, 34, 67, 104] is a hyperparameter optimization problem, where a model is trained on a dataset with part of training labels corrupted. Thus, it aims to reduce the influence of noisy examples by adding weights to the train samples and learning these weights in a bi-level optimization manner. Here we extend data hyper-cleaning to a multi-task setting, where we aim to train a model on multiple corrupted datasets.

Specifically, suppose that there are m corrupted datasets. $\mathcal{D}_i^{\text{tr}} = \{x_{i,j}, y_{i,j}\}_{j=1}^{N_i}$ and $\mathcal{D}_i^{\text{val}}$ denote the noisy training set and the clean validation set for the i-th dataset, respectively, where $x_{i,j}$ denotes the j-th training sample in the i-th dataset, $y_{i,j}$ is the corresponding label, and N_i denotes the size of the i-th training dataset. Let ω denote the model parameters and $\alpha_{i,j}$ denotes the weight of the training sample $x_{i,j}$. Let $\mathcal{L}_i^{\text{val}}(\omega; \mathcal{D}_i^{\text{val}})$ be the average loss of model ω on the clean validation set of the i-th dataset and

$$\mathcal{L}_i^{\text{tr}}(\alpha, \omega; \mathcal{D}_i^{\text{tr}}) = \frac{1}{N_i} \sum_{j=1}^{N_i} \sigma(\alpha_{i,j}) \ell(\omega; x_{i,j}, y_{i,j})$$

be the weighted average loss on the noisy training set of the i-th dataset, where $\sigma(\cdot)$ is an element-wise sigmoid function to constrain each weight in the range $[0, 1]$ and $\ell(\omega; x, y)$ denotes the loss of model ω on sample (x, y). Therefore, the objective function of this

multi-task data hyper-cleaning is formulated as

$$\min_{\alpha,\omega} \left(\mathscr{L}_1^{\text{val}}(\omega; \mathscr{D}_1^{\text{val}}), \cdots, \mathscr{L}_m^{\text{val}}(\omega; \mathscr{D}_m^{\text{val}}) \right)^\top$$
$$\text{s.t. } \omega \in \mathscr{S}(\alpha) = \arg\min_{\omega} \sum_{i=1}^m \mathscr{L}_i^{\text{tr}}(\alpha, \omega; \mathscr{D}_i^{\text{tr}}).$$

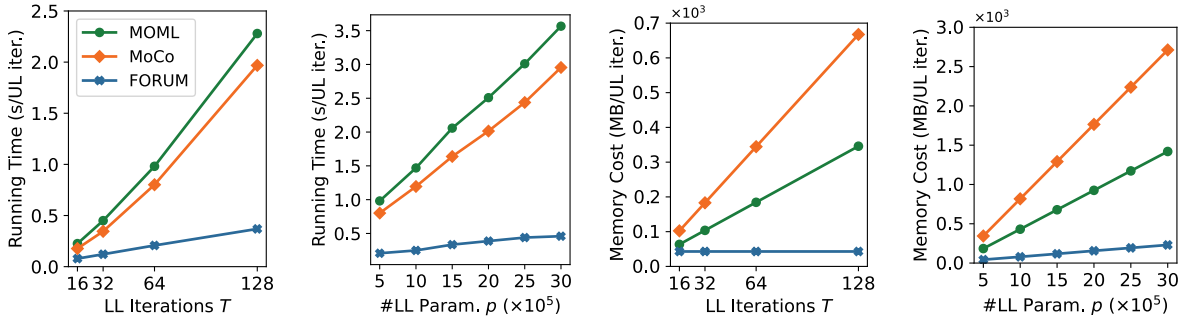
Datasets. We conduct experiments on the MNIST [61] and FashionMNIST [130] datasets. Each dataset corresponds to a 10-class image classification problem. All the images have the same size of 28×28. Following [4], we randomly sample 5000, 1000, 1000, and 5000 images from each dataset as the training set, first validation set, second validation set, and test set, respectively. The training set and first validation set are used to formulate the LL and UL subproblems, respectively. The second validation set is used to select the best model and the testing evaluation is conducted on the test set. Half of the samples in the training set are contaminated by assigning them to another random class.

Implementation Details. We conduct experiments on three MOBLO methods: MOML, MoCo, and FORUM. The same configuration is used for both the MOML, MoCo, and FORUM methods. Specifically, the hard-parameter sharing architecture [146] is used, where the bottom layers are shared among all datasets and each dataset has its specific head layers. The shared module contains two linear layers with input size, hidden size, and output size of 784, 512, and 256. Each layer adopts a ReLU activation function. Each dataset has a specific linear layer with an output size of 10. The batch size is set to 100. For the LL subproblem, the SGD optimizer with a learning rate $\eta = 0.3$ is used for updating $T = 64$ iterations. For the UL subproblem, the total number of UL iterations K is set to 1200, and an SGD optimizer with the learning rate as 10 is used for updating weight α while another SGD optimizer with the learning rate as 0.3 is used for updating model parameters ω. We set $\rho = 0.5$ and $\beta_k = (k+1)^{-\frac{3}{4}}$ for FORUM. For Figures 3.1(b) and 3.1(d), we increase the number of LL parameters p by adding some linear layers with the hidden size of 512 into the shared module. We use the build-in function `torch.cuda.max_memory_allocated` in PyTorch [91] to compute the GPU memory cost in Figures 3.1(c) and 3.1(d).

Results. The classification accuracy and F1 score computed on the test set are used as the evaluation metrics. The results are provided in Table 3.2. As can be seen, the FORUM method outperforms the MOML and MoCo in both datasets, which demonstrates its effectiveness.

Table 3.2: Performance of different methods on the MNIST and FashionMNIST datasets for the multi-objective data hyper-cleaning problem. 3 independent runs are conducted for each experiment. The mean and the standard deviation are reported. The best result is marked in **bold**.

Methods	**MNIST**		**FashionMNIST**	
	Accuracy (%)	**F1 Score**	**Accuracy (%)**	**F1 Score**
MOML [133]	$88.64_{\pm 0.94}$	$88.61_{\pm 0.98}$	$80.64_{\pm 0.35}$	$80.60_{\pm 0.49}$
MoCo [31]	$88.05_{\pm 1.21}$	$88.03_{\pm 1.27}$	$80.94_{\pm 0.19}$	$80.67_{\pm 0.25}$
FORUM [132]	$\mathbf{90.81}_{\pm 0.14}$	$\mathbf{90.81}_{\pm 0.15}$	$\mathbf{82.07}_{\pm 0.38}$	$\mathbf{81.72}_{\pm 0.57}$



(a) Running time vs. T. (b) Running time vs. p. (c) Memory cost vs. T. (d) Memory cost vs. p.

Figure 3.1: Results of different MOBLO methods on the multi-objective data hyper-cleaning problem. **(a)**: The running time per iteration varies over different LL update steps T with fixed numbers of LL parameters p. **(b)**: The running time per iteration varies over the different numbers of LL parameters p with $T = 64$. **(c)**: The memory cost varies over different LL update steps T with fixed numbers of LL parameters p. **(d)**: The memory cost varies over the different numbers of LL parameters p with $T = 64$.

Figures 3.1(a) and 3.1(b) show that MOML and MoCo need longer running time than FORUM in every configuration of the UL iteration T and the number of LL parameters p, respectively, which implies FORUM has a lower time complexity. Figures 3.1(c) and 3.1(d) show the change of memory cost per iteration with respect to the LL iteration T and the number of LL parameters p, respectively. As can be seen, the memory cost remains almost constant with different T's for FORUM and increases faster for MOML and MoCo. Moreover, the memory cost slightly increases in FORUM with increasing p, while it linearly increases in MOML and MoCo. In summary, the results in Figure 3.1 match the complexity analysis in Section 3.6 and demonstrate that FORUM is more efficient than MOML and MoCo in terms of both time and space complexity.

Table 3.3: Classification accuracy (%) on the **Office-31** dataset. 3 independent runs are conducted for each experiment. The average performance is reported. The best results over baselines except STL are marked in **bold**.

Methods	**A**	**D**	**W**	**Avg**	$\Delta_{\mathrm{p}}\uparrow$
STL	86.61	95.63	96.85	93.03	0.00
multi-task learning methods					
EW [146]	83.53	97.27	96.85	92.55	-0.61
UW [53]	83.82	97.27	96.67	92.58	-0.56
MGDA [103]	85.47	95.90	97.03	92.80	-0.27
PCGrad [141]	83.59	96.99	96.85	92.48	-0.68
GradDrop [17]	84.33	96.99	96.30	92.54	-0.59
GradVac [126]	83.76	97.27	96.67	92.57	-0.58
CAGrad [66]	83.65	95.63	96.85	92.04	-1.13
IMTL [70]	83.70	96.44	96.29	92.14	-1.02
Nash-MTL [89]	85.01	97.54	97.41	93.32	+0.24
RLW [64]	83.82	96.99	96.85	92.55	-0.59
first-order bi-level optimization methods					
BVFIM [72]	84.84	96.99	97.78	93.21	+0.11
BOME [67]	85.53	96.72	**98.15**	93.47	+0.41
multi-objective bi-level optimization methods					
MOML [133]	84.67	96.72	96.85	92.75	-0.36
MoCo [31]	84.38	97.26	97.03	92.89	-0.22
FORUM [132]	**85.64**	**98.63**	97.96	**94.07**	**+0.96**

3.7.2 Multi-Task Learning

Setup. We conduct experiments on three benchmark datasets among three different task categories, i.e., the Office-31 [101] dataset for the image classification task, the NYUv2 [105] dataset for the scene understanding task, and the QM9 dataset [97] for the molecular property prediction task.

Datasets. The Office-31 dataset [101] includes images from three different sources: Amazon (A), digital SLR cameras (D), and Webcam (W). It contains 31 categories for each source and a total of 4652 labeled images. We use the data split in RLW [64]: 60% for training, 20% for validation, and 20% for testing. The NYUv2 dataset [105], an indoor scene understanding dataset, has 795 and 654 images for training and testing,

Table 3.4: Results on the NYUv2 dataset. 3 independent runs are conducted for each experiment. The average performance is reported. The best results over baselines except STL are marked in **bold**. ↑ (↓) indicates that the higher (lower) the result, the better the performance.

Methods	Segmentation		Depth		Surface Normal Prediction					Δ_p↑
					Angle Distance		Within t°			
	mIoU↑	PAcc↑	AErr↓	RErr↓	Mean↓	Median↓	11.25↑	22.5↑	30↑	
STL	53.50	75.39	0.3926	0.1605	21.9896	15.1641	39.04	65.00	75.16	0.00
multi-task learning methods										
EW [146]	53.93	75.53	0.3825	0.1577	23.5691	17.0149	35.04	60.99	72.05	-1.78
UW [53]	**54.29**	75.64	0.3815	0.1583	23.4805	16.9206	35.26	61.17	72.21	-1.52
MGDA [103]	53.52	74.76	0.3852	0.1566	22.7400	16.0000	37.12	63.22	73.84	-0.64
PCGrad [141]	53.94	75.62	0.3804	0.1578	23.5226	16.9276	35.19	61.17	72.19	-1.57
GradDrop [17]	53.73	75.54	0.3837	0.1580	23.5392	16.9587	35.17	61.06	72.07	-1.85
GradVac [126]	54.21	75.67	0.3859	0.1583	23.5804	16.9055	35.34	61.15	72.10	-1.75
CAGrad [66]	53.97	75.54	0.3885	0.1588	22.4701	15.7139	**37.77**	63.82	74.30	-0.27
IMTL [70]	53.63	75.44	0.3868	0.1592	22.5800	15.8500	37.44	63.52	74.09	-0.57
Nash-MTL [89]	53.41	74.95	0.3867	0.1612	22.5662	15.9365	37.30	63.40	74.09	-1.01
RLW [64]	54.13	75.72	0.3833	0.1590	23.2125	16.6166	35.88	61.84	72.74	-1.27
first-order bi-level optimization methods										
BVFIM [72]	53.29	75.07	0.3981	0.1632	22.3552	15.9710	37.15	63.44	74.27	-1.68
BOME [67]	54.15	**75.79**	0.3831	0.1578	23.3378	16.8828	35.29	61.31	72.40	-1.45
multi-objective bi-level optimization methods										
MOML [133]	53.59	75.48	0.3839	0.1577	23.1487	16.5319	36.06	62.05	72.89	-1.26
MoCo [31]	53.73	75.63	0.3838	0.1560	23.1922	16.5737	36.02	61.93	72.82	-1.06
FORUM [132]	54.04	75.64	**0.3795**	**0.1555**	**22.1870**	**15.6815**	37.71	**64.04**	**74.67**	**+0.65**

respectively. It has three tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction. The QM9 dataset [97], a molecular property prediction dataset. We use the same data split as in Nash-MTL [89]: 110,000 for training, 10,000 for validation, and 10,000 for testing. The QM9 dataset contains 11 tasks and each task is a regression task for one property.

Baselines. The proposed MOML and FORUM methods are compared with a number of baseline methods from four different categories: *single-task learning* (STL) method that trains each task independently; a comprehensive set of state-of-the-art *MTL methods*, including Equal Weighting (EW) [146], UW [53], MGDA [103], PCGrad [141], GradDrop [17], GradVac [126], CAGrad [66], IMTL[70], Nash-MTL [89], and RLW [64]; two *first-order BLO methods*: BVFIM [72] and BOME [67], where we simply transform MOBLO to BLO by aggregating multiple objectives in the UL subproblem with equal weights into a single objective so that we can apply those BLO methods to solve the MOBLO problem; one *gradient-based MOBLO method*: MoCo [31].

Evaluation Metrics. For the Office-31 dataset, following RLW [64], we use classification accuracy as the evaluation metric for each task and the average accuracy as the overall metric. For the NYUv2 dataset, following RLW [64], we use the mean intersection over union (MIoU) and the class-wise pixel accuracy (PAcc) for the semantic segmentation task, the relative error (RErr) and the absolute error (AErr) for the depth estimation task, and the mean and median angle error as well as the percentage of normals within t° ($t = 11.25, 22.5, 30$) for the surface normal prediction task. For the QM9 dataset, following Nash-MTL [89], we use mean absolute error (MAE) as the evaluation metric.

Following [63, 64, 136], we use Δ_p as a metric to evaluate the overall performance on all the tasks. It is defined as the mean of the relative improvement of each task over the STL method, which is formulated as

$$\Delta_\mathrm{p} = 100\% \times \frac{1}{m} \sum_{i=1}^{m} \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{(-1)^{s_{i,j}}(M_{i,j} - M_{i,j}^{\mathrm{STL}})}{M_{i,j}^{\mathrm{STL}}}, \tag{3.33}$$

where N_i denotes the number of metrics for the i-th task, $M_{i,j}$ denotes the performance of an MTL method for the j-th metric in the i-th task, $M_{i,j}^{\mathrm{STL}}$ is defined in the same way for the STL method, and $s_{i,j}$ is set to 0 if a larger value represents better performance for the j-th metric in i-th task and otherwise $s_{i,j}$ is set to 1.

Following [66, 70, 89], 3 independent runs are conducted for each experiment. The average performance is reported.

Implementation Details. For the Office-31 dataset, we follow the approach outlined in RLW [64] by utilizing a pre-trained ResNet-18 network as a shared backbone for all tasks, complemented by a fully connected layer that serves as the task-specific head. All input images are resized to 224×224 pixels. We set the batch size to 64 and employ the cross-entropy loss function across all tasks. The number of upper-level (UL) epochs K is configured to 100. To update the model parameters ω in the UL subproblem, we utilize the Adam optimizer with a learning rate of 10^{-4} and a weight decay of 10^{-5}. .

For the NYUv2 dataset, in line with the approach of RLW [64], we employ the DeepLabV3+ architecture [14], which utilizes a ResNet-50 network with dilated convolutions as a shared encoder across all tasks. Additionally, we incorporate three Atrous Spatial Pyramid Pooling (ASPP) [14] modules as task-specific heads. All input images are resized to 288×384. The batch size is set to 8. The loss functions employed for the three tasks include cross-entropy loss, L_1 loss, and cosine loss, respectively. The total number of UL epochs K is established at 200. An Adam optimizer with a learning rate of

Table 3.5: Mean absolute error (MAE) on the QM9 dataset. 3 independent runs are conducted for each experiment. The average performance is reported. The best results over baselines except STL are marked in **bold**.

Methods	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	$\Delta_{\text{p}}\uparrow$
STL	0.062	0.192	58.82	51.95	0.529	4.52	63.69	60.83	68.33	60.31	0.069	0.00
multi-task learning methods												
EW [146]	0.096	0.286	**67.46**	82.80	4.655	12.4	128.3	128.8	129.2	125.6	0.116	-146.3
UW [53]	0.336	0.382	155.1	144.3	**0.965**	**4.58**	**61.41**	**61.79**	**61.83**	**61.40**	0.116	-92.35
MGDA [103]	0.181	0.325	118.6	92.45	2.411	5.55	103.7	104.2	104.4	103.7	0.110	-103.0
PCGrad [141]	0.104	0.293	75.29	88.99	3.695	8.67	115.6	116.0	116.2	113.8	0.109	-117.8
GradDrop [17]	0.114	0.349	75.94	94.62	5.315	15.8	155.2	156.1	156.6	151.9	0.136	-191.4
GradVac [126]	0.100	0.299	68.94	84.14	4.833	12.5	127.3	127.8	128.1	124.7	0.117	-150.7
CAGrad [66]	0.107	0.296	75.43	88.59	2.944	6.12	93.09	93.68	93.85	92.32	0.106	-87.25
IMTL [70]	0.138	0.344	106.1	102.9	2.595	7.84	102.5	103.0	103.2	100.8	0.110	-104.3
Nash-MTL [89]	0.115	**0.263**	85.54	86.62	2.549	5.85	83.49	83.88	84.05	82.96	0.097	-73.92
RLW [64]	0.112	0.331	74.59	90.48	6.015	15.6	156.0	156.8	157.3	151.6	0.133	-200.9
first-order bi-level optimization methods												
BVFIM [72]	0.107	0.325	73.18	98.97	5.336	21.4	200.1	201.2	201.8	195.5	0.148	-228.5
BOME [67]	0.105	0.318	72.10	88.52	4.984	12.6	138.8	139.4	140.0	136.1	0.124	-164.1
multi-objective bi-level optimization methods												
MOML [133]	**0.083**	0.347	74.87	80.57	3.813	8.64	191.9	192.6	192.8	188.9	0.135	-165.1
MoCo [31]	0.086	0.427	69.60	**79.00**	5.693	10.2	295.5	296.6	297.0	290.1	0.169	-267.6
FORUM [132]	0.104	0.266	85.37	82.15	2.126	6.49	96.97	97.53	97.69	95.88	**0.097**	**-73.36**

10^{-4} and a weight decay of 10^{-5} is utilized to update the model parameters ω in the UL subproblem, with the learning rate being halved after 100 epochs.

For the QM9 dataset, following Nash-MTL [89], we use a graph neural network [39] as the shared encoder, and a linear layer as the task-specific head. The targets of each task are normalized to have zero mean and unit standard deviation. The batch size is set to 128. We use mean squared error (MSE) as the loss function. The total number of UL epochs K is set to 300. An Adam optimizer with a learning rate of 0.001 is used for updating model parameters ω in the UL subproblem. A ReduceLROnPlateau scheduler [91] is used to reduce the learning rate of ω once Δ_{p} on the validation dataset stops improving.

All methods are implemented using the open-source `LibMTL` library [65]. For the proposed FORUM method, we set $\rho = 0.1$, $\beta_k = (k+1)^{-\frac{3}{4}}$, use an SGD optimizer to update $T = 5$ iterations in the LL subproblem and use an Adam optimizer to update the loss weight α in the UL subproblem. The UL step size μ is set to 0.001 for all datasets, and the LL step size η is set to 0.01 for QM9 and 0.1 for other datasets. For the BOME, BVFIM, MOML, and MoCo methods, we use a similar configuration to the FORUM method and perform a grid search for hyperparameters of each method. Specifically, we search LL learning rate η over $\{0.05, 0.1, 0.5\}$ for both four methods, search ρ over $\{0.1, 0.5, 0.9\}$ for

BOME, search β over $\{0.05, 0.1, 0.5, 1\}$ for BVFIM, and set $T = 1$ for MOML and MoCo and $T = 5$ for BOME and BVFIM.

Results. Table 3.3 presents the results for the Office-31 dataset, where FORUM demonstrates superior performance compared to all baseline methods across various categories, as evidenced by its higher average classification accuracy and Δ_{p}. The results for the NYUv2 dataset are summarized in Table 3.4, where it is noteworthy that FORUM is the only method to surpass the performance of the single-task learning (STL) approach in terms of Δ_{p}. Additionally, FORUM excels in both depth estimation and surface normal prediction tasks. In Table 3.5, we examine the QM9 dataset, which poses significant challenges for multi-task learning. Notably, none of the MTL methods outperform the STL, aligning with findings from prior research [89]. However, FORUM once again surpasses all baselines regarding Δ_{p}. These consistent results affirm that FORUM achieves state-of-the-art performance and is more effective than previous MOBLO methods, including MOML and MoCo.

3.8 Other Applications

The proposed MOBLO framework not only can be applied to solving multi-task learning. It also can be used in various learning problems, such as Few-Shot Learning, Neural Architecture Search, and Reinforcement Learning. In this section, we provide detailed formulations of these learning problems under the MOBLO framework. The detailed experiments of these problems are put in [133] and [134].

3.8.1 Few-Shot Learning

Few-Shot Learning (FSL) aims to tackle the problem of training a model with only a few training samples [124]. Recently, FSL is widely studied from the perspective of meta-learning by using prior knowledge in the meta-training process. Most studies in FSL only consider the classification performance. However, in real-world applications, the performance is not the only important factor to consider. For example, we expect FSL models to not only have good performance but also be robust to adversarial attacks [57], which may help improve the generalization of FSL models. In the following, we can see that this setting can naturally be modeled by the proposed MOBLO framework.

Suppose there is a base dataset $\mathscr{D}_{base}$ with a category set $\mathscr{C}_{base}$ and a novel dataset $\mathscr{D}_{novel}$ with a category set $\mathscr{C}_{novel}$, where $\mathscr{C}_{base} \cap \mathscr{C}_{novel} = \emptyset$. The goal of FSL is to adapt

the knowledge learned from $\mathscr{D}_{base}$ to help the learning of $\mathscr{D}_{novel}$. In the i-th meta-training episode, we generate from $\mathscr{D}_{base}$ an N-way k-shot classification task, which consists of a support set $\mathscr{D}_{base}^{s(i)}$ and a query set $\mathscr{D}_{base}^{q(i)}$. For the robustness, we add perturbations generated by the Projected Gradient Descent (PGD) method [57] into each data point in $\mathscr{D}_{base}^{q(i)}$ to generate a perturbed query set $\mathscr{D}_{base}^{q(i),adv}$. The objective function of the FSL model that considers both the performance and the robustness can be formulated as

$$\begin{aligned} \min_{\alpha} \ & \left(\mathscr{L}_F(\omega^{*(i)}(\alpha), \alpha, \mathscr{D}_{base}^{q(i)}), \mathscr{L}_F(\omega^{*(i)}(\alpha), \alpha, \mathscr{D}_{base}^{q(i),adv})\right) \\ \text{s.t.} \quad & \omega^{*(i)}(\alpha) = \arg\min_{\omega} \mathscr{L}_F(\omega, \alpha, \mathscr{D}_{base}^{s(i)}), \end{aligned} \tag{3.34}$$

where ω represents model parameters, α denotes meta-parameters to encode common knowledge that can be transferred to novel tasks, and $\mathscr{L}_F(\omega, \alpha, \mathscr{D})$ denotes the average classification loss of a model with model parameters α and meta-parameters ω on a dataset $\mathscr{D}$. In the UL subproblem of problem (3.34), the first objective measures the classification loss on the query set based on $\omega^{*(i)}(\alpha)$ obtained by solving the LL subproblem and the second objective measures the robustness via the classification performance on the perturbed query set. Problem (3.34) is a general formulation for a robust FSL model under the MOBLO framework, which depends on what α and ω represent.

3.8.2 Neural Architecture Search

Neural Architecture Search (NAS) seeks to automate the design process of neural network architectures. Most NAS methods focus on searching architectures with the best classification accuracy. However, in real-world applications, we need to consider multiple factors in the architecture design. For example, we expect that the searched architecture has good performance, behaves robustly to noises, and consumes low resource. In this case, we formulate NAS with multiple objectives as a case of MOBLO.

By following the DARTS method [69], in an operation space denoted by $\mathscr{O}$, each element is an operation function $o(\cdot)$ and each cell is a directed acyclic graph with N nodes, where each node represents a hidden representation and each edge (i, j) denotes a candidate operation $o(\cdot)$ with a probability $\alpha_o^{(i,j)}$. Therefore, we denote the set of all the edges in all the cells by $\boldsymbol{E}$, and $\alpha = \{\alpha_o^{(i,j)}\}_{(i,j)\in\boldsymbol{E}, o\in\mathscr{O}}$ is a representation of the neural architecture. The entire dataset is split into two parts: a training dataset denoted by $\mathscr{D}_{tr}$ and a validation dataset denoted by $\mathscr{D}_{val}$.

The multi-objective NAS considers three objectives: classification accuracy, adversarial robustness, and the number of parameters. We formulate three corresponding

losses as $\mathscr{L}_N(\omega,\alpha,\mathscr{D}_{val}), \mathscr{L}_N(\omega,\alpha,\mathscr{D}_{val}^{adv})$, and $\mathscr{L}_{nop}(\alpha)$, where $\mathscr{D}_{val}$ denotes the validation dataset and $\mathscr{D}_{val}^{adv}$ denotes the perturbed validation dataset by adding perturbations on each data point, and the objective function under the MOBLO framework is formulated as

$$\begin{aligned} \min_{\alpha} \ & \left(\mathscr{L}_N(\omega^*(\alpha),\alpha,\mathscr{D}_{val}), \mathscr{L}_N(\omega^*(\alpha),\alpha,\mathscr{D}_{val}^{adv}), \mathscr{L}_{nop}(\alpha)\right) \\ \text{s.t.} \quad & \omega^*(\alpha) = \arg\min_{\omega} \ \mathscr{L}_N(\omega,\alpha,\mathscr{D}_{tr}), \end{aligned} \tag{3.35}$$

where ω denotes all the model parameters in the neural network.

In problem (3.35), the loss function $\mathscr{L}_N(\omega,\alpha,\mathscr{D})$ represents the average classification loss for a neural network with parameters ω and architecture α on a dataset $\mathscr{D}$. To formulate $\mathscr{L}_{nop}(\alpha)$, we denote n_o as the number of parameters associated with an operation o, and $N_{nop}(\alpha)$ as the total number of parameters in the searched architecture α. The quantity $N_{nop}(\alpha)$ can be computed using the formula: $N_{nop}(\alpha) = \sum_{(i,j)\in \boldsymbol{E}} n^{(i,j)}$, where $n^{(i,j)}$ denotes the number of parameters of the operation associated with the edge (i,j). Since we determine the operation for each edge by selecting the one with the highest probability, we have: $n^{(i,j)} = n_{\arg\max_{o\in\mathscr{O}} \alpha_o^{(i,j)}}$. However, because the argmax operation is non-differentiable, we approximate $N_{nop}(\alpha)$ using the softmax function. This allows us to express $N_{nop}(\alpha)$ in a differentiable manner, enabling smoother optimization during the search for the optimal architecture. Specifically, we define:

$$\hat{N}_{nop}(\alpha) = \sum_{(i,j)\in \boldsymbol{E}} \sum_{o\in\mathscr{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathscr{O}} \exp(\alpha_{o'}^{(i,j)})} n_o. \tag{3.36}$$

Therefore, to search a network architecture with an expected size L, $\mathscr{L}_{nop}(\alpha)$ can be formulated as

$$\mathscr{L}_{nop}(\alpha) = |\hat{N}_{nop}(\alpha) - L|. \tag{3.37}$$

In the lower-level (LL) subproblem of problem (3.35), once we have the architecture α, we can train a model to obtain optimal parameters ω using the training dataset. In the upper-level (UL) subproblem, our goal is to update α by balancing the validation loss, adversarial robustness, and the total number of parameters. It is easy to see that the DARTS method is a special case of problem (3.35) when its UL subproblem contains the first objective only and hence problem (3.35) generalizes the DARTS method by considering two more objectives.

3.8.3 Reinforcement Learning

Reinforcement learning aims to deal with the problem of how intelligent agents ought to take action in an environment to maximize the notion of cumulative reward. We consider the Multi-Task Reinforcement Learning (MTRL) problem, which aims to improve the performance of agents in multiple tasks and is a promising approach to train effective real-world agents [142]. In the following, we formulate the MTRL problem under the MOBLO framework.

Consider$m$ reinforcement learning tasks, each defined as a policy search problem within the framework of a Markov decision process (MDP) [142]. Each task $\mathcal{T}$ corresponds to an MDP, represented by the tuple (S, A, P, R, H, γ), where S denotes the state space, A indicates the action space, $P(s_{t+1}|s_t, a_t)$ represents the state transition probability of reaching the next state s_{t+1} given the current state s_t and action a_t, $R(s,a)$ is the reward function, H denote the horizon, and γ is the discount factor. The objective of Multi-Task Reinforcement Learning (MTRL) is to learn a policy $\pi(a|s,z)$ that maximizes the expected return across all tasks, formulated as $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[\mathbb{E}_{\pi}[\sum_{t=1}^{T}\gamma^t R_t(s_t,a_t)]]$. In this formulation, z represents an encoding of a specific task, allowing the policy to adapt and optimize performance based on the characteristics of each individual task.

We now introduce the Actor-Critic (AC) method [55] as an example to show how to view a reinforcement learning problem as a bi-level optimization problem. The AC method is a representative reinforcement learning technique that simultaneously learns a policy function and a value function. Consider a parameterized state-action value function $Q(s_t, a_t)$ and a tractable policy π. The AC method first uses the actor π to interact with the environment and to learn the value function Q by minimizing the temporal difference (TD) error. Then it uses the given value function to update the policy network by maximizing the expected discounted cumulative reward. Therefore, the value function can be considered to be optimized with respect to the optimum of the policy function. Thus, the AC method can be viewed as a bi-level optimization problem where the actor and critic correspond to the LL and UL variables, respectively [71].

In MTRL, suppose each task shares the same model, and we consider a parameterized state-action function $Q_\alpha(s_t, a_t)$ and a tractable policy $\pi_\omega(a_t|s_t, z)$, where α denotes parameters of the state-action value-function and ω denotes parameters of the policy network. Then, according to the above discussion, we can formulate the MTRL problem

under the MOBLO framework as

(3.38)
$$\min_{\alpha} \left(\mathscr{L}_Q(\omega^*(\alpha),\alpha;z_1),\ldots,\mathscr{L}_Q(\omega^*(\alpha),\alpha;z_m)\right) \quad \text{s.t.} \quad \omega^*(\alpha)=\arg\min_{\omega}\sum_{i=1}^{m} J_\pi(\omega,\alpha;z_i),$$

where $\mathscr{L}_Q(\omega,\alpha;z_i)$ and $J_\pi(\omega,\alpha;z_i)$ represent the loss function for the state-action function Q and the policy network of the i-th task, respectively, and z_i represents the encoding vector of the i-th task.

We adapt problem (3.38) into the Soft Actor-Critic (SAC) method [44], which is an off-policy actor-critic deep RL algorithm and has been widely used in most MTRL models [108, 142]. For the i-th task, we use $\mathscr{D}_{\pi_i}(s_t,a_t)$ and $\mathscr{D}_{\pi_i}(s_t)$ to denote the state-action marginals and state of the trajectory distribution induced by a policy $\pi(a_t|s_t,z_i)$. The SAC method for the i-th task is to find the policy to maximize $\mathbb{E}_{(s_t,a_t)\sim\mathscr{D}_{\pi_i}}\sum_t[R(s_t,a_t)+\beta_i\mathscr{H}(\pi(\cdot|s_t,z_i))]$, where β_i represents the temperature for task i and $\mathscr{H}(\cdot)$ denotes the entropy function. In SAC method, the soft Q-function parameters can be trained to minimize the soft Bellman residual. Therefore, the objective function of the soft Q-function for the i-th task is formulated as

(3.39)
$$\mathscr{L}_Q(\omega,\alpha;z_i)=\mathbb{E}_{(s_t,a_t)\sim\mathscr{D}_{\pi_i}}\left[\frac{1}{2}\Big(Q_\alpha(s_t,a_t)-\big(R(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim P}[V_{\hat{\alpha}}(s_{t+1},z_i)]\big)\Big)^2\right],$$

where $V(s_t,z_i)=\mathbb{E}_{a_t\sim\pi(a_t|s_t,z_i)}[Q(s_t,a_t)-\beta_i\log\pi(a_t|s_t,z_i)]$ and parameters in $\hat{\alpha}$ are an exponentially moving average of the soft Q-function weights, which is to stabilize training [88]. The policy parameters α can be learned by directly minimizing the expected KL-divergence, and the corresponding objective for the i-th task is formulated as

(3.40)
$$J_\pi(\omega,\alpha;z_i)=\mathbb{E}_{s_t\sim\mathscr{D}_{\pi_i}}\left[D_{\mathrm{KL}}\left(\pi_\omega(\cdot\mid s_t,z_i)\|\frac{\exp(Q_\alpha(s_t\mid\cdot))}{Z_\alpha(s_t)}\right)\right],$$

where $Z_\alpha(s_t)$ denotes the partition function to do the normalization. By following [44], we can minimize the objective $\mathscr{L}_Q$ w.r.t α and J_π w.r.t ω in problem (3.38) separately.

3.9 Summary

In this chapter, we introduce the MOBLO framework for solving the multi-task learning problem from the perspective of task weighting. Specifically, we use a meta-learning approach to effectively learn the task weights in the validation dataset and optimize the MTL model parameter based on the training dataset. To solve the proposed MOBLO framework, we proposed two gradient-based MOBLO methods, MOML and FORUM.

Notably, the MOML method is the first gradient-based MOBLO method. For the proposed two methods, we both provide the convergent analysis and complexity analysis. Since the proposed FORUM method is a first-order method and does not calculate the Hessain product. It is more efficient compared with the MOML method. Moreover, empirical studies on different learning problems demonstrate the proposed FORUM method is effective and efficient. In particular, FORUM achieves state-of-the-art performance on three benchmark datasets under the setting of multi-task learning.

CHAPTER 4

AN ADAPTIVE STOCHASTIC GRADIENT ALGORITHM FOR BLACK-BOX MTL

4.1 Chapter Abstract

This chapter presents a novel black-box multi-objective optimization method called ASMG. The proposed ASMG method is an effective weighting method for solving the black-box MTL problem. To the best of our knowledge, the proposed ASMG method is the first stochastic gradient algorithm for black-box MOO with a theoretical convergence guarantee. We explicitly provide the connection of the Pareto optimal and stationary conditions between the original MOO and the corresponding Gaussian smoothed MOO. Moreover, we prove the convergence rate for the proposed ASMG algorithm in both convex and non-convex cases. Empirically, the proposed ASMG algorithm achieves competitive performances on multiple numerical benchmark problems. Moreover, the state-of-the-art performance on black-box multi-task learning problems demonstrates the effectiveness of the proposed ASMG method.

4.2 Introduction

Multi-objective optimization (MOO) is an influential framework for solving multi-task learning problems and a typical MOO problem can be formulated as

$$\min_{\boldsymbol{x}\in\mathcal{X}} F(\boldsymbol{x}) := (F_1(\boldsymbol{x}), F_2(\boldsymbol{x}), \ldots, F_m(\boldsymbol{x})), \tag{4.1}$$

where $m \geq 2$ denotes the number of objectives, $\mathcal{X} \subseteq \mathbb{R}^d$ and d represents the parameter dimension. The objective function $F_i : \mathbb{R}^d \to \mathbb{R}$ satisfies $F_i(\boldsymbol{x}) \geq -\infty$ for $i = 1, \ldots, m$.

A prominent gradient-based method for solving MOO is the MGDA method. The core concept behind MGDA involves iteratively updating the variable $\boldsymbol{x}$ using a unified descent direction derived from a convex combination of the gradients corresponding to each objective. Numerous MGDA-based MOO algorithms [31, 66, 141, 149] have been developed to refine these multiple gradients, seeking a shared descent direction that simultaneously decreases all the objectives.

However, with the development of large language models, many large models such as large language models (LLMs) [23, 96, 139] are released in the service and are only allowed for access with APIs [8]. In this scenario, we also need to use one single model to handle multiple tasks. In such cases, the gradient of the objective $F(\boldsymbol{x})$ w.r.t. the variable $\boldsymbol{x}$ cannot be explicitly calculated, making the corresponding black-box MTL problem a black-box MOO problem (i.e. problem (4.1)) [123, 150]. In such scenarios, users can only query the large models without accessing gradients to accomplish tasks of interest [114, 116], and gradient-based MOO methods are no longer applicable since they all rely on the availability of true gradients or stochastic gradients w.r.t. the variable $\boldsymbol{x}$.

Several kinds of approaches have been widely studied for black-box MOO, such as Bayesian optimization (BO) [54, 145] and genetic algorithms (GA) [2, 11, 59, 123]. Among those methods, BO methods are good at dealing with low-dimensional expensive black-box MOO problems, while GA is to explore the entire Pareto optimal set, which is computationally expensive for machine learning problems, and usually lacks convergence analysis. Therefore, those limitations motivate us to design an algorithm for black-box MOO that can effectively reach a Pareto optimal solution or a Pareto stationary solution for relatively high-dimensional learning problems with affordable evaluations and convergence guarantee.

To achieve that, in this chapter, we propose a novel Adaptive Stochastic Multi-objective Gradient (ASMG) algorithm for black-box MOO by taking advantage of gradient-based MOO methods. Specifically, the ASMG method first smoothes each objective to their expectation over a Gaussian distribution, leading to Gaussian smoothed objectives.

Then it iteratively updates the parameterized distribution via a common search direction aggregated by the approximated stochastic gradients for all smoothed objectives. We explore the connections between the MOO and the corresponding Gaussian smoothed MOO and provide a convergence analysis for the proposed ASMG algorithm under both convex and non-convex scenarios. Moreover, experiments on various numerical benchmark problems and a black-box multi-task learning problem demonstrate the effectiveness of the proposed ASMG method.

4.3 Preliminary

Notation and Symbols. $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ denote the l_1 norm, l_2 norm, and l_∞ norm for vectors, respectively. $\|\cdot\|_F$ denote the Frobenius norm for matrices. Δ^m denotes an m-dimensional simplex. $\mathscr{S}^+$ denotes the set of positive semi-definite matrices. $\frac{X}{Y}$ denotes the elementwise division operation when X and Y are vectors, and the elementwise division operation for diagonal elements in X and Y when they are diagonal matrices. For a square matrix $\mathbf{X}$, diag($\mathbf{X}$) is a vector with diagonal entries in $\mathbf{X}$, and if $\mathbf{x}$ is a vector, diag($\mathbf{x}$) is a diagonal matrix with $\mathbf{x}$ as its diagonal entries. Define $\|X\|_Y := \sqrt{\langle X, YX\rangle}$ for a matrix $Y \in \mathscr{S}^+$ or a non-negative vector Y, where $\langle\cdot,\cdot\rangle$ denotes the inner product under the l_2 norm for vectors and the inner product under the Frobenius norm for matrices.

Multi-objective Optimization. In MOO [20], we are interested in finding solutions that can not be improved simultaneously for all the objectives, leading to the notion of Pareto optimality, the corresponding definition can be found in Appendix A.1.1. We then present the sufficient condition for determining Pareto optimal for MOO problem (4.1).

Proposition 4.1. *For MOO problem (4.1), if all objective $F_i(\boldsymbol{x})$ for $i = 1,\dots,m$ are convex functions and there exists $\boldsymbol{\lambda} \in \Delta^{m-1}$ such that $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \boldsymbol{\lambda}^\top F(\boldsymbol{x})$, then $\boldsymbol{x}^*$ is a Pareto optimal.*

The above proposition implies that the minimizer of any linearization is Pareto optimal [149]. In the general nonconvex cases, MOO aims to find the Pareto stationary solution [33]. If a point $\hat{\boldsymbol{x}}$ is a Pareto stationary solution, then there is no common descent direction for all $F_i(\boldsymbol{x})$'s ($i = 1,\dots,m$) at $\hat{\boldsymbol{x}}$. For the Pareto stationary condition, we have the following proposition according to Proposition 1 in [149].

Proposition 4.2. *For MOO problem (4.1), (i) we say* $\boldsymbol{x}^* \in \mathscr{X}$ *is a Pareto stationary solution if there exist* $\boldsymbol{\lambda} \in \Delta^{m-1}$ *such that* $\|\sum_{i=1}^m \lambda_i \nabla F_i(\boldsymbol{x}^*)\| = 0$ *; (ii) we say* $\boldsymbol{x}^* \in \mathscr{X}$ *is a* ϵ*-accurate Pareto stationary solution if* $\min_{\boldsymbol{\lambda}} \|\sum_{i=1}^m \lambda_i \nabla \boldsymbol{F}_i(\boldsymbol{x}^*)\|^2 \leq \epsilon$ *where* $\boldsymbol{\lambda} \in \Delta^{m-1}$.

Black-box Optimization. Several kinds of approaches have been studied for black-box optimization, such as Bayesian optimization (BO) [82, 111], evolution strategies (ES) [3, 46], and genetic algorithms (GA) [110]. BO-based methods are inefficient in handling high-dimensional problems and GA methods lack convergence analysis. ES-based methods are better for relatively high-dimensional problems and have been applied in many applications such as reinforcement learning [68] and prompt learning [114, 116]. Although BO achieves good query efficiency for low-dimensional problems, it often fails to handle high-dimensional problems with large sample budgets [28]. The computation of GP with a large number of samples itself is expensive, and the internal optimization of the acquisition functions is challenging.

Among ES-based methods, CMA-ES [46] is a representative one. It uses second-order information to search candidate solutions by updating the mean and covariance matrix of the likelihood of candidate distributions. The CMA-ES method is widely adopted in many learning tasks [45, 114, 116, 128]. Though it is designed for single-objective black-box optimization, it is also applied to black-box multi-task learning [115], where all objectives are aggregated with equal weights. Therefore, we consider CMA-ES as an important baseline method in our experiments.

Inspired by evolution strategy, the stochastic gradient approximation method [81, 127] for black-box optimization, instead of maintaining a population of searching points, iteratively updates a search distribution by stochastic gradient approximation.

The stochastic gradient approximation strategies employed in black-box optimization typically follow a general procedure. Firstly, a parameterized search distribution is utilized to generate a batch of sample points. Then the sample points allow the algorithm to capture the local structure of the fitness function and appropriately estimate the stochastic gradient to update the distribution. Specifically, when $\boldsymbol{\theta}$ denotes the parameters of the search distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $f(\boldsymbol{x})$ denotes a single objective function for sample $\boldsymbol{x}$, the expected fitness under the search distribution can be defined as $J(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}[f(\boldsymbol{x})]$. Based on this definition, we can obtain the Monte Carlo estimate of the search gradient as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{j=1}^{N} f(\boldsymbol{x}_j) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_j), \tag{4.2}$$

where N denotes the number of samples, and $\boldsymbol{x}_j$ denotes the j-th sample. Therefore, the stochastic gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ provides a search direction in the space of search distributions.

4.4 ASMG Algorithm

In this section, we introduce the proposed ASMG algorithm. Firstly, we formulate the black-box MOO as a min-max optimization problem and solve it in Section 4.4.1. Then in Section 4.4.2, we derive the update formula of parameters in the search distribution under the Gaussian sampling.

4.4.1 Black-box Multi-objective Optimization

We aim to minimize the MOO problem (4.1) with only function queries. Due to the lack of gradient information in black-box optimization, we use the stochastic gradient approximation method. Specifically, the objective of the original MOO is smoothed to the expectation of $F(\boldsymbol{x})$ under a parametric search distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ with parameter $\boldsymbol{\theta}$, i.e., $J_i(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}[F_i(\boldsymbol{x})]$ for $i = 1, \ldots, m$. Then the optimal parameter $\boldsymbol{\theta}$ is found by minimizing the following smoothed MOO problem as

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := (J_1(\boldsymbol{\theta}), J_2(\boldsymbol{\theta}), \ldots, J_m(\boldsymbol{\theta})). \tag{4.3}$$

By following [81, 127], the search distribution is assumed to be a Gaussian distribution, i.e., $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu}$ denotes the mean and $\boldsymbol{\Sigma}$ denotes the covariance matrix, and correspondingly $\boldsymbol{\theta}$ includes $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. We denote $J_i(\boldsymbol{\theta}) = J_i(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for $i = 1, \ldots, m$. This Gaussian-smoothed MOO method can effectively estimate stochastic gradients, and enable a more accurate search direction for the distribution to address high-dimensional black-box MOO problems. The connection between this Gaussian smoothed MOO problem (4.3) and problem (4.1) is shown in Section 4.5.

Here we aim to derive an update formulation for $\boldsymbol{\theta}$. To optimize all the Gaussian smoothed objective functions effectively, inspired by MGDA, we can find a parameter $\boldsymbol{\theta}$ to maximize the minimum decrease across all smoothed objectives in each iteration as

$$\max_{\boldsymbol{\theta}} \min_{i \in [m]} (J_i(\boldsymbol{\theta}_t) - J_i(\boldsymbol{\theta})) \approx \max_{\boldsymbol{\theta}} \min_{i \in [m]} \langle \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t - \boldsymbol{\theta} \rangle, \tag{4.4}$$

where $\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}_t}}[F_i(\boldsymbol{x})]$ denotes the derivative of $J_i(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ at $\boldsymbol{\theta}_t = \{\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\}$. In Eq. (4.4), the first-order Taylor expansion is used to derive the approximation with an assumption that the variable $\boldsymbol{\theta}$ is close to $\boldsymbol{\theta}_t$. To further make this assumption

hold, we add a regularization term to maximize $-\frac{1}{\beta_t}\mathrm{KL}(p_{\boldsymbol{\theta}}\|p_{\boldsymbol{\theta}_t})$ into Eq. (4.4), and then the objective function to update $\boldsymbol{\theta}$ is formulated as

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\lambda}^t\in\Delta^{m-1}} \langle \sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\theta}-\boldsymbol{\theta}_t\rangle + \frac{1}{\beta_t}\mathrm{KL}(p_{\boldsymbol{\theta}}\|p_{\boldsymbol{\theta}_t}). \tag{4.5}$$

Note that problem (4.5) is convex w.r.t. $\boldsymbol{\theta}$ and concave w.r.t. $\boldsymbol{\lambda}$ for Gaussian distribution $p_{\boldsymbol{\theta}}$. Then using Von Neumann-Fan minimax theorem [7], we can switch the order of the min and max operators, leading to an equivalent problem as

$$\max_{\boldsymbol{\lambda}^t\in\Delta^{m-1}} \min_{\boldsymbol{\theta}} \langle \sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\theta}-\boldsymbol{\theta}_t\rangle + \frac{1}{\beta_t}\mathrm{KL}(p_{\boldsymbol{\theta}}\|p_{\boldsymbol{\theta}_t}). \tag{4.6}$$

By solving the inner problem of problem (4.6), we can obtain the update formulations for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in the t-th iteration as

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \boldsymbol{\Sigma}_t \sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t), \quad \boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + 2\beta_t \sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t), \tag{4.7}$$

where $\nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)$ and $\nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)$ denote the derivative of $J_i(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ at $\boldsymbol{\mu}=\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_t$, respectively. To obtain those two gradients, in the following theorem, we prove that we only need function queries.

Theorem 4.1. *[127] The gradient of the expectation of an integrable function $F_i(\boldsymbol{x})$ under a Gaussian distribution $p_{\boldsymbol{\theta}} := \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ with respect to the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$ can be expressed as*

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})] = \mathbb{E}_{p_{\boldsymbol{\theta}}}[\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})F_i(\boldsymbol{x})], \tag{4.8}$$

$$\nabla_{\boldsymbol{\Sigma}} \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})] = \frac{1}{2}\mathbb{E}_{p_{\boldsymbol{\theta}}}[\big(\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}-\boldsymbol{\Sigma}^{-1}\big)F_i(\boldsymbol{x})]. \tag{4.9}$$

According to Theorem 4.1, to calculate the gradients, we need to calculate the inverse covariance matrix, which is computationally expensive in high dimensions, and hence to reduce the computational cost, we assume that the covariance matrix $\boldsymbol{\Sigma}$ is a diagonal matrix.

Then substituting Eq. (4.7) into problem (4.6), it can be approximated by the following quadratic programming (QP) problem as

$$\min_{\boldsymbol{\lambda}^t\in\Delta^{m-1}} \Big\| \sum_{i=1}^{m} \lambda_i^t \boldsymbol{p}_i^t \Big\|^2 + 2\Big\| \sum_{i=1}^{m} \lambda_i^t \boldsymbol{h}_i^t \Big\|^2, \tag{4.10}$$

where $\boldsymbol{p}_i^t = \boldsymbol{\Sigma}_t^{\frac{1}{2}} \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)$ and $\boldsymbol{h}_i^t = \mathrm{diag}(\boldsymbol{\Sigma}_t \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t))$. The detailed derivation is put in the Appendix A.2.1. Problem (4.10) is obviously convex and the objective function of problem (4.10) can be simplified to $\boldsymbol{\lambda}^{t\top}\Lambda^\top\Lambda\boldsymbol{\lambda}^t$, where

$$\boldsymbol{\lambda}^t = (\lambda_1^t,\ldots,\lambda_m^t)^\top, \text{and} \quad \Lambda = (((\boldsymbol{p}_i^t)^\top, \sqrt{2}(\boldsymbol{h}_i^t)^\top)^\top,\ldots,((\boldsymbol{p}_m^t)^\top, \sqrt{2}(\boldsymbol{h}_m^t)^\top)^\top). \tag{4.11}$$

The matrix $\Lambda^\top\Lambda$ is of size $m \times m$, which is independent of the dimension of $\boldsymbol{\mu}$. Therefore, the computational cost to solve problem (4.10) is negligible since m is usually very small. Here we use the open-source `CVXPY` library [24] to solve it.

4.4.2 Update Formulations for Gaussian Sampling

Since $\boldsymbol{p}_i^t$ and $\boldsymbol{h}_i^t$ in problem (4.10) and the update formulation of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in Eq. (4.7) need to calculate expectations of the black-box function. However, those expectations do not have analytical forms, and we estimate them by Monte Carlo sampling.

Specifically, according to Theorem 4.1, the stochastic approximation of $\boldsymbol{p}_i^t$ and $\boldsymbol{h}_i^t$ using Monte Carlo sampling are given as

$$\hat{\boldsymbol{p}}_i^t = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{\Sigma}_t^{-\frac{1}{2}}(\boldsymbol{x}_j - \boldsymbol{\mu}_t)\big(F_i(\boldsymbol{x}_j) - F_i(\boldsymbol{\mu}_t)\big), \tag{4.12}$$

$$\hat{\boldsymbol{h}}_i^t = \frac{1}{2N}\sum_{j=1}^{N} \big[\mathrm{diag}\big((\boldsymbol{x}_j - \boldsymbol{\mu}_t)(\boldsymbol{x}_j - \boldsymbol{\mu}_t)^\top\boldsymbol{\Sigma}_t^{-1} - \boldsymbol{I}\big)(F_i(\boldsymbol{x}_j) - F_i(\boldsymbol{\mu}_t))\big], \tag{4.13}$$

where $\boldsymbol{x}_j$ denotes the j-th sample and inspired by [81], subtracting $F_i(\boldsymbol{\mu}_t)$ is used to improve the computational stability while keeping them as unbiased estimations. By incorporating Theorem 4.1 into Eq. (4.7), the updated formulations for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using Monte Carlo sampling are rewritten as

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \sum_{i=1}^{m} \lambda_i^t \boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t, \quad \boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + 2\beta_t \sum_{i=1}^{m} \lambda_i^t \hat{G}_i^t, \tag{4.14}$$

where the stochastic gradients $\hat{\boldsymbol{g}}_i^t$ and $\hat{G}_i^t$ are formulated as

$$\hat{\boldsymbol{g}}_i^t = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{\Sigma}_t^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_t)\big(F_i(\boldsymbol{x}_j) - F_i(\boldsymbol{\mu}_t)\big), \tag{4.15}$$

$$\hat{G}_i^t = \frac{1}{2N}\sum_{j=1}^{N} \mathrm{diag}\Big[\boldsymbol{\Sigma}_t^{-1}\big[\mathrm{diag}\big((\boldsymbol{x}_j - \boldsymbol{\mu}_t)(\boldsymbol{x}_j - \boldsymbol{\mu}_t)^\top\boldsymbol{\Sigma}_t^{-1} - \boldsymbol{I}\big)(F_i(\boldsymbol{x}_j) - F_i(\boldsymbol{\mu}_t))\big]\Big]. \tag{4.16}$$

Note that $\hat{\boldsymbol{g}}_i^t$ is an unbiased estimator for the gradient $\nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)$ as proved in Lemma A.6. To avoid the scaling problem, in practice, we can employ monotonic transformation for the aggregated objective, more details can be found in Appendix A.2.5.

To ensure the convergence, the sequence of weighted vector $\{\boldsymbol{\lambda}^t\}_{t=0}^{T-1}$ is usually required to be a convergent sequence [31, 75, 149]. However, directly solving problem (4.10) in each iteration cannot guarantee that. Moreover, since solving the composite weights $\boldsymbol{\lambda}^t$ depends on $\hat{\boldsymbol{p}}_i^t$ and $\hat{\boldsymbol{h}}_i^t$, which are related to stochastic gradients $\hat{\boldsymbol{g}}_i^t$ and

$\hat{\boldsymbol{G}}_i^t$, the estimation of the composite stochastic gradient may contain some bias, i.e. $\mathbb{E}[\sum_{i=1}^m \lambda_i^t \boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t] \neq \sum_{i=1}^m \mathbb{E}[\lambda_i^t]\mathbb{E}[\boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t]$. To generate a stable composite weights sequence and reduce the bias caused by the correlation of weights and the stochastic gradients, we apply a momentum strategy [149] to $\boldsymbol{\lambda}$. Specifically, in k-th iteration, we first solve problem (4.10) to obtain $\tilde{\boldsymbol{\lambda}}_k$, and then update the weights by

$$\boldsymbol{\lambda}^k = (1-\gamma_t)\boldsymbol{\lambda}^{k-1} + \gamma_t \tilde{\boldsymbol{\lambda}}^k, \tag{4.17}$$

where γ_t is a coefficient and $\gamma_t \in (0,1]$. To preserve the advantage of maximizing the minimum decrease across all the Gaussian smoothed objectives, the coefficient γ_t is set as 1 at the beginning and then decays to 0 as $t \to +\infty$. In Lemma A.7, we show that the bias caused by solving $\boldsymbol{\lambda}^t$ decreases to zero as $\gamma \to 0$.

The complete algorithm is shown in Algorithm 3. Since the computational cost associated with solving problem (4.10) in each iteration is negligible, the computational cost for the ASMG method per iteration is on the order of $\mathcal{O}(mNd)$.

Relationship to Gradient-based MOO. For previous methods on MOO, the most relevant method to our approach is gradient-based MOO methods [31, 66, 141, 149], as they also aim at finding the Pareto optimal solution or the Pareto stationary solution. A typical gradient-based method is the MGDA method [22], which also solves a max-min optimization problem to obtain the weights. However, the proposed ASMG method is not a typical MGDA-type method. The max-min optimization problem proposed in MGDA-type methods is related to the true gradient of the parameters. They add a regularization term $\|d\|^2$ to control the norm of the aggregated gradient. In our case, the update is conducted on a Gaussian distribution, and we need to jointly update the mean and covariance matrix. We use Kullback-Leibler divergence to regularize the distance between two distributions, i.e. θ and θ_t. Therefore, the form of the proposed max-min optimization, i.e. Eq. (4.4), differs from MGDA-type methods, and the solution process is also different. However, our max-min problem can also lead to a simple quadratic programming problem for the aggregation weights computation.

4.5 Convergence Analysis

In this section, we provide a comprehensive convergence analysis for the proposed ASMG method. All the proofs are put in Appendix A.2.3. Firstly, we make a standard assumption for problem (4.3).

Algorithm 3 The ASMG Method

Input: number of iterations T, step size β, number of samples N.

Initialized $\boldsymbol{\theta}_0 = (\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\gamma_0 = 1$;
for $t = 0$ **to** $T-1$ **do**
  Take i.i.d samples $\boldsymbol{z}_j \sim \mathcal{N}(0, I)$ for $j \in \{1, \ldots, N\}$;
  Set $\boldsymbol{x}_j = \boldsymbol{\mu}_t + \boldsymbol{\Sigma}_t^{\frac{1}{2}} \boldsymbol{z}_j$ for $j \in \{1, \ldots, N\}$;
  Query the batch observations $\{F_1(\boldsymbol{x}_1), \ldots, F_1(\boldsymbol{x}_N), \ldots, F_m(\boldsymbol{x}_1), \ldots, F_m(\boldsymbol{x}_N)\}$;
  Query the batch observations $\{F_1(\boldsymbol{\mu}_t), \ldots, F_m(\boldsymbol{\mu}_t)\}$;
  Compute $\hat{\boldsymbol{p}}_i^t = \frac{1}{N} \sum_{j=1}^N \boldsymbol{\Sigma}_t^{-\frac{1}{2}} (\boldsymbol{x}_j - \boldsymbol{\mu}_t) \big(F_i(\boldsymbol{x}_j) - F_i(\boldsymbol{\mu}_t)\big)$;
  Compute $\hat{\boldsymbol{h}}_i^t = \frac{1}{2N} \sum_{j=1}^N [\mathrm{diag}\big((\boldsymbol{x}_j - \boldsymbol{\mu}_t)(\boldsymbol{x}_j - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1} - \boldsymbol{I}\big)(F_i(\boldsymbol{x}_j) - F_i(\boldsymbol{\mu}_t))]$;
  Compute $\tilde{\boldsymbol{\lambda}}^t$ by solving the QP problem (4.10);
  Update the weights for the gradient composition $\boldsymbol{\lambda}^t = (1 - \gamma_t)\boldsymbol{\lambda}^{t-1} + \gamma_t \tilde{\boldsymbol{\lambda}}^t$;
  Compute the stochastic gradients $\hat{\boldsymbol{g}}_i^t$ and $\hat{\boldsymbol{G}}_i^t$ according to Eqs. (4.15) (4.16), respectively;
  Set $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \beta_t \sum_{i=1}^m \lambda_i^t \boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t$ and set $\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + 2\beta_t \sum_{i=1}^m \lambda_i^t \hat{\boldsymbol{G}}_i^t$;
end for
return $\boldsymbol{\theta}_T = (\boldsymbol{\mu}_T, \boldsymbol{\Sigma}_T)$.

Assumption 4.1. *The functions $J_i(\boldsymbol{\theta}), \ldots, J_m(\boldsymbol{\theta})$ are H-Lipschitz and L-smoothness w.r.t. $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \in \Theta$, where $\Theta := \{\boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \boldsymbol{\mu} \in \mathbb{R}^d, \boldsymbol{\Sigma} \in \mathscr{S}^+\}$. The functions $F_i(\boldsymbol{x})$ $(i = 1, \ldots, m)$ are H-Lipschitz w.r.t. $\boldsymbol{x}$.*

The smoothness assumption in Assumption 4.1 is widely adopted in MOO [31, 149]. Then, we provide a boundedness result for the covariance matrix $\boldsymbol{\Sigma}$.

Theorem 4.2. *Suppose that the gradient $\hat{\boldsymbol{G}}_i$ are positive semi-definite matrix, i.e., $\hat{\boldsymbol{G}}_i \succeq \xi \boldsymbol{I}$ for $i = 1, \ldots, m$, where $\xi \geq 0$ and that the covariance matrix is a diagonal matrix. Then for Algorithm 3, we have $\boldsymbol{\Sigma}_T \preceq \frac{\boldsymbol{I}}{\xi \sum_{t=1}^T \beta_t + \boldsymbol{\Sigma}_0^{-1}}$.*

Theorem 4.2 establishes the upper bound for $\boldsymbol{\Sigma}$ throughout the optimization process and is useful to analyze the convergence properties in the non-convex scenario as shown in Section 4.5.2.

4.5.1 Convex Cases

In this section, we assume that each objective in problem (4.1), i.e., $F_i(\mathbf{x})$ $(i = 1, \ldots, m)$, is convex w.r.t. $\boldsymbol{x}$. Note that the proposed ASMG algorithm approximates the gradients of the objectives of the Gaussian smoothed MOO problem, i.e., problem (4.3). It is necessary to study the relation between the optimal solutions of the original MOO problem (4.1)

and the corresponding Gaussian-smoothed MOO problem (4.3), and we put the results in the following proposition.

Proposition 4.3. *Suppose $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a Gaussian distribution with $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and the functions $F_i(\boldsymbol{x})$, $i = 1, \ldots, m$ are all convex functions. Let $J_i(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})]$. Then for any $\boldsymbol{\lambda} \in \Delta^{m-1}$ and $\boldsymbol{\mu}^* \in \mathscr{X}$, we have $\sum_{i=1}^m \lambda_i \big(F_i(\boldsymbol{\mu}) - F_i(\boldsymbol{\mu}^*)\big) \leq \sum_{i=1}^m \lambda_i \big(J_i(\boldsymbol{\mu}, \boldsymbol{\Sigma}) - J_i(\boldsymbol{\mu}^*, \mathbf{0})\big)$, where $\mathbf{0}$ denotes a zero matrix with appropriate size and $J_i(\boldsymbol{\mu}^*, \mathbf{0}) = F_i(\boldsymbol{\mu}^*)$.*

When $\boldsymbol{\mu}^*$ is a Pareto-optimal solution of problem (4.1), Proposition 4.3 implies that the distance to the Pareto-optimal objective values of the original MOO problem is upper-bounded by that of the Gaussian smoothed MOO problem. Then the following theorem captures the convergence of $\boldsymbol{\mu}$ for convex objective functions.

Theorem 4.3. *Suppose that $F_i(\boldsymbol{x})$ $(i = 1, \ldots, m)$ is a convex function, $J_i(\boldsymbol{\theta})$ is c-strongly convex w.r.t. $\boldsymbol{\mu}$, $\hat{\boldsymbol{G}}_i$ is positive semi-definite matrix such that $\xi \boldsymbol{I} \preceq \hat{\boldsymbol{G}}_i \preceq \frac{c\boldsymbol{I}}{4}$ with $\xi \geq 0$, $\boldsymbol{\Sigma}_0 \in \mathscr{S}^+$, and $\boldsymbol{\Sigma}_0 \preceq R\boldsymbol{I}$ where $R > 0$. If $\beta \leq \frac{1}{L}$ and the sequence $\{\boldsymbol{\mu}_t\}$ generated by Algorithm 3 satisfies that the distance between the sequence $\{\boldsymbol{\mu}_t\}$ and the Pareto set is bounded, i.e., $\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\| \leq D$ where $\boldsymbol{\mu}^*$ denotes a Pareto optimal solution of problem (4.1), then with Assumption 4.1, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{z}} \left[\sum_{i=1}^m \lambda_i^t \big(J_i(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_t) - J_i(\boldsymbol{\mu}^*, 0)\big) \right] = \mathscr{O}\left(\frac{1}{\beta T} + \frac{\log T}{T} + \gamma\right). \tag{4.18}$$

Based on Theorem 4.3 and Proposition 4.3, when $\beta = \mathscr{O}(1)$ and $\gamma = \mathscr{O}(T^{-1})$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{z}} \left[\sum_{i=1}^m \lambda_i^t \big(F_i(\boldsymbol{\mu}_{t+1}) - F_i(\boldsymbol{\mu}^*)\big) \right] = \mathscr{O}(\frac{\log T}{T}). \tag{4.19}$$

Therefore, the proposed ASMG algorithm possesses a convergence rate $\mathscr{O}(\frac{\log T}{T})$ in convex cases. Note that Theorem 4.3 does not require each objective function $F_i(\boldsymbol{x})$ to be differentiable. Hence, Theorem 4.3 holds for non-smooth convex functions $\{F_i(\boldsymbol{x})\}$. If $F_i(\boldsymbol{x})$ is c-strongly convex, then $J_i(\boldsymbol{\theta})$ is also c-strongly convex and Theorem 4.3 holds.

4.5.2 Non-Convex Cases

In many practical problems, the objective functions of problem (4.1) are non-convex, and we aim to find a Pareto stationary solution. Similar to Proposition 4.3, we have the following result to reveal the relation between Pareto stationary solutions of problems (4.1) and (4.3).

Proposition 4.4. *Suppose $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a Gaussian distribution with $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and $F_i(\boldsymbol{x})$ $(i = 1, \ldots, m)$ is a L_F-Lipschitz smooth function. Let $J_i(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})]$ and $\boldsymbol{\Sigma}$ be a diagonal matrix. If $\boldsymbol{\mu}^*$ is a Pareto stationary solution of problem (4.3) and there exists $\boldsymbol{\lambda} \in \Delta^{m-1}$ such that $\|\sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\mu}^*)\| = 0$, then we have $\|\sum_{i=1}^m \lambda_i \nabla F_i(\boldsymbol{\mu}^*)\|^2 \leq L_F^2 \|\mathrm{diag}(\boldsymbol{\Sigma})\|_1$ and this implies that $\boldsymbol{\mu}^*$ is a ϵ-accurate Pareto stationary solution of problem (4.1) with $\epsilon = L_F^2 \|\mathrm{diag}(\boldsymbol{\Sigma})\|_1$.*

According to Proposition 4.4, a Pareto stationary solution of problem (4.3) is a ϵ-accurate Pareto stationary solution of problem (4.1). The following theorem establishes the convergence of the proposed ASMG method under the non-convex case.

Theorem 4.4. *Suppose that $J_i(\boldsymbol{\theta})$ $(i = 1, \ldots, m)$ is bounded, i.e., $|J_i(\boldsymbol{\theta})| \leq B$, $\hat{G}_i$ is positive semi-definite matrix such that $\xi \boldsymbol{I} \preceq \hat{G}_i \preceq b\boldsymbol{I}$ with $b \geq \xi > 0$, $\boldsymbol{\Sigma}_0 \in \mathscr{S}^+$, and $\boldsymbol{\Sigma}_0 \preceq R\boldsymbol{I}$ with $R > 0$. If $\beta \leq \frac{1}{RL\sqrt{d}}$, then with Assumption 4.1 we have*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{z}} \left[\left\| \sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t) \right\|^2 \right] = \mathscr{O}\left(\frac{\gamma}{\beta} + \frac{1}{\beta T} + \gamma + \beta \right). \tag{4.20}$$

According to Theorem 4.4, if $\beta = \mathscr{O}(T^{-\frac{1}{2}})$ and $\gamma = \mathscr{O}(T^{-1})$, the proposed ASMG method possesses a $\mathscr{O}(T^{-\frac{1}{2}})$ convergence rate to reach a Pareto stationary solution for problem (4.3), which is a ϵ-accurate Pareto stationary solution of problem (4.1). According to Theorem 4.2, when $\beta = \mathscr{O}(T^{-\frac{1}{2}})$, diagonal entries of $\boldsymbol{\Sigma}_T$ converge to zero as $T \to \infty$ and hence $\epsilon \to 0$, leading to a Pareto stationary solution for problem (4.1).

4.6 Empirical Study

In this section, we empirically evaluate the proposed ASMG method on different problems. The experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU.

4.6.1 Synthetic Problems

We compare the proposed ASMG method with CMA-ES [46], ES [102], BES [36], and MMES [47] methods on the following three d-dimensional synthetic benchmark test

problems:

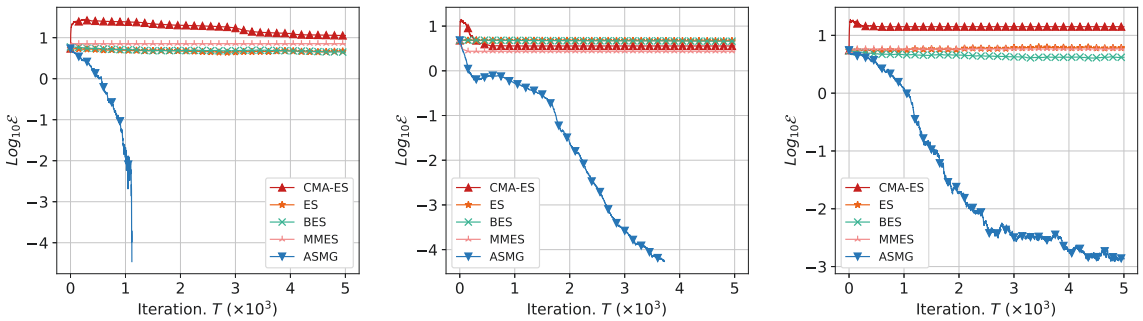
$$F(\boldsymbol{x}) = \Big(\sum_{i=1}^{d} 10^{\frac{2(i-1)}{d-1}} |\boldsymbol{x}_i - 0.01|, \sum_{i=1}^{d} 10^{\frac{2(i-1)}{d-1}} |\boldsymbol{x}_i + 0.01| \Big), \tag{4.21}$$

$$F(\boldsymbol{x}) = \Big(\sum_{i=1}^{d} |\boldsymbol{x}_i - 0.1|^{\frac{1}{2}}, \sum_{i=1}^{d} |\boldsymbol{x}_i + 0.1|^{\frac{1}{2}} \Big), \tag{4.22}$$

$$F(\boldsymbol{x}) = \Big(\sum_{i=1}^{d} 10^{\frac{2(i-1)}{d-1}} |\boldsymbol{x}_i|^{\frac{1}{2}}, 10d + \sum_{i=1}^{d} \big((10^{\frac{(i-1)}{d-1}} \boldsymbol{x}_i)^2 - 10\cos(2\pi 10^{\frac{(i-1)}{d-1}} \boldsymbol{x}_i) \big) \Big). \tag{4.23}$$

Test problems (4.21)-(4.23) are called the shift l_1-ellipsoid, shift $l_{\frac{1}{2}}$-ellipsoid, and mixed ellipsoid-rastrigin 10, respectively. For the baseline methods, by following [115], we aggregate multiple objectives with equal weights to become a single objective.

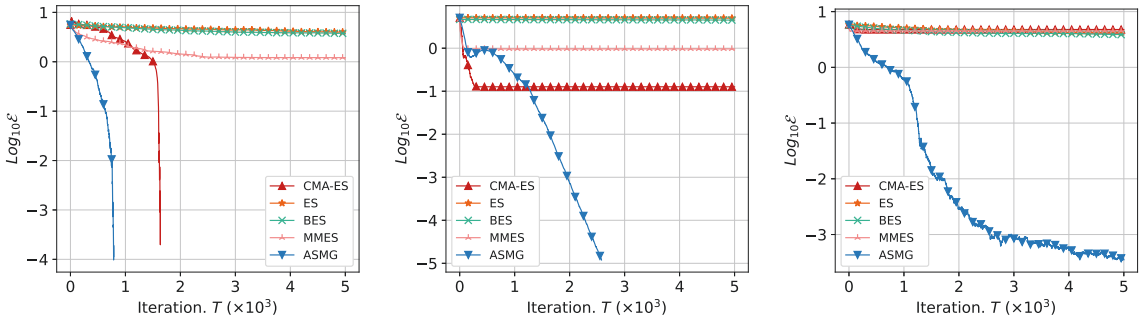
Evaluation Metrics. The results are evaluated by calculating the Euclidean distance between the solution $\boldsymbol{x}$ and the optimal solution set $\mathscr{P}$, i.e., $\mathscr{E} = \text{dist}(\boldsymbol{x}, \mathscr{P})$. The Pareto optimal set of problem (4.21) is $\mathscr{P}_1 = \{\boldsymbol{x} \mid x_i \in [-0.01, 0.01]\}$. Therefore. the result is evaluated by calculating the Euclidean distance between solution $\boldsymbol{x}$ and the set $\mathscr{P}_1$, which is denoted $\mathscr{E} = \text{dist}(\boldsymbol{x}, \mathscr{P}_1)$. We denote the Pareto optimal set of problem (4.22) as $\mathscr{P}_2$. Since the Pareto front of problem (4.22) is concave, the solution of the ASMG method will go to the boundary of its Pareto optimal set, i.e. $\hat{\mathscr{P}}_2 = \{\boldsymbol{x} \mid x_i \in \{-0.1, 0.1\}\} \subset \mathscr{P}_2$. Moreover, For ES and CMA-ES methods, since their optimization objective is $F_1(x)+F_2(x)$, the corresponding solution set is $\hat{\mathscr{P}}_2$. Therefore, the result of these three methods on problem (4.22) is evaluated by $\mathscr{E} = \text{dist}(\boldsymbol{x}, \hat{\mathscr{P}}_2)$. The Pareto optimal set of problem (4.23) is $\mathscr{P}_3 = \{\boldsymbol{x} \mid \boldsymbol{x} = 0\}$. Therefore, the result is evaluated by $\mathscr{E} = \text{dist}(\boldsymbol{x}, \mathscr{P}_3)$.



(a) Shift l_1-Ellipsoid. (b) Shift $l_{\frac{1}{2}}$-Ellipsoid. (c) Mixed Ellipsoid-Rastrigin 10.

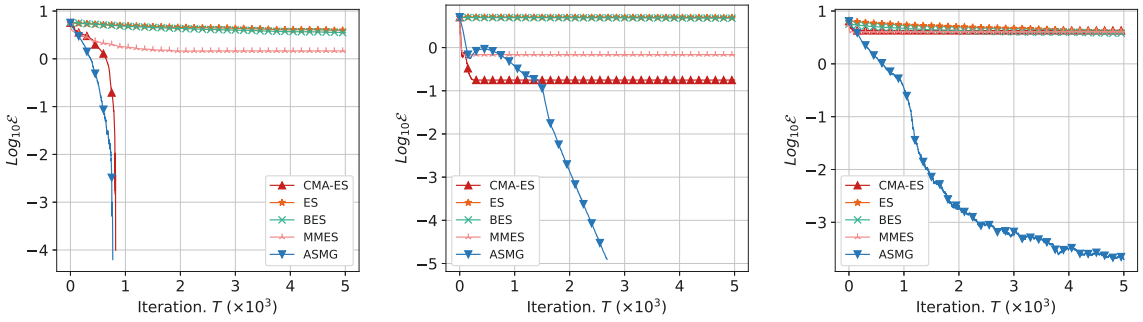
Figure 4.1: Results on the synthetic problems with 10 samples (i.e., $N = 10$).

Implementation Details. For all the methods, we initialize $\boldsymbol{\mu}_0$ from the uniform distribution Uni$[0, 1]$, and set $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$. The ASMG method uses a fixed step size of $\beta = 0.1$



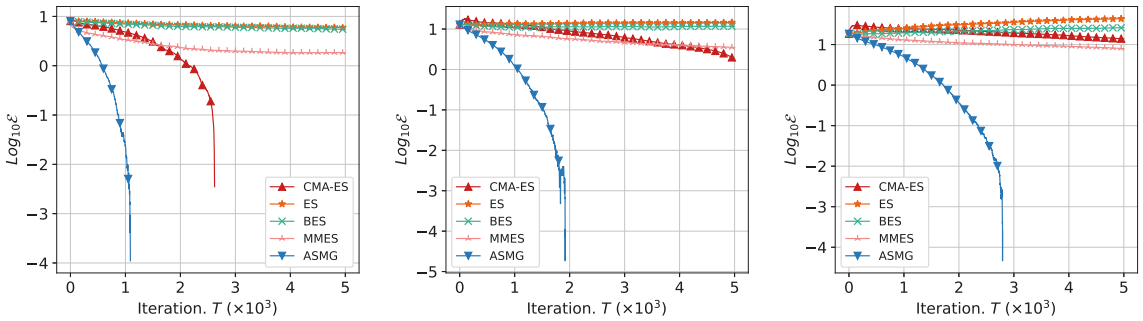
(a) Shift l_1-Ellipsoid. (b) Shift $l_{\frac{1}{2}}$-Ellipsoid. (c) Mixed Ellipsoid-Rastrigin 10.

Figure 4.2: Results on the synthetic problems with 50 samples (i.e., $N = 50$).



(a) Shift l_1-Ellipsoid. (b) Shift $l_{\frac{1}{2}}$-Ellipsoid. (c) Mixed Ellipsoid-Rastrigin 10.

Figure 4.3: Results on the synthetic problems with 100 samples (i.e., $N = 100$).



(a) d=200. (b) d=500. (c) d=1000.

Figure 4.4: Results on the shift l_1-ellipsoid problem with $N = 100$ and different problem dimension d.

and $\gamma_t = 1/(t+1)$. For the ES method, we employ the default step size from [102], i.e., $\beta = 0.01$. For the BES method, we adopt the default step size from [36], i.e., $\beta = 0.01$. We employ the default hyperparameter setting from [47] for the MMES method. We then assess these methods using varying sample sizes, i.e. $N \in \{10, 50, 100\}$. The mean value of $\mathcal{E}$ over 3 independent runs is reported.

Result. Figure 4.2 shows the results on those three d-dimensional synthetic problems with 50 samples (i.e., $N = 50$) and $d = 100$. The proposed ASMG method approximately achieves a linear convergence rate in the logarithm scale and can arrive at solutions with a high precision, i.e., 10^{-4}, on three cases. The CMA-ES method can converge with high precision on problem (4.21) but only achieve 10^{-1} precision on problem (4.22). The MMES method also cannot reach a high precision on these problems. Moreover, the CMA-ES and MMES methods fail on problem (4.23) and the ES and BES methods fail on all three problems. The results show that it could be challenging for ES and BES to optimize non-smooth or non-convex test functions without adaptively updating mean and covariance. Those results consistently demonstrate the effectiveness of the proposed ASMG method. Figure 4.1 and 4.3 show the results on three 100-dimensional synthetic problems with sample sizes $N = 10$ and $N = 100$, respectively. Combining these results with the result from Figure 4.2, we observe consistent performance from the proposed ASMG method, consistently achieving high precision across all three cases, i.e. 10^{-4}, for $N = 10, 50, 100$.

The CMA-ES method shows convergence with high precision on the Shift l_1-Ellipsoid problem when $N = 10$ and 50. However, it fails to converge when the sample size is very small, i.e., $N = 10$. The same performance also occurs on the Shift $l_{\frac{1}{2}}$-Ellipsoid problem for the CMA-ES method. It can achieve 10^{-1} precision when $N = 50$ and 100, but only achieve 10^{1} precision when $N = 10$. It still fails on the Mixed Ellipsoid-Rastrigin10 problem when $N = 50$ and 100. The MMES method also cannot reach a high precision on these problems. The ES and BES methods do not converge in any of the settings, indicating that it could be challenging for these methods to optimize these non-smooth or non-convex problems. These results show the effectiveness of the proposed ASMG method.

Figure 4.4 presents the results for the shift $l_{\frac{1}{2}}$-ellipsoid problem with a sample size of $N = 100$ across various problem dimensions, i.e. $d \in \{200, 500, 1000\}$. The CMA-ES method can still converge when $d = 200$, but it does not converge when $d = 1000$. In contrast, the ASMG method consistently achieves high precision across all three settings, demonstrating its effectiveness in handling high-dimensional problems.

4.6.2 Black-box Multi-task Learning

In this section, we apply the proposed ASMG method to black-box multi-task learning. Multi-task learning (MTL) [10, 146] is a widely adopted paradigm and aims to train a single model to handle multiple tasks simultaneously. MTL can commonly be viewed as

a MOO problem by treating each task as an individual objective. Given m tasks, task i has a training dataset $\mathscr{D}_i$. Let $\mathscr{L}_i(\mathscr{D}_i;\mathscr{M}_\Phi)$ denote the average loss on $\mathscr{D}_i$ for task i using the model $\mathscr{M}$ with parameter Φ. Then MTL can be formulated as a MOO problem with m objectives as

$$\min_{\Phi}(\mathscr{L}_1(\mathscr{D}_1;\mathscr{M}_\Phi),\mathscr{L}_2(\mathscr{D}_2;\mathscr{M}_\Phi),\ldots,\mathscr{L}_m(\mathscr{D}_m;\mathscr{M}_\Phi)). \tag{4.24}$$

For the conventional MTL setting, the model $\mathscr{M}$ is available for backward propagation, allowing the optimization problem to be solved using the gradients of the model parameters, i.e., $\nabla_\Phi \mathscr{L}_i$. However, in many practical scenarios, such as multi-task prompt tuning for extremely large pre-trained models [115], part of the model $\mathscr{M}$ remains fixed in the service and is only accessible through an inference API. This results in the gradient of the objectives $\mathscr{L}_i$ with respect to the local parameters $\phi \subset \Phi$ being unavailable. For cases where the gradients of task losses with respect to the learned parameter ϕ cannot be explicitly calculated in MTL, we refer to black-box MTL.

Problem Formulation. We consider a specific black-box MTL problem, where our focus is to learn a shared prompt for all tasks using pre-trained vision-language models [76, 115, 125]. Following the setup in [76], we employ the CLIP model [95] as the base model. In this context, our model $\mathscr{M}$ can be expressed as $\mathscr{M} = \{\mathscr{M}_c, \boldsymbol{p}\}$, where $\mathscr{M}_c$ represents the fixed CLIP model, and $\boldsymbol{p} \in \mathbb{R}^D$ is the token embedding of the prompt. Note that the CLIP model is treated as a black-box model, making it impossible to calculate the gradient of the token embedding $\boldsymbol{p}$ in the text encoder using backward propagation. Inspired by [114, 116], we optimize $\boldsymbol{v} \in \mathbb{R}^d$ and employ a fixed randomly initialized matrix $\boldsymbol{A} \in \mathbb{R}^{D\times d}$ to project $\boldsymbol{v}$ onto the token embedding space instead of directly optimizing the prompt $\boldsymbol{p}$. Consequently, the corresponding black-box MTL problem can be formulated as

$$\min_{\boldsymbol{v}\in\mathbb{R}^d}\big(\mathscr{L}_1(\mathscr{D}_1;\{\mathscr{M}_c,\boldsymbol{A}\boldsymbol{v}\}),\ldots,\mathscr{L}_m(\mathscr{D}_m;\{\mathscr{M}_c,\boldsymbol{A}\boldsymbol{v}\})\big). \tag{4.25}$$

Details of CLIP. CLIP is a widely adopted vision-language model that trains an image encoder $h_{\text{image}}(\cdot)$ and a text encoder $h_{\text{text}}(\cdot)$ jointly by aligning the embedding space of images and text. Given an image $\boldsymbol{x}$ and a set of class names $\{y_i\}_{i=1}^K$, CLIP obtain image features $h_{\text{image}}(\boldsymbol{x})$ and a set of text features $\{h_{\text{text}}(\boldsymbol{p};\boldsymbol{y_i})\}_{i=1}^K$ where $\boldsymbol{p} \in \mathbb{R}^D$ represents the token embedding of the shared prompt. The image $\boldsymbol{x}$ is classified into the class y_i that corresponds to the highest similarity score $h_{\text{image}}(\boldsymbol{x}) \cdot h_{\text{text}}(\boldsymbol{p};\boldsymbol{y}_i)$ among the cosine similarities between the image features and all the text features. In the zero-shot setup, the shared token embedding $\boldsymbol{p}$ is transformed from the prompt "a photo of a", while in

the prompt tuning setup, the token embedding of the shared prompt is optimized directly to enhance performance.

Loss function $\mathscr{L}_i$. In the context of multi-task learning, we consider a scenario involving m tasks, each having its own dedicated training dataset. For task i, we have dataset $\mathscr{D}_i = \{(\boldsymbol{x}_k, \hat{\boldsymbol{y}}_k)\}$. For each training epoch, we sample a mini-batch $\mathscr{B}_i$ from $\mathscr{D}_i$ and the function $\mathscr{L}_i$ in Eq. (4.25) can be formulated as

$$\mathscr{L}_i(B_i; \{\mathscr{M}_c, \boldsymbol{A}\boldsymbol{v}\}) = \sum_{(\boldsymbol{x}, \bar{\boldsymbol{y}}) \in \mathscr{B}_i} \ell(\mathscr{M}_c(\boldsymbol{A}\boldsymbol{v}; \boldsymbol{x}), \bar{\boldsymbol{y}}),$$

where ℓ can be cross-entropy function for classification problem and $\mathscr{M}_c$ denotes CLIP model in our setting.

Datasets. We conduct experiments on two MTL benchmark datasets, i.e., *Office-31* [101] and *Office-home* [122]. The *Office-31* dataset includes images from three different sources: Amazon (**A**), digital SLR cameras (**D**), and Webcam (**W**). It contains 31 categories for each source and a total of 4652 labeled images. The *Office-home* dataset includes images from four sources: artistic images (**Ar**), clip art (**Cl**), product images (**Pr**), and real-world images (**Rw**). It contains 65 categories for each source and a total of 15,500 labeled images. For those two datasets, we treat the multi-class classification problem on each source as a separate task.

Implementation Details. Following the setup of [148], we conduct experiments based on the CLIP model with ResNet-50 as the image encoder and use a prompt with 4 tokens for the text encoder where both the image and text encoders are kept frozen during the experiments. For zero-shot, we apply the default prompt "a photo of a {class}".

For all methods, we set $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$ as initialization. For all baseline methods except the zero-shot setting, we optimize the prompt with a batch size of 64 for 200 epochs, the population size N is set as 20 for *Office-31* and 40 for *Office-home*, while $\boldsymbol{A}$ is sampled from the normal distribution as described in the [114], i.e. $\mathscr{N}(0, \frac{\sigma_e}{\sqrt{d}})$, where σ_e is the standard deviation of word embeddings in CLIP. For ASMG and ASMG-EW methods, the step size is fixed as $\beta = 0.5$. The coefficient γ in the ASMG method is set as $\gamma_t = 1/(t+1)$. For the ES method, we employ the default step-size of ES in [102], i.e., 0.01. For the BES method, we perform grid search on step size, i.e., β is chosen from $\{0.5, 0.1, 0.01\}$. For the MMES method, we employ the default hyperparameter setting from [47]. Additionally, we evaluate the performance of the method on different

dimensions of $\boldsymbol{z}$, specifically $d \in \{256, 512, 1024\}$. The CMA-ES method is implemented using the official implementation available [1] while we implement the ES and BES method by ourselves.

Baselines. The proposed ASMG method is compared with the zero-shot setting, which evaluates the model on the downstream datasets that were not seen during the training phase without prompt tuning [95]; four ES-based black-box optimization methods, i.e., ES [102], BES [36], MMES [47] and CMA-ES [46], where we simply transform multiple objectives into one single objective by equal weights; the ASMG-EW method, where we fix the weighted vector as $\boldsymbol{\lambda}^t = \frac{\mathbf{1}}{m}$ for ASMG during optimization.

Table 4.1: Results on the *Office-31* and *Office-home* datasets. 3 independent runs are conducted for each experiment. The mean classification accuracy (%) is reported. The best result across all groups is in **bold** and the best result in each comparison group is underlined.

Method	*Office-31*				*Office-home*				
	A	**D**	**W**	**Avg**	**Ar**	**Cl**	**Pr**	**Rw**	**Avg**
Zero-shot	73.68	79.51	66.67	73.28	73.06	51.68	83.79	81.41	72.48
	Dimension $d = 256$								
ES	75.10	80.05	71.67	75.61 $_{\pm 1.18}$	71.16	47.96	80.33	80.90	$70.09_{\pm 0.51}$
BES	72.65	80.60	73.52	$75.59_{\pm 0.90}$	68.94	45.97	81.53	79.42	$68.97_{\pm 1.40}$
MMES	75.90	83.33	76.67	$78.63_{\pm 0.59}$	71.85	49.26	82.10	81.37	$71.14_{\pm 0.69}$
CMA-ES	76.24	87.98	75.93	$80.05_{\pm 1.34}$	69.26	50.09	85.73	82.13	$71.80_{\pm 0.22}$
ASMG-EW	76.52	83.88	77.22	$79.21_{\pm 1.20}$	70.02	47.13	80.23	79.50	$69.22_{\pm 1.39}$
ASMG	77.83	86.61	80.56	**81.67**$_{\pm 0.64}$	74.26	53.52	86.23	83.03	**74.26** $_{\pm 1.06}$
	Dimension $d = 512$								
ES	75.95	81.69	75.37	77.67 $_{\pm 0.91}$	70.78	48.39	82.06	81.15	$70.60_{\pm 0.36}$
BES	75.73	82.51	74.81	$77.68_{\pm 1.88}$	69.39	48.18	82.94	80.29	$70.20_{\pm 0.51}$
MMES	76.01	84.70	77.22	$79.31_{\pm 0.34}$	70.40	50.45	85.10	82.56	$72.13_{\pm 1.19}$
CMA-ES	76.75	87.16	77.22	$80.38_{\pm 0.48}$	70.46	50.02	86.26	82.02	$72.19_{\pm 0.27}$
ASMG-EW	78.01	84.70	76.67	$79.79_{\pm 1.45}$	69.20	46.91	80.51	80.68	$69.33_{\pm 0.52}$
ASMG	78.63	87.43	78.33	$81.47_{\pm 0.37}$	73.50	52.84	85.88	83.78	$74.00_{\pm 0.81}$
	Dimension $d = 1024$								
ES	72.59	78.14	74.81	$75.18_{\pm 1.91}$	70.34	47.38	82.59	80.54	$70.21_{\pm 0.08}$
BES	72.14	79.51	71.67	$74.44_{\pm 0.84}$	70.27	48.25	79.94	80.00	$69.62_{\pm 0.81}$
MMES	77.09	81.42	75.74	$78.09_{\pm 0.95}$	71.03	49.19	84.29	81.95	$71.61_{\pm 0.41}$
CMA-ES	76.87	87.16	77.59	$80.54_{\pm 0.41}$	71.28	50.92	85.73	82.49	$72.61_{\pm 0.39}$
ASMG-EW	77.15	82.51	77.78	$79.15_{\pm 1.48}$	69.20	47.09	81.36	80.86	$69.63_{\pm 0.88}$
ASMG	76.30	87.70	80.19	$81.40_{\pm 0.49}$	73.18	51.82	85.84	83.21	$73.51_{\pm 0.07}$

Results. Table 4.1 presents experimental results on the *Office-31* and *Office-home* datasets for three different dimensions of $\boldsymbol{z}$. We can see that the ASMG method con-

[1] `https://github.com/CMA-ES/pycma`

sistently outperforms all baselines in terms of average classification accuracy across different settings, highlighting its effectiveness. When comparing ASMG with ASMG-EW, the results demonstrate the effectiveness of adaptive stochastic gradient. Notably, even in the high-dimensional setting (i.e., $d = 1024$), our method maintains good performance. Remarkably, ASMG achieves the highest average classification accuracy when $d = 256$, surpassing zero-shot by 8.4% on *Office-31* and 1.8% on *Office-home*. This further validates the effectiveness of the proposed ASMG method.

4.7 Summary

In this chapter, to solve the black-box MTL problem, we consider it as a black-box MOO and present ASMG, a novel and effective adaptive stochastic gradient-based method. Specifically, we smooth the black-box MOO problem to a Gaussian smoothed MOO and we propose a novel adaptive stochastic gradient approximation approach to solve it. Theoretically, we explore the connections between the MOO and the corresponding Gaussian smoothed MOO, and we provide a convergence guarantee for ASMG under both convex and non-convex scenarios. Moreover, empirical studies on synthetic problems and black-box MTL demonstrate the effectiveness of the proposed ASMG method.

CHAPTER 5

A NEURAL ODE-BASED MODEL FOR MTL

5.1 Chapter Abstract

In this chapter, we present a Neural Ordinal diffeRential equation based Multi-tAsk Learning (**NORMAL**) model. In the proposed NORMAL method, feature transformations of different tasks, which are placed after the shared feature extractor in the HPS architecture, are assumed to follow a dynamic flow and such task-specific feature transformations for different tasks could be modeled as different time points, which are called task positions, in a Neural Ordinary Differential Equation (NODE or Neural ODE) [15]. Different from NODEs, the task positions of different feature transformations in the dynamic flow, corresponding to the given time information in NODEs, are unknown, and the proposed NORMAL method utilizes a time-aware neural ODE block to learn task positions automatically via gradient descent methods. By finding the best positions of different tasks in the continuous flow of the feature space, our model is well-positioned to find the feature representation that the task really needs. Empirically, extensive experiments demonstrate that the proposed NORMAL method outperforms state-of-the-art methods on benchmark datasets. Moreover, the learned task positions can reflect the task relations, which verifies the reasonableness of the learned task positions.

5.2 Introduction

Among all the architectures for deep MTL, the Hard Parameter Sharing (HPS) architecture, which typically shares one feature extractor among tasks and after that has a task-specific head for each task, is the earliest and the most used one. Though it is simple, several works [6, 85, 118, 143] point out that the HPS architecture is effective in improving the performance of each task. However, due to the use of a shared feature extractor to obtain a shared feature representation among tasks, the HPS architecture often faces the problem of competing for shared parameters among tasks during the training process, specifically in the form of gradient conflict which often leads to performance degradation for some tasks [70, 141]. To alleviate this problem, based on the HPS architecture, some recent studies [16, 66, 70, 89, 103, 141] propose loss weighting and gradient manipulation methods to help model training. Meanwhile, some works [56, 131] propose task grouping methods to mitigate the competition between tasks by combining more relevant tasks together to learn an HPS model. These methods require a lot of (pre-)grouping computation and increase the total model size.

To employ task similarity to model the representation of each task, we assume that a dynamic flow exists in the feature space, and each task lies in one specific position in this dynamic flow. Then, by putting each task in its proper location in this dynamic flow according to their task similarity, we can reduce their competition to the shared model parameter. To model this dynamic flow in the feature space, we use the Neural Ordinary Differential Equation (NODE) methods. This approach enables us to model complex feature space and optimize the MTL model via auto differentiation.

5.3 Preliminary

In this section, we introduce the first-order and second-order NODEs.

First-order Neural ODEs. First-order NODEs [15] are proposed recently to model deep neural networks with continuous depths. NODEs model the dynamic of hidden features $z(t) \in \mathbb{R}^n$ via first-order Ordinary Differential Equations (ODEs), which is parameterized by a neural network $g(z(t), t, \varphi) \in \mathbb{R}^n$ with learnable parameters φ, i.e.,

$$\frac{dz(t)}{dt} = g(z(t), t; \varphi). \tag{5.1}$$

For a given initial value $z(t_0)$, NODEs obtain the output at time t with a black-box numerical ODE solver as

$$z(t) = z(t_0) + \int_{t_0}^{t} g(z(s), s; \varphi) ds = \text{ODEsolver}(z(t_0), g, t_0, t; \varphi). \tag{5.2}$$

The main technical difficulty in training such NODEs is how to do the back-propagation through the ODE solver efficiently. In [15], an adjoint sensitivity method is proposed to solve this issue. This method has a low memory cost, and can explicitly control numerical errors.

Second-order Neural ODEs. As the first-order NODEs are easy to suffer from unstable training, slow speed, and lack of highly expressive power, there are many works [19, 26, 129] to improve first-order NODEs. Among them, the Heavy Ball NODE (HBNODE) [129] is more accurate and stable, and it is formulated as

$$\frac{d^2 z(t)}{dt^2} + \gamma \frac{dz(t)}{dt} + g(z(t), t) = 0. \tag{5.3}$$

where $\gamma \geq 0$ is a damping factor and $g(\cdot, \cdot)$ represents a continuous function. In practice, γ can be treated as a hyperparameter or a learnable parameter, and $g(\cdot, \cdot)$ can be parameterized by a neural network. Eq. (5.3) can be reformulated as a first-order NODE system as

$$\frac{dz(t)}{dt} = -q(t), \ \frac{dq(t)}{dt} = -\gamma q(t) + g(z(t), t), \tag{5.4}$$

where $q(t) \in \mathbb{R}^n$ is a momentum function, and the starting point $q(0)$ is computed as $q(0) = -dz(t)/dt \mid_{t=0}$, which represents the initial velocity of $z(t)$. Following the idea of skip connection, one extra term $\xi z(t)$ is added to the second-order ODE system in HBNODE and the final formulation is

$$\begin{cases} \dfrac{dz(t)}{dt} = -q(t), \\ \dfrac{dq(t)}{dt} = -\gamma q(t) + g(z(t), t) + \xi z(t). \end{cases} \tag{5.5}$$

NODEs [15] can learn from irregularly sampled data and are particularly suitable for learning complex dynamical systems. NODE-based methods have shown promising performance on a number of tasks including building normalizing flows [32], modeling continuous time data [137], and generative models [41]. However, training NODEs on large datasets is not an easy task and often leads to poor performance. This is because the training process of NODE is very slow [129], and NODE often fails to learn long-term

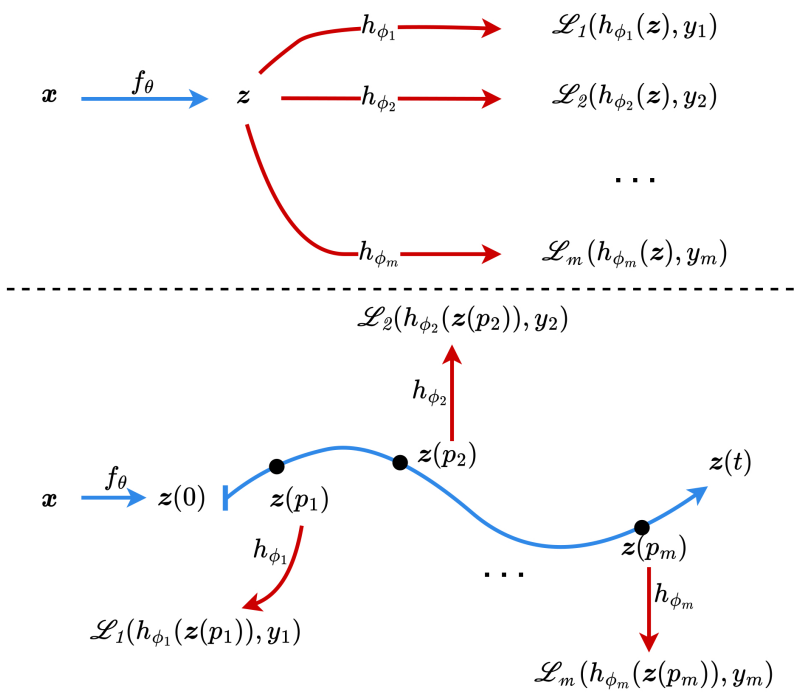


Figure 5.1: Comparison between the HPS-based MTL model (top) and the proposed NORMAL model (bottom). The HPS-based MTL model maps inputs into a shared intermediate representation. The NORMAL method uses task-specific feature transformation which is modeled by task positions in NODE to map inputs into task-specific feature representations. The blue color indicates task-shared components, and the red color denotes task-specific components.

dependencies in sequential data [60]. The HBNODE [129] is based on second-order ODEs with a damping term, which can significantly accelerate the training process and provide a stable result.

5.4 NORMAL Model

In this section, we introduce the proposed NORMAL model.

5.4.1 The Entire Model

To mitigate competition for shared parameters by learning task-specific feature representations while retaining the benefits of MTL to learn feature representations, we treat feature transformations of different tasks as points embedded in a dynamic flow of

transformations and use a NODE to model smoothly varying embeddings. So different task positions on the dynamic flow can be converted into outputs at different times on the NODE.

The NORMAL model consists of a shared feature extractor f_θ which is parameterized by θ, a task-shared dynamic flow modeled by a time-aware neural ODE block Q_φ which is parameterized by φ, and will be introduced in the next section, learnable task positions $\{p_i\}$, and task-specific heads $\{h_{\phi_i}\}$. Here Q_φ is to model feature transformations from different tasks in a dynamic flow. Thus, as shown in Figure 5.1, for a sample in the ith task, the NORMAL model first obtains a hidden representation via f_θ, then moves to task position p_i over Q_φ to learn a feature transformation to obtain a feature representation with task specificity, and finally feed into h_{ϕ_i} to obtain the final output.

Mathematically, the objective function of the NORMAL model is formulated as

$$\min_{\Theta,\{p_i\}} \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}\mathscr{L}_i(y_i^j, h_{\phi_i}(Q_\varphi(f_\theta(x_i^j), p_i))), \tag{5.6}$$

where Θ denotes all the network parameters, including φ, θ, and $\{\phi_i\}_{i=1}^m$, and $Q_\varphi(\cdot, p_i)$ denotes the output of the time-aware neural ODE block at task position p_i. Thus, in terms of notations of NODE as introduced in the previous section, we have $Q_\varphi(f_\theta(x_i), p_i) = z(p_i)$, where $z(0) = f_\theta(x_i)$.

5.4.2 Time-aware Neural ODE Block

In the time-aware neural ODE block, $f_\theta(x_i^j)$ extracted by the shared feature extractor is considered as the state at initial time/position 0 in the dynamic flow, and if p_i is known, then task-specific feature transformations could be learned. However, $\{p_i\}$ are usually unknown, and we aim to learn them.

Though first-order NODEs easily model dynamic flow, due to their instability, using them to implement Q_φ often leads to poor performance. Therefore, some second-order NODE (e.g., HBNODE) is used to build this block. Specifically, based on Eq. (5.5), by using $f_\theta(x_i^j)$ as the initial value $z(0)$ of $z(t)$ for task i, we have

$$Q_\varphi(f_\theta(x_i^j), p_i) = f_\theta(x_i^j) + \int_0^{p_i} -q(t)dt \tag{5.7}$$

$$= f_\theta(x_i^j) - \int_0^{p_i}\left(q(0) + \int_0^t \frac{dq(l)}{dl}dl\right)dt, \tag{5.8}$$

where $\frac{dq(l)}{dl} = -\gamma q(l) + g(z(l), l) + \xi z(l)$ and a mapping function $u(z): \mathbb{R}^n \to \mathbb{R}^n$ maps $f_\theta(x_i^j)$ to the initial velocity $q(0)$. The initial velocity mapping $u(z;\varphi_v)$ and the ODE function

Algorithm 4 The NORMAL model.

Input: training data and learning rates μ, η

Output: Task-shared parameters θ, φ, task-specific parameters $\{\phi_i\}$, $\{p_i\}$

1: **for** $k = 1$ **to** K **do**
2: Compute and save outputs of the time-aware neural ODE block for each task: $z(p_i)$ and $q(p_i)$;
3: Compute total loss $\mathscr{L}$ according to Eq. (5.6);
4: Compute the gradient d_{p_i} with respect to p_i according to Eq. (5.11);
5: Compute the gradient $\nabla_\Theta \mathscr{L}$ with respect to $\{\theta, \{\phi_t\}, \varphi\}$;
6: Update Θ as $\Theta := \Theta - \mu \nabla_\Theta \mathscr{L}$;
7: Update p_i as $p_i := p_i - \eta d_{p_i}$;
8: **end for**

$g(z, t; \varphi_o)$ are parameterized by φ_v and φ_o, respectively. Thus, the time-aware neural ODE block Q_φ is formulated as

$$
\begin{aligned}
Q_\varphi(f_\theta(x_i^j), p_i) = f_\theta(x_i^j) - \int_0^{p_i} \Big(u(f_\theta(x_i^j); \varphi_v) + \\
\int_0^t \big(-\gamma q(l) + g(z(l), l; \varphi_o) + \xi z(l) dl \big) \Big) dt,
\end{aligned} \tag{5.9}
$$

where $\gamma > 0$ is treated a learnable parameter and ξ is treated as a hyperparameter to be tuned. To guarantee the positiveness of γ, we reparameterize it as $\gamma = \text{sigmoid}(\omega)$, where ω is a learnable parameter. For simplicity, we denote these parameters by $\varphi = \{\varphi_v, \varphi_o, \omega\}$.

By using the block presented above, we can now calculate $z(t)$ for any given initial value $z(0) = f_\theta(x_i^j)$ and p_i. Existing studies on NODEs assume that p_i is available before training. In the proposed NORMAL method, if a common p_i is used for all the tasks, the NODE could be absorbed into $f_\theta(\cdot)$ and different tasks could have identical feature representations, which degenerates to the HPS architecture. If different tasks use different fixed p_i, setting them is too inefficient and does not have good performance according to our empirical observations. Therefore, in the NORMAL method, to achieve high expressive power in learned feature representations and low manual costs, we learn $\{p_i\}$ for all the tasks, which will be detailed in the next section.

5.4.3 Optimization

To learn parameters in the time-aware neural ODE block, we apply the adjoint sensitivity method, which can significantly reduce the memory cost during calculating the gradient. However, we cannot use auto differentiation to update $\{p_i\}$ since current mainstream frameworks (e.g., Tensorflow and Pytorch) do not support automatically computing the

gradient for task positions $\{p_i\}$ in NODEs, and we need to manually compute the gradient with respect to $\{p_i\}$. Specifically, for task i, the average loss L_i is defined as

$$L_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathscr{L}_i(y_i^j, h_{\phi_i}(z(p_i))), \tag{5.10}$$

where $z(p_i) = Q_\varphi(f_\theta(x_i^j), p_i)$.

Based on the chain rule, we compute the gradient with respect to p_i as

$$d_{p_i} = \frac{dL_i}{dt}\bigg|_{t=p_i} = \frac{dL_i}{dz(t)} \frac{dz(t)}{dt}\bigg|_{t=p_i} = -\frac{dL_i}{dz(p_i)} q(p_i). \tag{5.11}$$

In Eq. (5.11), $q(p_i)$ denotes the output of the momentum function at time p_i, which can be saved in the forward propagation process. Thus, we need to calculate the gradient of the loss L_i with respect to the output of the time-aware neural ODE block at time p_i. This does not require us to backpropagate the entire model, so it is not computationally expensive. Therefore, we can use d_{p_i} to update p_i as $p_i := p_i - \eta d_{p_i}$, where η represents the step size.

The gradients of other model parameters in the NORMAL method can be computed by auto differentiation and stochastic gradient descent methods can be used to update them. In the NORMAL method, we can update all the learnable parameters jointly or alternatively. Algorithm 4 summarizes the training algorithm for the NORMAL method.

5.5 Empirical Study

In this section, we empirically evaluate the proposed NORMAL method on four benchmark datasets, including Office-31 [101], Office-Home [122], NYUv2 [105], and CelebA [77]. All experiments are performed on a single NVIDIA GeForce RTX 3090 GPU.

Baselines. Here, we compare the proposed method with state-of-the-art MTL methods, including EW that adopts an equal weight on training losses of different tasks, UW [53], GradNorm [16], MGDA [103], PCGrad [141], IMTL [70], CAGrad [66], and Nash-MTL [89].

Evaluation metric. For the Office-31, Office-Home, and CelebA datasets, all of which involve classification tasks, we present the classification accuracy for each individual task as well as the average classification accuracy across all tasks. For the NYUv2 dataset which has three tasks: 13-class semantic segmentation, depth estimation, and surface

normal prediction, by following [84], we use the average of the relative improvement of each task over the EW method, i.e. Δ_p as the evaluation metric, which is defined in Eq. (3.33) in Section 3.7.2

5.5.1 Experimental Results

Datasets. The **Office-31** dataset [101] includes images from three different sources: images downloaded from www.amazon.com (**A**mazon), images from digital SLR cameras (**D**slr), and images from webcams (**W**ebcam). It contains 31 categories for each source and a total of 4652 labeled images. The **Office-Home** dataset [122] includes images from four sources: artistic images (**Ar**), clip art (**Cl**), product images (**Pr**), and real-world images (**Rw**). It contains 65 categories for each source and a total of 15,500 labeled images. Under the multi-task learning setting, we treat each source as a separate task, so these two datasets can be used as a multi-task classification problem.

Implementation Details. On both datasets, we use a ResNet-18 network pre-trained on the ImageNet dataset as f_θ, the Euler's method as the ODE solver, and a task-specific fully connected layer as the corresponding head for each task. The architectures of the initial velocity $u(z;\varphi_v)$ and ODE function $g(z,t;\varphi_o)$ are constructed as follows.

- Initial Velocity: → FC → LeakyReLU → FC →
- ODE Function: → FC → LeakyReLU → FC →

All the fully connected (FC) layers of both functions have a dimension of 512 for inputs and outputs.

Results. The results on the Office-31 and Office-Home datasets are shown in Table 5.1. We can see that on both datasets, the NORMAL method outperforms state-of-the-art baseline methods in terms of average classification accuracy. Compared with the unbalanced performance of several baseline methods on different tasks, the NORMAL method achieves a boost on almost all tasks over the EW method. For example, on the Office-31 dataset, the MGDA method performs well on the **D** and **W** tasks but performs very poorly on the **A**, which does not occur in the NORMAL method. This result demonstrates the advantage of the NORMAL method in that it can learn better feature representations for each task by learning task-specific feature transformations over dynamic flow. Moreover, compared with baselines, the proposed NORMAL method

Table 5.1: Classification accuracy (%) of different methods on the **Office-31** and **Office-Home** datasets. 3 independent runs are conducted for each experiment. The mean performance is reported. The best results for each task are shown in **bold**.

Method	Office-31				Office-Home				
	A	**D**	**W**	**Avg**	**Ar**	**Cl**	**Pr**	**Rw**	**Avg**
EW	84.67	98.09	98.70	93.82	64.77	79.05	90.11	80.44	78.59
UW	84.62	97.81	98.89	93.77	66.03	79.09	89.69	79.78	78.65
GradNorm	84.22	98.09	98.89	93.73	64.84	78.73	89.86	**80.58**	78.50
MGDA	78.69	98.09	98.70	91.83	65.40	75.05	89.76	79.96	77.54
PCGrad	84.67	97.81	98.70	93.73	65.27	78.37	90.08	79.89	78.40
IMTL	83.02	98.09	98.89	93.33	65.27	77.72	89.90	80.54	78.36
CAGrad	84.33	97.81	**99.07**	93.74	64.90	78.48	**90.47**	80.18	78.50
Nash-MTL	83.82	98.91	**99.07**	93.93	66.79	78.66	90.29	79.82	78.89
NORMAL	**86.32**	**99.18**	98.88	**94.80**	**69.26**	**80.39**	**90.47**	80.22	**80.08**

Table 5.2: Performance on three tasks (i.e. 13-class semantic segmentation, depth estimation, and surface normal prediction) in the **NYUv2** dataset. 3 independent runs are conducted for each experiment. The mean performance is reported. The best results for each task are shown in **bold**.↑(↓) means that the higher (lower) the value, the better the performance.

Method	Segmentation		Depth		Surface Normal					$\Delta\uparrow$
	mIoU↑	**Pix Acc ↑**	**Abs Err ↓**	**Rel Err↓**	**Angle Distance**		**Within t°**			
					Mean ↓	**Median ↓**	**11.25 ↑**	**22.5 ↑**	**30 ↑**	
EW	**0.4875**	0.7183	0.4179	0.1734	25.42	18.84	0.3243	0.5729	0.6845	0%
UW	0.4866	0.7165	**0.4085**	0.1711	25.42	18.77	0.3212	0.5703	0.6829	0.45%
GradNorm	0.4789	0.7106	0.4134	0.1686	25.40	18.61	0.3237	0.5733	0.6848	0.26%
MGDA	0.4138	0.6631	0.4416	0.1825	**24.33**	**17.48**	**0.3423**	**0.5975**	**0.7065**	-3.97%
PCGrad	0.4835	0.7155	0.4124	0.1718	25.40	18.66	0.3230	0.5726	0.6844	0.21%
IMTL	0.4769	0.7112	0.4141	0.1711	24.76	17.90	0.3355	0.5881	0.6978	0.89%
CAGrad	0.4777	0.7113	0.4128	**0.1676**	24.80	17.92	0.3356	0.5874	0.6973	1.29%
Nash-MTL	0.4764	0.7103	0.4155	0.1704	24.64	17.71	0.3409	0.5911	0.6999	1.12%
NORMAL	0.4857	**0.7184**	0.4113	0.1691	24.98	18.34	0.3352	0.5827	0.6923	**1.33%**

achieves the best result in some tasks, such as the best classification accuracy of 86.32% in task **A** for the Office-31 dataset and 69.26% classification accuracy in task **Ar** for the Office-Home dataset.

Dataset. The NYUv2 dataset [105] comprises video sequences of diverse indoor environments captured using RGB and depth cameras from the Microsoft Kinect. It includes a total of 1,449 labeled images, with 795 images designated for training and 654 for validation. The dataset has three distinct tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction.

Implementation Details. On the NYUv2 dataset, we use the DeepLabV3+ architecture [14] with HBNODE. Specifically, we use pre-trained Resnet-18 with dilated convolutions [138] as the feature extractor shared by all tasks, the Euler's method as the ODE solver, and the Atrous Spatial Pyramid Pooling (**ASPP**) [14] as the task-specific header for each task. The architectures of the initial velocity $u(z;\varphi_v)$ and ODE function $g(z,t;\varphi_o)$ are constructed as follows.

- Initial Velocity: → Conv → LeakyReLU → Conv →
- ODE Function: → Conv → LeakyReLU → Conv →

The numbers of the input channel and output channel of the convolution layers of both functions are set to 512, the size of the kernel is 1, the stride size is 1, and the padding is 0.

Results. The results on the NYUv2 dataset are shown in Table 5.2. Overall, the proposed NORMAL method achieves good performance when compared with the state-of-the-art baseline methods. The MGDA method obtains the best results on all metrics of the surface normal prediction task, but performs poorly in the other two tasks, thus its overall performance is not so good. In contrast, The NORMAL method achieves relatively balanced performance on all the tasks, and hence its overall performance in terms of Δ exceeds all other methods. This illustrates the ability of the NORMAL method to effectively improve performance on all the tasks by finding task positions.

Dataset. The **CelebA** dataset [77] includes a total of 202,599 face images and 40 face attribute annotations. In the multi-task learning setup, each face attribute is treated as a task. Thus, there are 40 classification tasks in this dataset.

Implementation Details. On the CelebA dataset, we use Resnet-18 with an average-pooling as the task-shared feature extractor f_θ, the Euler's method as our ODE solver, and a fully connected layer as the task-specific head for each task. The architectures of the initial velocity $u(z;\varphi_v)$ and ODE function $g(z,t;\varphi_o)$ are constructed as follows.

- Initial Velocity: → FC → LeakyReLU → FC →
- ODE Function: → FC → LeakyReLU → FC →

All fully connected layers of both functions have a dimension of 2048 for both inputs and outputs.

Table 5.3: Average classification accuracy (%) of different methods on the **CelebA** dataset with 40 tasks. 3 independent runs are conducted for each experiment. The mean performance is reported. The best results are shown in **bold**.

Method	**Avg**
EW	90.78
UW	90.82
GradNorm	90.69
MGDA	90.40
PCGrad	90.93
IMTL	90.46
CAGrad	90.73
Nash-MTL	90.83
NORMAL	**91.00**

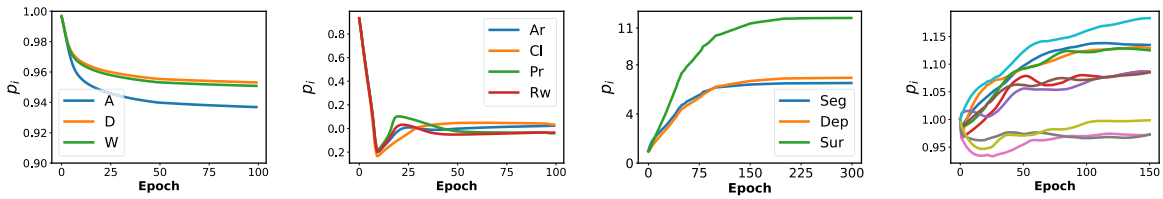
Results. As demonstrated in Table 5.3, the NORMAL method achieves the highest average classification accuracy on the CelebA dataset, outperforming all other baseline methods. Notably, none of the competing approaches surpass the 91% average accuracy threshold, further underscoring the superiority of the NORMAL method.

5.5.2 Analysis on Learned Task Positions

In this section, we analyze the learned task positions $\{p_i\}$ to see why the NORMAL method achieves good performance on these datasets.

The training curves of all $\{p_i\}$ on the four benchmark datasets are shown in Figure 5.2, where due to the large number of tasks in the CelebA dataset, we randomly select a portion of the tasks for better illustration. According to Figure 5.2, we have two observations.

Firstly, the proposed NORMAL method does successfully learn task positions. Taking the training curves of $\{p_i\}$ on the NYUv2 dataset in Figure 5.2(c) as an example, we can see that its training trajectory is very smooth and all task positions eventually converge to their own convergent points. We can also find similar results in Figures 5.2(a) and 5.2(b). In Figure 5.2(d), though the training trajectory is not as smooth as the other three datasets, the proposed NORMAL method can still differentiate tasks and find their own task positions.



(a) Office-31 (b) Office-Home (c) NYUv2 (d) CelebA

Figure 5.2: Task positions $\{p_i\}$ throughout the training process on the four datasets.

Secondly, $\{p_i\}$ learned by the NORMAL method are able to reflect task relations. For example, according to Figures 5.2(a) and 5.2(b), the task positions in these two datasets converge to a very similar value. This result is consistent with the nature of the Office-31 and the Office-Home datasets in that different tasks in each dataset are semantically similar due to the shared label space among tasks. In Figure 5.2(c), we find that p_{sur} learned for the surface normal prediction task is different from p_{seg} and p_{dep} learned for the semantic segmentation and depth estimation tasks, which indicates that the surface normal prediction task is not so related to the other two tasks, and this observation matches some previous study [113], which verifies that the learned $\{p_i\}$ can reflect task relations. In Figure 5.2(d), we can see that different tasks tend to form several groups based on learned task positions $\{p_i\}$. For example, some tasks (e.g., Sideburns and Wavy_Hair) have similar task positions as face attributes corresponding to those tasks are similar. Moreover, some tasks (e.g., Receding_ Hairline and 5_o_Clock_Shadow) have different task positions since face attributes corresponding to those tasks are totally different. Those results show that the learned task positions could help identify task clusters.

In summary, the proposed NORMAL method could learn meaningful task positions that can reveal task relations.

5.5.3 Ablation Studies

In this section, we conduct ablation studies on the Office-31 and Office-Home datasets to answer several questions which are placed at the beginning of the following paragraphs.

Are the learned task positions advantageous compared with fixed task positions? We try to use a fixed parameter p shared by all tasks and use task-specific fixed parameters $\{p_i\}$ for different tasks. For the former setting, we use $p = 1$. In the latter setting, for the Office-31 dataset, we try to set task positions of the three tasks

Table 5.4: Ablation studies on the **Office-31** and **Office-Home** datasets in terms of the classification accuracy (%). 3 independent runs are conducted for each experiment. The mean performance is reported.

Setting	Office-31				Office-Home				
	A	**D**	**W**	**Avg**	**Ar**	**Cl**	**Pr**	**Rw**	**Avg**
Constant and identical $\{p_i\}$	84.62	98.36	98.89	93.95	61.35	75.37	87.75	75.28	74.94
Constant but different $\{p_i\}$	85.24	98.36	98.52	94.04	61.35	75.37	87.75	75.28	74.94
Different form of p_i: $p_i = e^{v_i}$	85.47	97.81	98.70	94.00	68.82	79.23	89.55	80.68	79.57
Using first-order NODE	84.79	98.36	98.89	94.01	65.27	77.57	88.98	76.97	77.45
Adding task-shared layers	83.87	97.81	98.15	93.28	58.13	73.28	85.06	72.43	72.22
Adding task-specific layers	83.87	97.81	97.59	93.09	60.34	74.50	87.25	74.31	74.10
NORMAL	86.32	99.18	98.88	94.80	69.26	80.39	90.47	80.22	80.08

to each permutation of a set $\{1, 1.25, 1.5\}$ and select the best performed one, and for the Office-Home dataset, the set to generate task positions is $\{1, 1.25, 1.5, 1.75\}$. According to the results shown in Table 5.4, the performance of the two settings for task positions is inferior to the learning of task positions in the proposed NORMAL method, which demonstrates the effectiveness of the learning strategy for task positions in the NORMAL method.

How do different forms of learning task positions impact the performance? Here we try positive task positions and parameterize task positions $\{p_i\}$ as $p_i = e^{v_i}$. As shown in Table 5.4, the model learned here is inferior to that of the NORMAL model without any constraint on task positions by 0.80% and 0.51% on the Office-31 and Office-Home datasets, respectively, which shows that learning task positions without the positive requirement may be better. Based on Tables 5.1 and 5.4, this variant to learn positive task positions still performs better than baseline methods, which again verifies the effectiveness of the NORMAL method.

How do different NODE algorithms impact the performance? The NORMAL method uses a second-order ODE method but not first-order NODEs. Here we explore whether different methods in the NODE family impact the performance of the NORMAL method. We evaluate the performance of the NORMAL method using the first-order NODE [15]. According to results shown in Tables 5.1 and 5.4, we can see that on the Office-Home dataset, the performance of the NORMAL method using the first-order NODE method is worse than the NORMAL method and baseline methods. Some possible reasons are that first-order NODEs usually cannot be trained stably and that second-order NODEs are more expressive. For the Office-31 dataset, the NORMAL method with

the first-order NODE performs slightly worse than the NORMAL method but slightly better than the baseline methods. One possible reason is that the Office-31 dataset is easier than the Office-Home dataset. In summary, second-order NODE methods are preferred to be used in the NORMAL method.

Does the enhancement of the NORMAL method result from the addition of certain parameters? The time-aware neural ODE block in the NORMAL method introduces a small number of parameters, which are almost negligible compared to other model parameters. We aim to investigate whether such an increasing number of parameters brings performance gain. Therefore, we experiment to add task-shared layers in f_θ and task-specific layers in $\{h_{\phi_i}\}$, respectively, to match with the number of parameters in NORMAL method, and show results in Table 5.4. According to the results, we can see that the introduction of additional layers does not bring a performance improvement, and even lead to performance degradation. One possible reason could be the overfitting issue. Through this experiment, we can see that the performance of the NORMAL method is attributed from the entire model design but not the introduction of more parameters.

5.6 Summary

In this chapter, we present a model parameterization method called NORMAL to model multiple learning tasks from the perspective of dynamic flow. By learning the task positions in the NODE, the NORMAL method can model the task relations in terms of the relative task positions. Experiments on benchmark datasets demonstrate the effectiveness of the proposed NORMAL method.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This thesis focuses on designing effective weighting algorithms and parameterized models for MTL. Specifically, this thesis presents several works to address three important research questions for multi-task learning: 1) How to design an effective weighting method for multi-task learning; 2) How to design an effective balancing method for multi-task learning in black-box scenarios; 3) How to use task similarity to design a parameterized model to improve performance in multi-task learning.

To answer research question 1, we propose a bi-level optimization based framework to find the task weighting for training the MTL model. Two reliable algorithms, MOML and FORUM with comprehensive theoretical results were drawn to solve the proposed framework. Extensive experiments demonstrate the effectiveness and efficiency of the proposed framework and methods

To answer research question 2, we consider black-box MTL as a black-box MOO problem and propose a novel ASMG algorithm to solve it. We prove the convergence rate for the proposed ASMG algorithm in both convex and non-convex cases. Moreover, the proposed ASMG algorithm achieves state-of-the-art performance on black-box multi-task learning problems.

To answer research question 3, we use a dynamic flow to model the feature space in MTL and propose a Neural Neural Ordinary Differential Equation (NODE) based

parameterized model called NORMAL for MTL training. The task positions learned by the NORMAL method can be used to evaluate the relevance of different tasks, which could improve the interpretability of the proposed NORMAL method.

In conclusion, we present two reliable weighting algorithms and one novel model parameterization method, contributing to the development of more effective multi-task learning systems.

6.2 Future Work

This thesis outlines several promising avenues for future research:

- *Multi-modal multi-task learning.* In real-world scenarios, different types of data often provide complementary insights for solving the same set of tasks. Multi-modal multi-task learning aims to leverage this by training a single model to handle multiple tasks across diverse input data modalities, such as text, images, audio, and video. By sharing parameters across these tasks and modalities, multi-modal MTL can more effectively capture and integrate cross-modal relationships. This not only enhances the model's ability to generalize but also allows for more robust performance in complex tasks that rely on multiple data sources. For instance, in applications like autonomous driving or healthcare, combining visual data with sensor or textual data can significantly improve decision-making and prediction accuracy. By jointly learning from multiple modalities, multi-task learning methods can reduce redundancy, promote efficient learning, and better utilize complementary information, leading to more powerful and flexible multi-modal models.

- *Black-box multi-task learning.* Black-box Multi-Task Learning (MTL) plays a crucial role in the development of Large Language Models (LLMs) by enabling these models to effectively handle and optimize multiple tasks simultaneously without requiring explicit understanding of their internal mechanisms. Since many LLMs are only allowed access with APIs, when we try to develop an adapter model for these black-box models, the gradient of these model parameters is no longer available. Therefore, we need to design a more effective black-box MTL model and algorithm to solve the problems raised by current LLM studies.

APPENDIX

A.1 Additional Material for Chapter 3

A.1.1 Notations and Terminologies

We first recall some definitions in multi-objective optimization, including the definition of the minimality and convexity of vector-valued functions and Kuratowski-Painlevé set-convergence [79].

Let P be the set of non-negative real vectors $\mathbb{R}^m_+ = \{l \in \mathbb{R}^m : \forall l_i \geq 0\}$, where l_i denote the i-th entry in l. The interior int$P$ denotes the set of positive real vectors int$P = \{l \in \mathbb{R}^m : \forall l_i > 0\}$. P is a closed and convex cone, and the interior int$P$ induce a partial order for any two points in $\mathbb{R}^m$. That is, for any $l^1, l^2 \in \mathbb{R}^m$, we define

$$l^1 \leq l^2 \iff l^2 - l^1 \in P$$
$$l^1 < l^2 \iff l^2 - l^1 \in \text{int}P.$$

That is, for $l^1, l^2 \in \mathbb{R}^m$, the partial ordering $l^1 \leq l^2$ and $l^1 < l^2$ imply that $l^1_i \leq l^2_i$ and $l^1_i < l^2_i$ for all $i \in \{1, ..., m\}$, respectively. Given $l^1 \in \mathbb{R}^m$, we define $l^1 - P = \{l \in \mathbb{R}^m : l \leq l^1\}$ and $l^1 - \text{int}P = \{l \in \mathbb{R}^m : l < l^1\}$.

We now recall the notions of minimality for a subset in $\mathbb{R}^m$.

Definition A.1. For a nonempty set $C \in \mathbb{R}^m$, the set of all minimal points in C w.r.t the ordering cone P is defined as

$$\text{Min } C := \{l \in C : C \cap (l - P) = \{l\}\}.$$

The weakly minimal points of the set C is

$$\text{WMin } C := \{l \in C : C \cap (l - \text{int}P) = \emptyset\}.$$

In the MOP, for the given objective function $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ $(m, n \in \mathbb{N}, m \geq 2)$, where $z \in \mathcal{Z}$, we denote by Min $g(z)$ the set of all the minimal points of the function g. We also call it as the Pareto frontier or Pareto-optimal set. Thus, the corresponding efficient solution or Pareto-optimal solution of $g(z)$ can be defined as

$$\text{Eff}\,(g(z)) := \{z \in \mathcal{Z} : g(z) \in \underset{z \in \mathcal{Z}}{\text{Min}}\, g(z)\}.$$

Similarly, we denote by WMin $g(z)$ the set of weakly minimal points of the function $g(z)$ and by WEff $(g(z))$ the corresponding weakly efficient solution set.

Definition A.2. The function $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ is a P-convex function if for every $z_1, z_2 \in \mathbb{R}^n$ and for every $\lambda \in [0, 1]$, we have

$$g(\lambda z_1 + (1 - \lambda)z_2) \leq \lambda g(z_1) + (1 - \lambda)g(z_2).$$

$g(z)$ is a strictly P-convex function, if for every $z_1, z_2 \in \mathbb{R}^n$, $z_1 \neq z_2$ and for every $\lambda \in (0, 1)$,

$$g_i(\lambda z_1 + (1 - \lambda)z_2) < \lambda g_i(z_1) + (1 - \lambda)g_i(z_2).$$

Remark A.1. *For a given vector-valued function $g(z)$, we have* Min $g(z) \subseteq$ WMin $g(z)$. *If g is strictly P-convex, we have* Min $g(z) =$ WMin $g(z)$ *and* WEff $(g(z)) =$ Eff $(g(z))$.

Definition A.3. Consider $\{A_n\}$ as a sequence of subsets of a set X in an Euclidean space. Li A_n is defined as the lower limit of the sequence of sets $\{A_n\}$, that is,

$$\text{Li } A_n := \{a \in X : a = \lim_{n \to +\infty} a_n, a_n \in A_n, \text{ for sufficiently large } n\}.$$

Ls A_n is defined as the upper limit of the sequence of sets $\{A_n\}$, that is,

$$\text{Ls } A_n := \{a \in X : a = \lim_{n \to +\infty} a_n, a_n \in A_{n_k}, \text{ for } n_k \text{ as a selection of integers.}\}.$$

A sequence $\{A_n\}$ converges in the Kuratowski sense to one set $A \subseteq X$, when

$$\text{Ls } A_n \subseteq A \subseteq \text{Li } A_n,$$

and we denote such convergence by $A_n \to A$.

A.1.2 Proofs of Theorems in Section 3.4.3

For the sake of clarity, we first introduce some notation from [79]. The sublevel set of a function $g(z):\mathbb{R}^n \to \mathbb{R}^m$ at height $h \in \mathbb{R}^m$ is defined as $g^h := \{z \in \mathbb{R}^n : g(z) \leq h\}$. If A is a closed convex set, we can define then the recession cone of A as $0^+(A) := \{d \in \mathbb{R}^n : a + td \in A, \forall a \in A, \forall t \geq 0\}$. The recession cone of the sublevel set of the function $g(z)$ is denoted by H_g.

To prove theorems in Section 3.4.3, we will use Theorems 3.1 and 3.2 in [80], and list them in the following for completeness.

Theorem A.1. *Suppose that $\mathcal{Z}$ is a nonempty closed, convex set in $\mathbb{R}^n$ and $g(z):\mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function with $z \in \mathcal{Z}$. Then, if $g_n(z) \to g(z)$ w.r.t. the continuous convergence, we have* $\mathrm{LsWMin}\ g_n(z) \subseteq \mathrm{WMin}\ g(z)$.

Theorem A.2. *Suppose that $\mathcal{Z}$ is a nonempty closed, convex set in $\mathbb{R}^n$ and $g(z):\mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function with $z \in \mathcal{Z}$. Then, if $g_n(z) \to g(z)$ w.r.t. the continuous convergence, $g_n(z)$ and $g(z)$ are both P-convex functions and $0^+(\mathcal{Z}) \cap H_g = \{0\}$, we have* $\mathrm{Min}\ g(z) \subseteq \mathrm{LiMin}\ g_n(z)$.

A.1.2.1 Proof of Theorem 3.1

Proof. To show that $F(\omega^*(\alpha), \alpha)$ is continuous on α, we need to prove that for any convergent sequence $\alpha_n \to \bar{\alpha}$, $F(\omega^*(\alpha_n), \alpha_n)$ converges to $F(\omega^*(\bar{\alpha}), \bar{\alpha})$.

Suppose that $\{\alpha_n\}$ is a sequence in $\mathcal{A}$ satisfying $\alpha_n \to \bar{\alpha}$. As $\arg\min_\omega f(\omega, \alpha)$ is a singleton, we have $\omega^*(\alpha_n) = \arg\min_\omega f(\omega, \alpha_n)$. Since $\{\omega^*(\alpha)\}$ is bounded for $\alpha \in \mathcal{A}$, according to Bolzano-Weierstrass theorem [5], there exists a convergent subsequence $\{\omega^*(\alpha_{kn})\}$ such that $\omega^*(\alpha_{kn}) \to \bar{\omega}$ for some $\bar{\omega} \in \mathbb{R}^p$. Since $\alpha_{kn} \to \bar{\alpha}$ and $f(\omega, \alpha)$ is jointly continuous, we can get that $\forall \omega \in \mathbb{R}^p$, $f(\bar{\omega}, \bar{\alpha}) = \lim_n f(\omega^*(\alpha_{kn}), \alpha_{kn}) \leq \lim_n f(\omega(\alpha_{kn}), \alpha_{kn}) = f(\omega(\bar{\alpha}), \bar{\alpha})$. Therefore, we obtain $\omega^*(\bar{\alpha}) = \bar{\omega}$. This means $\{\omega^*(\alpha_{kn})\}$ has only one cluster point $\omega^*(\bar{\alpha})$. Thus, $\omega^*(\alpha_n)$ converges to $\omega^*(\bar{\alpha})$ as $\alpha_n \to \bar{\alpha}$. Because F is jointly continuous, we have $F(\omega^*(\alpha_n), \alpha_n) \to F(\omega^*(\bar{\alpha}), \bar{\alpha})$ as $\alpha_n \to \bar{\alpha}$. ■

A.1.2.2 Proof of Theorem 3.2

Proof. To prove the first claim of Theorem 3.2, we firstly show that $\varphi_T(\alpha)$ continuously converges to $\varphi(\alpha)$. Suppose there exists a sequence $\{\alpha_n\}$ in $\mathcal{A}$ satisfying $\alpha_n \to \alpha$. Then for

any $\varphi_T(\alpha)$ and sequence α_n, we have

$$\|\varphi_T(\alpha_n)-\varphi(\alpha)\| = \|F(\omega_T(\alpha_n),\alpha_n)-F(\omega^*(\alpha),\alpha)\| \tag{A.1}$$

$$\leq \|F(\omega_T(\alpha_n),\alpha_n)-F(\omega^*(\alpha_n),\alpha_n)\| + \|F(\omega^*(\alpha_n),\alpha_n)-F(\omega^*(\alpha),\alpha)\|. \tag{A.2}$$

According to the continuity property in Theorem 3.1, we have $F(\omega^*(\alpha_n),\alpha_n) \to F(\omega^*(\alpha),\alpha)$ as $\alpha_n \to \alpha$. Furthermore, because $F(\cdot,\alpha)$ is uniformly Lipschitz continuous, we have

$$\|F(\omega_T(\alpha_n),\alpha_n)-F(\omega^*(\alpha_n),\alpha_n)\| \leq L\|\omega_T(\alpha_n)-\omega^*(\alpha_n)\|. \tag{A.3}$$

According to assumption (i) in Theorem 3.2, $\omega_T(\alpha)$ converges to $\omega^*(\alpha)$ uniformly as $K \to +\infty$. Therefore, $\varphi_T(\alpha)$ continuously converges to $\varphi(\alpha)$.

Since Min $\varphi(\alpha) \subseteq$ WMin $\varphi(\alpha)$ and Theorem A.1, we have

$$\text{LsMin } \varphi_T(\alpha) \subseteq \text{LsWMin } \varphi_T(\alpha) \subseteq \text{WMin } \varphi(\alpha). \tag{A.4}$$

Because $\mathscr{A}$ is a compact convex set in $\mathbb{R}^n$, $0^+(\mathscr{A}) = \{0\}$. Then, the condition $0^+(\mathscr{A}) \cap H_\varphi = \{0\}$ is naturally satisfied for function $\varphi(\alpha)$. According to assumption (iii) in Theorem 3.2, $\varphi(\alpha)$ and $\varphi_T(\alpha)$ are both P-convex functions. Then we obtain the lower part of the set convergence from Theorem A.2 as

$$\text{Min } \varphi(\alpha) \subseteq \text{LiMin } \varphi_T(\alpha) \subseteq \text{LiWMin } \varphi_T(\alpha). \tag{A.5}$$

Because $\varphi(\alpha)$ is strictly P-convex, we have WMin φ = Min φ and then we get Min $\varphi_T(\alpha) \to$ Min $\varphi(\alpha)$ according to Definition A.3.

For the second claim, let $\alpha_n \in$ Eff $\varphi_T(\alpha)$ and $\alpha_n \to \bar{\alpha}$. Since Min $\varphi_T(\alpha) \to$ Min $\varphi(\alpha)$, we get $\varphi_T(\alpha_n) \to \varphi(\bar{\alpha})$ and $\bar{\alpha} \in$ Min $\varphi(\alpha)$, which implies LsEff $\varphi_T(\alpha) \subseteq$ Eff $\varphi(\alpha)$.

For the lower limit, by defining $\bar{\alpha} \in$ Eff $\varphi(\alpha)$, the corresponding minimal point satisfies $\bar{l} = \varphi(\bar{\alpha}) \in$ Min $\varphi(\alpha)$. Based on the first claim, there is a sequence $\{l_T\}$ in Min $\varphi_T(\alpha)$ such that $l_T \to \bar{l}$. Then we can take a bounded sequence $\{\alpha_T\}$, where $\alpha_T = \varphi_T^{-1}(l_T)$ and the subsequence of $\{\alpha_T\}$ has a cluster point. Because $\varphi(\alpha)$ is strictly P-convex, this cluster point is $\bar{\alpha}$. Then, we have $\alpha_T \to \bar{\alpha}$, which implies Eff $\varphi(\alpha) \subseteq$ LiEff $\varphi_T(\alpha)$. Combined with the upper limit convergence, we can get Eff $\varphi_T(\alpha) \to$ Eff $\varphi(\alpha)$. ■

Remark A.2. *In fact, if we consider the weakly minimal points under the same assumptions in Theorems 3.1 and 3.2, we can still obtain similar convergence results to those in Theorem 3.2, i.e.*

$$\text{WMin } \varphi_T(\alpha) \to \text{WMin } \varphi(\alpha),\ \text{WEff } \varphi_T(\alpha) \to \text{WEff } \varphi(\alpha).$$

Since $\varphi(\alpha)$ is strictly P-convex, the first claim can be directly obtained according to Eqs. (A.4) *and* (A.5)*. Then, the proof of the convergence of the weakly efficient solution follows that of Theorem 3.2.*

A.1.2.3 Proof of Theorem 3.3

Proof. Theorem 3.3 can be directly obtained from Lemma 6 of [50]. ■

A.1.2.4 Proof of Theorem 3.4

Proof. For the sake of notation simplicity, we denote the i-th entry of the true gradient by $g_i(\alpha_k) = \nabla\varphi_i(\alpha)$ and the approximated gradient by $\widetilde{g}_i(\alpha_k) = \frac{\partial F_i(\omega_T(\alpha_k),\alpha_k)}{\partial \alpha_k}$. We denote by γ_k the corresponding convex combination vector calculated based on $g(\alpha_k)$. The corresponding weights calculated by $\widetilde{g}(\alpha_k)$ is denoted by $\widetilde{\gamma}_k$. According to Theorem 3.3, we have $\|\widetilde{g}_i(\alpha_k) - g_i(\alpha_k)\| \leq \Gamma(T)$. Then we get

$$\|\Lambda(\widetilde{g}(\alpha_k),\widetilde{\gamma}_k)\|^2 \leq \|\widetilde{g}_{\max}(\alpha_k)\|^2 \leq 2L^2 + 2\Gamma(T)^2, \tag{A.6}$$

where the second inequality is due to the triangle inequality and $L = \max_{1\leq i\leq m} L_i$. By setting $A_k = \alpha_k - \alpha_*$, we have

$$\|A_{k+1}\|^2 = \|A_k - \nu_k \Lambda(\widetilde{g}(\alpha_k),\widetilde{\gamma}_k)\|^2 = \|A_k\|^2 - 2\nu_k \langle A_k, \Lambda(\widetilde{g}(\alpha_k),\widetilde{\gamma}_k)\rangle + \nu_k^2 \|\Lambda(\widetilde{g}(\alpha_k),\widetilde{\gamma}_k)\|^2. \tag{A.7}$$

Since the i-th entry of φ_T is c_i-strongly-convex, then for any $\widetilde{\gamma}$, function $\Lambda(\varphi_T(\alpha),\widetilde{\gamma})$ is c-strongly-convex, where $c = \min_{1\leq i\leq m} c_i$. Then, for a given vector $\widetilde{\gamma}$, we have

$$\Lambda(\varphi_T(\alpha^*),\widetilde{\gamma}_k) \geq \Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k) + \Lambda(\nabla\varphi_T(\alpha_k),\widetilde{\gamma}_k)^\top(\alpha^* - \alpha_k) + \frac{c}{2}\|\alpha^* - \alpha_k\|^2. \tag{A.8}$$

We define $S_k = \Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k) - \Lambda(\varphi_T(\alpha^*),\widetilde{\gamma}_k)$. Then, plugging inequalities (A.6) and (A.7) into (A.8), we can have

$$2\nu_k S_k \leq (1-\nu_k c)\|A_k\|^2 + \nu_k^2(2L^2 + 2\Gamma(T)^2) - \|A_{k+1}\|^2. \tag{A.9}$$

By setting $\nu_k = \frac{2}{c(k+1)}$, we obtain

$$S_k \leq \frac{c(k-1)}{4}\|A_k\|^2 - \frac{c(k+1)}{4}\|A_{k+1}\|^2 + \frac{2(L^2+\Gamma(T)^2)}{c(k+1)}. \tag{A.10}$$

Multiplying by k on both sides of (A.10), and summing over $k = 1,...,K$ yields

$$\sum_{k=1}^{K} kS_k \leq \frac{-ck(k+1)}{4}\|A_{k+1}\|^2 + \sum_{k=1}^{K}\frac{2k(L^2+\Gamma(T)^2)}{c(k+1)} \leq \frac{2K(L^2+\Gamma(T)^2)}{c}. \tag{A.11}$$

Dividing both sides by $\sum_{k=1}^{K} k$ gives us

$$\frac{\sum_{k=1}^{T} k\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k) - \sum_{k=1}^{K} k\Lambda(\varphi_T(\alpha_*),\widetilde{\gamma}_k)}{\sum_{k=1}^{K} k} \leq \frac{4(L^2+\Gamma(T)^2)}{c(K+1)}. \tag{A.12}$$

By setting $\bar{\gamma}_k = \frac{\sum_{k=1}^{K} k\widetilde{\gamma}_k}{\sum_{k=1}^{K} k}$, we get

(A.13)
$$\min_{k=1,\ldots,K} \Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k) - \Lambda(\varphi_T(\alpha_*),\bar{\gamma}_k) \leq \frac{\sum_{k=1}^{K} k\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}_k) - \sum_{k=1}^{K} k\Lambda(\varphi_T(\alpha_*),\widetilde{\gamma}_k)}{\sum_{k=1}^{K} k}.$$

Thus, the proof is completed by combining (A.12) and (A.13). ■

A.1.2.5 Proof of Theorem 3.5

Proof. Since the $\bar{\alpha}^*$ and α^* are the unique solution of the objectives $\Lambda(\alpha(\alpha),\widetilde{\gamma}^*)$ and $\Lambda(\alpha_T(\alpha),\widetilde{\gamma}^*)$, respectively. Then, according to the optimality condition, we have

(A.14)
$$\|\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi_T(\alpha^*),\widetilde{\gamma}^*)\| = 0.$$

For a given $\widetilde{\gamma}^*$, by using Theorem 3.3, we have $\|\Lambda(\nabla\varphi_T(\alpha^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*)\| \leq \Gamma(T)$. Therefore, using triangle inequality, we have

(A.15)
$$\|\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*)\| \leq \Gamma(T).$$

Since the i-th entry of φ is $\hat{c}_i$-strongly-convex, then for any $\widetilde{\gamma}^*$, function $\Lambda(\varphi(\alpha),\widetilde{\gamma}^*)$ is $\hat{c}$-strongly-convex, where $\hat{c} = \min_{1\leq i\leq m} \hat{c}_i$. Then, for a given vector $\widetilde{\gamma}^*$, we have

(A.16)
$$(\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*))^\top(\bar{\alpha}^* - \alpha^*) \geq \hat{c}\|\bar{\alpha}^* - \alpha^*\|.$$

Note that $\mathscr{A}$ is a bounded set, there exists a positive diameter δ such that $\|\alpha - \alpha^*\| \leq \delta \leq \infty$ holds for any two points α and α^* in $\mathscr{A}$. Then we have

(A.17)
$$\|\bar{\alpha}^* - \alpha^*\| \leq \frac{\delta}{\hat{c}}\|\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*)\| \leq \frac{\delta}{\hat{c}}\Gamma(T),$$

where the first inequality is due to the Cauchy-Schwarz inequality. Based on the strong-convexity of the lower-level function $f(\cdot,\alpha)$, for given α, we have $\|\omega^*(\alpha) - \omega_T(\alpha)\| \leq (1-\mu\vartheta)^T\|\omega^0 - \omega^*(\alpha)\|$, where ω^0 is the initialization of ω in the inner loop. Then we have

(A.18)
$$\|\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\varphi_T(\alpha^*),\widetilde{\gamma}^*)\|$$

(A.19)
$$\leq \|\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\varphi(\alpha^*),\widetilde{\gamma}^*)\| + \|\Lambda(\varphi(\alpha^*),\widetilde{\gamma}^*) - \Lambda(\varphi_T(\alpha^*),\widetilde{\gamma}^*)\|$$

(A.20)
$$\leq L\|\alpha^* - \bar{\alpha}^*\| + \|\varphi_T(\alpha^*) - \varphi(\alpha^*)\|$$

(A.21)
$$\leq \frac{\delta L}{\hat{c}}\Gamma(T) + (1-\mu\vartheta)^T\hat{L}\|\omega^0 - \omega^*(\alpha_t)\|,$$

where the second inequality is due to the Cauchy-Schwarz inequality and the Lipschitz assumption of the function $\varphi(\alpha)$, and the third inequality is due to the Lipschitz assumption of the functions F_i w.r.t ω. This finishes the proof. ■

A.1.2.6 Proof of Theorem 3.6

Proof. For a given $\widetilde{\gamma}_k$, the function $\Lambda(\varphi_T(\alpha),\widetilde{\gamma}_k)$ is c-strongly-convex. Since we have $\Lambda(\nabla\varphi_T(\alpha^*),\widetilde{\gamma}_k)^\top(\alpha_k-\alpha^*)\geq 0$, then

$$\Lambda(\nabla\varphi_T(\alpha_k),\widetilde{\gamma}_k)^\top(\alpha_k-\alpha^*)\leq[\Lambda(\nabla\varphi_T(\alpha_k),\widetilde{\gamma}_k)-\Lambda(\nabla\varphi_T(\alpha^*),\widetilde{\gamma}_k)]^\top(\alpha_k-\alpha^*) \tag{A.22}$$

$$\leq c\|\alpha_k-\alpha^*\|^2. \tag{A.23}$$

Then, plugging inequalities (A.6) and (A.7) into (A.22), we have

$$\|A_{k+1}\|^2\leq(1-2\nu_k c)\|A_k\|^2+\nu_k^2L^2. \tag{A.24}$$

Therefore, with $\nu_k=\xi/k$ and $\xi\geq 1/2c$, we have $\|\alpha_k-\alpha^*\|\leq\max\{2\xi L(2c\xi-1)^{-1},L\|\alpha^0-\alpha^*\|^2\}/k$. Therefore, for a given $\widetilde{\gamma}^*$, we have

$$\|\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}^*)-\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)\| \tag{A.25}$$

$$\leq\|\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}^*)-\Lambda(\varphi_T(\alpha^*),\widetilde{\gamma}^*)\|+\|\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}^*)-\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)\| \tag{A.26}$$

$$\leq M\|\alpha_k-\alpha^*\|+\|\Lambda(\varphi_T(\alpha_k),\widetilde{\gamma}^*)-\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)\|\|, \tag{A.27}$$

where the first inequality is due to the triangle inequality and the second inequality is due to the Cauchy-Schwarz inequality and the Lipschitz assumption of the function $\varphi_T(\alpha)$. Thus, the proof is completed by using the upper bound of $\|\alpha_k-\alpha^*\|$ and the result in Theorem 3.5 directly. ■

A.1.3 Proofs of Theorems in Section 3.5.3

A.1.3.1 Lemmas

We first provide the following lemmas for the LL subproblem under Assumption 3.4.

Lemma A.1. *Under Assumption 3.4, we have the following results.*

(a) $\|\nabla\widetilde{q}(z)-\nabla q(z)\|\leq L_f\|\tilde{\omega}^T-\omega^*(\alpha)\|$.

(b) The function $\nabla q(z)$ *is* L_q*-Lipschitz continuous w.r.t.* z*, where* $L_q=L_f(2+\frac{L_f}{c})$.

(c) If the step size of the LL subproblem satisfies $\eta\leq\frac{2}{L_f}$*, then for* $\tilde{\omega}^0=\omega$ *and* $\tilde{\omega}^{T+1}=\tilde{\omega}^T-\eta\nabla_\omega q(\alpha,\tilde{\omega}^T)$*, we have* $q(\alpha,\tilde{\omega}^T)\leq\Gamma(T)q(\alpha,\omega)$*, where* $\Gamma(T)$ *represents an exponential decay w.r.t.* T.

(d) $\|\nabla_z q(z)\|^2\leq\frac{2L_q^2}{c}q(z)$

Proof. (a) We have $\|\nabla\widetilde{q}(z)-\nabla q(z)\| \le \|\nabla_\alpha f(\alpha,\tilde{\omega}^T)-\nabla_\alpha f(\alpha,\omega^*)\| \le L_f\|\tilde{\omega}^T-\omega^*(\alpha)\|$.

(b) This result can be directly obtained from Lemma 5 in [109].

(c) Since $\nabla_\omega q(z)=\nabla_\omega f(z)$ and $\nabla_z f(z)$ is L_f-Lipschitz continuous w.r.t ω, $\nabla_\omega q(z)$ is L_f-Lipschitz continuous w.r.t ω. Then we have

$$\begin{aligned} q(\alpha,\tilde{\omega}^{T+1}) &\le q(\alpha,\tilde{\omega}^T)-(\eta-\frac{L_f\eta^2}{2})\|\nabla_\omega q(\alpha,\tilde{\omega}^T)\|^2 \\ &= q(\alpha,\tilde{\omega}^T)-(\eta-\frac{L_f\eta^2}{2})\|\nabla_\omega f(\alpha,\tilde{\omega}^T)\|^2 \\ &\le (1-c(2\eta-L_f\eta^2))q(\alpha,\tilde{\omega}^T), \end{aligned}$$

where the second inequality is due to $\|\nabla_\omega f(\alpha,\tilde{\omega}^T)\|^2 \ge 2c(f(\alpha,\tilde{\omega}^T)-f(\alpha,\omega^*))=2cq(\alpha,\tilde{\omega}^T)$. Since $\tilde{\omega}^0=\omega$, we have $q(\alpha,\tilde{\omega}^T)\le(1-c(2\eta-L_f\eta^2))^T q(\alpha,\omega)$. If $\eta\le\frac{2}{L_f}$, $2\eta-L_f\eta^2\ge 0$. Let $\Gamma(T)=(1-c(2\eta-L_f\eta^2))^T$, which decays exponentially w.r.t. T. Then we reach the conclusion.

(d) Since $\nabla q(\alpha,\omega^*(\alpha))=0$, we have $\|\nabla q(z)\|^2=\|\nabla q(z)-\nabla q(\alpha,\omega^*(\alpha))\|^2\le L_q^2\|\omega-\omega^*(\alpha)\|^2$. Since $f(z)$ is c-strongly convex with respect to ω, we have $\|\omega-\omega^*(\alpha)\|^2\le\frac{2}{c}(f(\alpha,\omega)-f(\alpha,\omega^*(\alpha))=\frac{2q(z)}{c}$, then we reach the conclusion.

■

Then for the Algorithm 2, we have following result about the constraint function.

Lemma A.2. *Under Assumption 3.4, suppose the sequence $\{z_k\}_{k=0}^K$ generated by Algorithm 2 satisifes $q(z_k)\le B$, where B is a positive constant. Then there exists a constant $C>0$, if $T\ge C$, the following results hold.*

(a) $q(z_k)\le\Gamma_1(k)B+\mathcal{O}(\Gamma(T)+\mu)$, *where $\Gamma_1(k)$ represents an exponential decay w.r.t k.*

(b) $\|\nabla\widetilde{q}(z)-\nabla q(z)\|\le L_f\sqrt{\frac{\Gamma(T)}{c^2}}\|\nabla q(\alpha,\omega)\|$.

(c) There exists a positive constant $C_b<1$, such that $\|\nabla\widetilde{q}(z)\|\ge C_b\|\nabla q(z)\|$.

(d) $\sum_{k=0}^{K-1}\|\nabla\widetilde{q}(z_k)\|^2=\mathcal{O}(\frac{1}{\mu}+K\Gamma(T)+K\mu)$.

Proof. (a) According to Lemma A.1 (d) and the boundedness assumption of $q(z_k)$, the gradient norm $\|\nabla q(z_k)\|$ is also bounded. Let $G(z_k)=\sum_{i=1}^m\tilde{\lambda}_i^k\nabla F_i(z_k)$, we have $d=-\mu(G(z_k)+\nu\nabla q(z_k))$ and $\nu=\max(\frac{\rho\|\nabla\widetilde{q}(z_k)\|^2-\langle\nabla\widetilde{q}(z_k),G(z_k)\rangle}{\|\nabla\widetilde{q}(z_k)\|^2},0)$. Then we can applying

Lemma A.1 to Lemma 10 in [67], we obtain that there exist a constant $C > 0$, if $T \geq C$, we have

$$q(\alpha_k, \omega_k) \leq \Gamma_1(k) q(\alpha_0, \omega_0) + \mathcal{O}(\Gamma(T) + \mu), \tag{A.28}$$

where $\Gamma_1(k) = (1 - \mu C_a)^k$ represents an exponential decay w.r.t k and C_a is a positive constant depending on η and c. Then we reach the conclusion.

(b) We have $\|\nabla \widetilde{q}(z) - \nabla q(z)\| \leq L_f \|\tilde{\omega}^T - \omega^*(\alpha)\| \leq L_f \sqrt{\frac{2(f(\alpha,\tilde{\omega}^T) - f(\alpha,\omega^*(\alpha)))}{c}}$, where the first inequality is due to Lemma A.1 (a). Then we have

$$\|\nabla \widetilde{q}(z) - \nabla q(z)\| \leq L_f \sqrt{\frac{2\Gamma(T) q(z)}{c}} \leq L_f \sqrt{\frac{\Gamma(T)}{c^2}} \|\nabla q(\alpha, \omega)\|,$$

where the first inequality is due to Lemma A.1 (c) and the second inequality is due to the strongly convex assumption of $f(\alpha, \omega)$ w.r.t ω and $\|\nabla_\omega q(\alpha, \omega)\| \leq \|\nabla q(z)\|$.

(c) By the triangle inequality and Lemma A.2 (b), we obtain

$$\|\nabla \widetilde{q}(z)\| \geq \|\nabla q(z)\| - \|\nabla \widetilde{q}(z) - \nabla q(z)\| \geq (1 - L_f \sqrt{\frac{\Gamma(T)}{c^2}}) \|\nabla q(z)\|.$$

Then there exists a positive constant $C > 0$ such that when $T \geq C$, we have $0 < L_f \sqrt{\frac{\Gamma(C)}{c^2}} < 1$. Let $C_b = 1 - L_f \sqrt{\frac{\Gamma(C)}{c^2}}$, then we have $\|\nabla \widetilde{q}(z)\| \geq C_b \|\nabla q(z)\|$ and $C_b < 1$.

(d) By the triangle inequality and Lemma A.2 (b), we obtain

$$\|\nabla \widetilde{q}(z)\| \leq \|\nabla \widetilde{q}(z) - \nabla q(z)\| + \|\nabla q(z)\| \leq (1 + L_f \sqrt{\frac{\Gamma(T)}{c^2}}) \|\nabla q(z)\|.$$

By defining $C_e = \left(1 + L_f \sqrt{\frac{\Gamma(T)}{c^2}}\right)^2$, we have

$$\sum_{k=0}^{K-1} \|\nabla \widetilde{q}(z_k)\|^2 \leq \sum_{k=0}^{K-1} C_e \|\nabla q(z_k)\|^2 \leq \sum_{k=0}^{K-1} \frac{2L_q^2 C_e}{c} (\Gamma_1(k) B + \mathcal{O}(\Gamma(T) + \mu)),$$

where the second inequality is due to Lemma A.1 (d) and Lemma A.2 (a). Since $\sum_{k=0}^{K} (1 - \mu C_a)^k = \mathcal{O}(\frac{1}{\mu})$ and C_e decays to 1 as $T \to +\infty$, we have $\sum_{k=0}^{K-1} \|\nabla \widetilde{q}(z_k)\|^2 = \mathcal{O}(\frac{1}{\mu} + K\Gamma(T) + K\mu)$.

■

Remark A.3. *Lemma A.2 (a) uses the result of Lemma 10 in [67]. It is worth noting that the assumptions 1 and 3 in [67], which assume that the LL subproblem both satisfies PL-inequality and the gradient boundedness, are self-contradicted. In our analysis, we address this theoretical problem by assuming that $q(z_k)$ is bounded for the generated sequence $\{z_k\}_{k=1}^K$. This means the convergence can still be held locally for a finite sequence $\{z_k\}_{k=1}^K$.*

A.1.3.2 Proof of the Theorem 3.7

Since $F_i(z)$ is L_F-Lipschitz continuous, we have

$$\begin{aligned}\sum_{i=1}^{m}\tilde{\lambda}_i^k(F_i(z_{k+1})-F_i(z_k)) &\le \mu\left\langle\sum_{i=1}^{m}\tilde{\lambda}_i^k\nabla F_i(z_k),d_k\right\rangle+\frac{\sum_{i=1}^{m}\tilde{\lambda}_i^k L_F\mu^2}{2}\|d_k\|^2\\ &=\mu\langle-\nu_k\nabla\widetilde{q}(z_k)-d_k,d_k\rangle+\frac{L_F\mu^2}{2}\|d_k\|^2\\ &=(\frac{L_F\mu^2}{2}-\mu)\|d_k\|^2-\mu\nu_k\langle\nabla\widetilde{q}(z_k),d_k\rangle.\end{aligned}$$

According to the complementary slackness condition of problem (3.20), We have $\nu_k(\langle\nabla\widetilde{q}(z_k),d_k\rangle+\phi_k)=0$, where $\nu_k=\nu(\tilde{\lambda}^k)$. Therefore $-\nu_k\langle\nabla\widetilde{q}(z_k),d_k\rangle=\nu_k\frac{\rho}{2}\|\nabla\widetilde{q}(z_k)\|^2$. Then if $\mu\le\frac{1}{L_F}$, we have

$$\sum_{i=1}^{m}\tilde{\lambda}_i^k(F_i(z_{k+1})-F_i(z_k))\le-\frac{\mu}{2}\|d_k\|^2+\frac{\rho\mu\nu_k}{2}\|\nabla\widetilde{q}(z_k)\|^2. \tag{A.29}$$

Let $G(z_k)=\sum_{i=1}^{m}\tilde{\lambda}_i^k\nabla F_i(z_k)$, we have $\nu_k\|\nabla\widetilde{q}(z_k)\|^2\le\rho\|\nabla\widetilde{q}(z_k)\|^2-\langle G(z_k),\nabla\widetilde{q}(z_k)\rangle$. Summing the inequality (A.29) over $k=0,1,\ldots,K-1$ yields

$$\begin{aligned}\sum_{k=0}^{K-1}\sum_{i=1}^{m}\tilde{\lambda}_i^k(F_i(z_{k+1})-F_i(z_k)) &\le\sum_{k=0}^{K-1}(-\frac{\mu}{2}\|d_k\|^2+\frac{\rho\mu\nu_k}{2}\|\nabla\widetilde{q}(z_k)\|^2)\\ &\le-\frac{\mu}{2}\sum_{k=0}^{K-1}\|d_k\|^2+\sum_{k=0}^{K-1}\frac{\rho^2\mu}{2}\|\nabla\widetilde{q}(z_k)\|^2-\sum_{k=0}^{K-1}\frac{\rho\mu}{2}\langle G(z_k),\nabla\widetilde{q}(z_k)\rangle\\ &\le-\frac{\mu}{2}\sum_{k=0}^{K-1}\|d_k\|^2+\sum_{k=0}^{K-1}\frac{\rho^2\mu}{2}\|\nabla\widetilde{q}(z_k)\|^2+\sum_{k=0}^{K-1}\frac{\rho\mu M}{2}\|\nabla\widetilde{q}(z_k)\|,\end{aligned}$$

where the last inequality is by Cauchy-Schwarz inequality and $\|G(z_k)\|\le\|\nabla_z F_i(z_k)\|\le M$. Therefore, we further have

$$\sum_{k=0}^{K-1}\|d_k\|^2\le\frac{2\sum_{k=0}^{K-1}\sum_{i=1}^{m}\tilde{\lambda}_i^k(F_i(z_{k+1})-F_i(z_k))}{\mu}+\sum_{k=0}^{K-1}\rho^2\|\nabla\widetilde{q}(z_k)\|^2+\sum_{k=0}^{K-1}\rho M\|\nabla\widetilde{q}(z_k)\|. \tag{A.30}$$

For the first term of the right-hand side of the inequality (A.30), we have

$$\begin{aligned}\sum_{k=0}^{K-1}\sum_{i=1}^{m}\tilde{\lambda}_i^k(F_i(z_{k+1})-F_i(z_k)) &=\sum_{k=0}^{K-1}\sum_{i=1}^{m}(\tilde{\lambda}_i^k-\tilde{\lambda}_i^{k+1})F(z_{k+1})+\sum_{i=1}^{m}(\tilde{\lambda}_i^{K-1}F(z_K)-\tilde{\lambda}_i^0F(z_0))\\ &\le\sum_{k=0}^{K-1}\sum_{i=1}^{m}|\tilde{\lambda}_i^k-(1-\beta)\tilde{\lambda}_i^k-\beta\lambda_i^{k+1}|M+2M\\ &\le\sum_{k=0}^{K-1}\beta\sum_{i=1}^{m}|\tilde{\lambda}_i^k-\lambda_i^{k+1}|M+2M.\end{aligned}$$

Since $\lambda \in \Delta^{m-1}$, we have $|\tilde{\lambda}_i^k - \lambda_i^{k+1}| \leq 2$. Then we have

$$\begin{aligned}\sum_{k=0}^{K-1} \|d_k\|^2 &\leq \frac{2mMK\beta + 2M}{\mu} + \sum_{k=0}^{K-1} \rho^2 \|\nabla \widetilde{q}(z_k)\|^2 + \rho M\sqrt{K}\sqrt{\sum_{k=0}^{K-1} \|\nabla \widetilde{q}(z_k)\|^2} \\ &= \mathcal{O}(\frac{K\beta}{\mu}) + \mathcal{O}(\frac{M}{\mu}) + \mathcal{O}(\frac{1}{\mu} + K\Gamma(T) + K\mu) + \mathcal{O}(\sqrt{\frac{K}{\mu}} + K\sqrt{\Gamma(T)} + K\sqrt{\mu}) \\ &= \mathcal{O}(\frac{K\beta}{\mu} + K\Gamma(T) + \sqrt{\frac{K}{\mu}} + K\sqrt{\mu}),\end{aligned}$$

where the first inequality is by Holder's inequality and the first equality is due to Lemma A.2 (d). For the measure of stationarity $\mathscr{K}(z_k)$, we obtain that

$$\mathscr{K}(z_k) = \|G(z_k) + \nu_k \nabla q(z_k)\|^2 \leq 2\|d_k\|^2 + 2\|\nu_k(\nabla \widetilde{q}(z_k) - \nabla q(z_k))\|^2. \tag{A.31}$$

According to A.1 (d), we have $\|\nabla q(z_k)\| \leq \sqrt{\frac{2BL_q^2}{c}}$. Then we obtain

$$\begin{aligned}\|\nu_k(\nabla \widetilde{q}(z_k) - \nabla q(z_k))\| &\leq \left|\rho - \frac{\langle G(z_k), \nabla \widetilde{q}(z_k)\rangle}{\|\nabla \widetilde{q}(z_k)\|^2}\right| \|\nabla \widetilde{q}(z_k) - \nabla q(z_k)\| \\ &\leq \rho\|\nabla \widetilde{q}(z_k) - \nabla q(z_k)\| + |\langle G(z_k), \frac{\nabla \widetilde{q}(z_k)}{\|\nabla \widetilde{q}(z_k)\|}\rangle| \frac{\|\nabla \widetilde{q}(z_k) - \nabla q(z_k)\|}{\|\nabla \widetilde{q}(z_k)\|} \\ &\leq L_f\sqrt{\frac{\Gamma(T)}{c^2}}(\rho\|\nabla q(z_k)\| + \frac{1}{C_b}|\langle G(z_k), \frac{\nabla \widetilde{q}(z_k)}{\|\nabla \widetilde{q}(z_k)\|}\rangle|) \\ &\leq L_f\sqrt{\frac{\Gamma(T)}{c^2}}(\rho L_q\sqrt{\frac{2B}{c}} + \frac{M}{C_b}),\end{aligned}$$

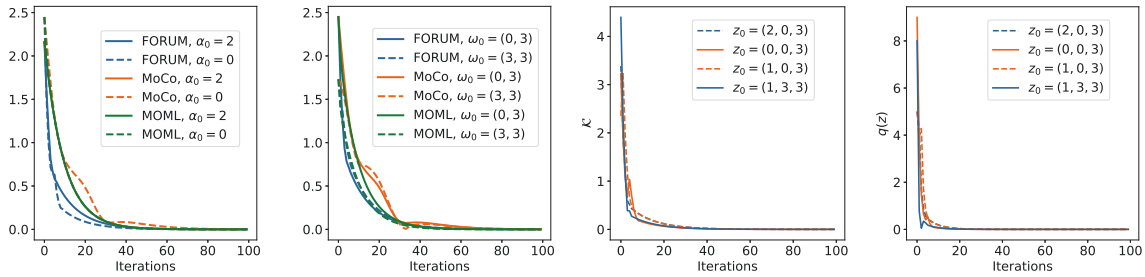
where the third inequality is due to Lemma A.2 (b) and (c), and the last inequality is due to the Cauchy-Schwarz inequality. Therefore, we have $\|\nu_k(\nabla \widetilde{q}(z_k) - \nabla q(z_k))\|^2 = \mathcal{O}(\Gamma(T))$. Then we can get

$$\begin{aligned}\min_{k<K} \mathscr{K}(z_k) &\leq \frac{1}{K}\sum_{k=0}^{K-1} \mathscr{K}(z_k) \\ &= 2\mathcal{O}(\frac{K\beta}{\mu K} + \Gamma(T) + \sqrt{\frac{1}{\mu K}} + \sqrt{\mu}) + 2\mathcal{O}(\Gamma(T)) \\ &= \mathcal{O}(\frac{\beta}{\mu} + \Gamma(T) + \sqrt{\frac{1}{\mu K}} + \sqrt{\mu}),\end{aligned}$$

where the first equality is due to Eq. (A.31). According to Lemma A.2 (a), we obtain $q(z_k) = \mathcal{O}(\Gamma_1(k) + \Gamma(T) + \mu)$. Thus we get

$$\max\left\{\min_{k<K} \mathscr{K}(z_k), q(z_k)\right\} = \mathcal{O}\left(\sqrt{\mu} + \sqrt{\frac{1}{\mu K}} + \frac{\beta}{\mu} + \Gamma(T)\right).$$

Let $\mu = \mathcal{O}(K^{-1/2})$ and $\beta = \mathcal{O}(K^{-3/4})$, we reach the conclusion.



(a) $\mathscr{E}$ vs. K. (b) $\mathscr{E}$ vs. K. (c) $\mathscr{K}(z)$ vs. K. (d) $q(z)$ vs. K.

Figure A.1: Results on the problem (A.32) with different initialization points. **(a)**: Fix $\omega_0 = (0,3)$ and vary $\alpha_0 = 0,2$. The optimality gap $\mathscr{E}$ curves. **(b)**: Fix $\alpha_0 = 2$ and vary $\omega_0 = (0,3),(3,3)$. The optimality gap $\mathscr{E}$ curves. **(c)**: The stationarity gap $\mathscr{K}(z)$ curves. **(d)**: The value of the constraint function $q(z)$ curves.

A.1.4 Synthetic MOBLO

In this section, we use one synthetic MOBLO problem to illustrate the convergence of the proposed FORUM method. We first consider the following problem,

$$\min_{\alpha\in\mathbb{R},\omega\in\mathbb{R}^2} (\|\omega-(1,\alpha)\|_2^2, \|\omega-(2,\alpha)\|_2^2) \quad \text{s.t.} \quad \omega \in \arg\min_{\omega\in\mathbb{R}^2} \|\omega-\alpha\|_2^2, \tag{A.32}$$

where $(\cdot,\cdot)$ denotes a two-dimensional vector and $\omega = (\omega_1,\omega_2)$. By simple calculation, we can find that the optimal solution set of problem (A.32) is $\mathscr{P} = \{(\alpha,\omega) \mid \alpha = \omega_1 = \omega_2 = c, c \in [1,2]\}$.

We apply GD optimizer for both UL and LL subproblems and the step sizes are set to $\mu = 0.3$ and $\eta = 0.05$ for all methods. We run 50 LL iterations to ensure that for a given α, they all reach the minimum point for the LL subproblem. For FORUM, we set $\rho = 0.3$ and $\beta_k = (k+1)^{-3/4}$. The result is evaluated by calculating the Euclidean distance between solution z and the optimal solution set $\mathscr{P}$, which is denoted by $\mathscr{E} = \text{dist}(z,\mathscr{P})$.

Figures A.1(a) and A.1(b) show the numerical results of the MOML, MoCo, and FORUM methods with different initializations. It can be observed that the proposed FORUM method can achieve an optimal solution in all the settings, i.e., $\mathscr{E} \to 0$, and different initializations only slightly affect the convergence speed. Figures A.1(c) and A.1(d) show that both $\mathscr{K}(z)$ and $q(z)$ converge to zero in all the settings. Thus FORUM solves the corresponding constrained optimization problem. This result demonstrates our convergence result.

A.2 Additional Material Chapter 4

A.2.1 Proof of the Result in Section 4.4.1

A.2.1.1 Proof of updated rule

The objective of the inner minimization of problem (4.6) can be rewritten as

$$\begin{aligned}\langle\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\theta}}J_i(\boldsymbol{\theta}_t),\boldsymbol{\theta}\rangle+\frac{1}{\beta_t}\mathrm{KL}(p_{\boldsymbol{\theta}}\|p_{\boldsymbol{\theta}_t})&=\boldsymbol{\mu}^\top(\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t))+\sum_{i=1}^{m}\lambda_i\mathrm{tr}(\boldsymbol{\Sigma}\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t))\\&+\frac{1}{2\beta_t}\left[\mathrm{tr}(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma})+(\boldsymbol{\mu}-\boldsymbol{\mu}_t)^\top\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_t)+\log\frac{|\boldsymbol{\Sigma}_t|}{|\boldsymbol{\Sigma}|}-d\right],\end{aligned}\tag{A.33}$$

where $\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)$ and $\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)$ denotes the derivative w.r.t $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ taking at $\boldsymbol{\mu}=\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_t$, respectively. We can see the above problem is convex with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Taking the derivative w.r.t $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and setting them to zero, we can obtain that

$$\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)+\frac{1}{\beta_t}\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_t)=0,\tag{A.34}$$

$$\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)+\frac{1}{2\beta_t}[\boldsymbol{\Sigma}_t^{-1}-\boldsymbol{\Sigma}^{-1}]=0.\tag{A.35}$$

Substituting the above equalities into the regularization term of the objective of the outer optimization problem we have

$$\begin{aligned}\frac{1}{\beta_t}\mathrm{KL}(p_{\boldsymbol{\theta}}\|p_{\boldsymbol{\theta}_t})&=\frac{1}{2\beta_t}\left[\mathrm{tr}(\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma})+(\boldsymbol{\mu}-\boldsymbol{\mu}_t)^\top\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_t)+\log\frac{|\boldsymbol{\Sigma}_t|}{|\boldsymbol{\Sigma}|}-d\right]\\&=\frac{1}{2\beta_t}\mathrm{tr}\Big(\boldsymbol{I}-2\beta_t\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)\boldsymbol{\Sigma}\Big)-\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_t)^\top\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\\&\quad+\frac{1}{2\beta_t}\log\left(|\boldsymbol{\Sigma}_t(\boldsymbol{\Sigma}_t^{-1}+2\beta_t\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t))|\right)-\frac{d}{2\beta_t}\end{aligned}\tag{A.36}$$

$$=\frac{d}{2\beta_t}-\langle\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t),\boldsymbol{\Sigma}\rangle-\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_t)^\top\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta})\tag{A.37}$$

$$+\frac{1}{2\beta_t}\log(|\boldsymbol{I}+2\beta_t\boldsymbol{\Sigma}_t\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)|)-\frac{d}{2\beta_t}\tag{A.38}$$

$$=-\langle\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t),\boldsymbol{\Sigma}-\boldsymbol{\Sigma}_t\rangle-\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_t)^\top\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta})+\frac{Q_t}{2\beta_t}\tag{A.39}$$

where Q_t is given as below.

$$Q_t=\log(|\boldsymbol{I}+2\beta_t\boldsymbol{\Sigma}_t\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)|)-2\beta_t\langle\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t),\boldsymbol{\Sigma}_t\rangle\tag{A.40}$$

$$=\log(|\boldsymbol{I}+2\beta_t\boldsymbol{\Sigma}_t\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)|)-\mathrm{tr}(2\beta_t\boldsymbol{\Sigma}_t\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)).\tag{A.41}$$

Since $\boldsymbol{\Sigma}_t$ and $\nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)$ are both diagonal matrix, we denote $\text{diag}(\boldsymbol{\Sigma}_t \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)) = (v_t^1, \ldots, v_t^d)$ in t-th iteration, then we have

(A.42)
$$Q_t = \log(\prod_{i=1}^d (1+2\beta_t v_t^i)) - \sum_{i=1}^d 2\beta_t v_t^i = \sum_{i=1}^d \left(\log(1+2\beta_t v_t^i) - 2\beta_t v_t^i\right) = \sum_{i=1}^d -2\beta_t^2 (v_t^i)^2 + \mathcal{O}(\beta_t^3 (v_t^i)^3),$$

where the last equality is due to the Taylor expansion. Note that $\mathcal{O}(\beta_t^3 (v_t^i)^3)$ decrease to zero when $\beta_t \to 0$. We can approximate Q_t by $\sum_{i=1}^d -2\beta_t^2 (v_t^i)^2$. Then substituting Eqs. (A.34) (A.39) and Q_t into the outer optimization problem of problem (4.6), we have

(A.43)
$$\langle \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\theta} - \boldsymbol{\theta}_t \rangle + \frac{1}{\beta_t} \text{KL}(p_{\boldsymbol{\theta}} \| p_{\boldsymbol{\theta}_t})$$

(A.44)
$$= \langle \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\mu} - \boldsymbol{\mu}_t \rangle + \langle \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\Sigma} - \boldsymbol{\Sigma}_t \rangle + \frac{1}{\beta_t} \text{KL}(p_{\boldsymbol{\theta}} \| p_{\boldsymbol{\theta}_t})$$

(A.45)
$$= \frac{1}{2} \langle \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\mu} - \boldsymbol{\mu}_t \rangle + \frac{Q_t}{2\beta_t}$$

(A.46)
$$= -\frac{\beta_t}{2} \langle \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\Sigma}_t \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t) \rangle - \beta_t \langle \boldsymbol{\Sigma}_t \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t), \boldsymbol{\Sigma}_t \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t) \rangle .$$

Therefore, the outer optimization problem is equivalent to the following problem

(A.47)
$$\min_{\boldsymbol{\lambda}^t \in \Delta^{m-1}} \left\| \boldsymbol{\Sigma}_t^{\frac{1}{2}} \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t) \right\|^2 + 2 \left\| \text{diag}(\boldsymbol{\Sigma}_t \sum_{i=1}^m \lambda_i \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)) \right\|^2,$$

where we reach the result in Eq. (4.10).

A.2.1.2 Proof of Theorem 4.1

We now provide the proof of the gradient of $\mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})]$ w.r.t $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

(A.48)
$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})] = \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x}) \nabla_{\boldsymbol{\mu}} \log(p(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}))]$$

(A.49)
$$= \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x}) \nabla_{\boldsymbol{\mu}} (-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})]$$

(A.50)
$$= \mathbb{E}_{p_{\boldsymbol{\theta}}}[\boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) F_i(\boldsymbol{x})].$$

We further have

$$\nabla_{\boldsymbol{\Sigma}}\mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})] = \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})\nabla_{\boldsymbol{\Sigma}}\log(p(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}))] \tag{A.51}$$

$$= \mathbb{E}_{p_{\boldsymbol{\theta}}}[F_i(\boldsymbol{x})\nabla_{\boldsymbol{\Sigma}}(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}))] \tag{A.52}$$

$$= \frac{1}{2}\mathbb{E}_{p_{\boldsymbol{\theta}}}[\big(\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\big)F_i(\boldsymbol{x})], \tag{A.53}$$

where we reach the conclusion.

A.2.2 Technical Lemmas

In this section, we introduce the following technical lemmas for analysis. The proof of all technical lemmas is put in Appendix A.2.4.

Lemma A.3. *Suppose $\boldsymbol{\Sigma}$ and $\hat{\boldsymbol{\Sigma}}$ are two d-dimensional diagonal matrix and $\boldsymbol{z}$ is a d-dimensional vector, then we have $\|\boldsymbol{\Sigma}\boldsymbol{z}\| \le \|\boldsymbol{\Sigma}\|_F\|\boldsymbol{z}\|$ and $\|\boldsymbol{\Sigma}\hat{\boldsymbol{\Sigma}}\|_F \le \|\boldsymbol{\Sigma}\|_F\|\hat{\boldsymbol{\Sigma}}\|_F$.*

Lemma A.4. *Given a convex function $f(\boldsymbol{x})$, for Gaussian distribution with parameters $\boldsymbol{\theta} := \{\boldsymbol{\mu}, \boldsymbol{\Sigma}^{\frac{1}{2}}\}$, let $\bar{J}(\boldsymbol{\theta}) := \mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\theta})}[f(\boldsymbol{x})]$. Then $\bar{J}(\boldsymbol{\theta})$ is a convex function with respect to $\boldsymbol{\theta}$.*

Lemma A.5. *Suppose that the gradient $\hat{G}_i$ are positive semi-definite matrix and satisfies $\xi\boldsymbol{I} \preceq \hat{G}_i \preceq b\boldsymbol{I}$. Then for algorithm 1, we have the following results.*

(a) The (diagonal) covariance matrix $\boldsymbol{\Sigma}_T$ satisfies

$$\frac{1}{2b\sum_{t=1}^T\beta_t\boldsymbol{I} + \boldsymbol{\Sigma}_0^{-1}} \preceq \boldsymbol{\Sigma}_T \preceq \frac{1}{2\xi\sum_{t=1}^T\beta_t\boldsymbol{I} + \boldsymbol{\Sigma}_0^{-1}}.$$

(b) $\|\boldsymbol{\Sigma}_t\|_F \le \frac{\sqrt{d}}{2\xi\sum_{t=1}^T\beta_t}$.

(c) $\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F \le \frac{b\beta_t d^{\frac{3}{2}}}{2\xi^2(\sum_{t=1}^T\beta_t)^2}$.

Lemma A.6. *Suppose the gradient estimator $\hat{\boldsymbol{g}}_i^t$ for the i-th objective in t-th iteration as*

$$\hat{\boldsymbol{g}}_i^t = \boldsymbol{\Sigma}_t^{-\frac{1}{2}}\boldsymbol{z}\big(F_i(\boldsymbol{\mu}_t + \boldsymbol{\Sigma}_t^{\frac{1}{2}}\boldsymbol{z}) - F_i(\boldsymbol{\mu}_t)\big),$$

where $\boldsymbol{z} \sim \mathcal{N}(0, I)$. Suppose assumption 4.1 holds, the gradient $\hat{G}_i$ are positive semi-definite matrix and satisfies $\xi\boldsymbol{I} \preceq \hat{G}_i \preceq b\boldsymbol{I}$ and $\boldsymbol{\Sigma}_0 \preceq R\boldsymbol{I}$, where $\xi, b, R \ge 0$. Then we have

(a) $\hat{\boldsymbol{g}}_i^t$ is an unbiased estimator of the gradient $\nabla_{\boldsymbol{\mu}}\mathbb{E}_{p_{\boldsymbol{\theta}_t}}[F_i(\boldsymbol{x})]$.

(b) $\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t\|^2] \leq \frac{H^2(d+4)^2}{4\xi^2(\sum_{k=1}^t \beta_k)^2}$.

(c) $\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t] = \mathbb{E}_{\boldsymbol{z}}[\|\hat{\boldsymbol{g}}_i^t - \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2] \leq \frac{H^2 C(d+4)^2}{N}$, *where* $C = \max(\frac{b}{\xi}, \|\boldsymbol{\Sigma}_0^{-1}\|_\infty)$.

Lemma A.7. *Suppose* $\boldsymbol{\lambda}^t = (1-\gamma_t)\boldsymbol{\lambda}^{t-1} + \gamma_t \tilde{\boldsymbol{\lambda}}^t$, *then we have* $\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] = \mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\lambda}^t - \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\|^2] \leq 2\gamma_t^2$.

Lemma A.8. *Suppose assumption 4.1 holds, if* $\hat{\boldsymbol{g}}_1^t, \ldots, \hat{\boldsymbol{g}}_m^t$ *are unbiased estimates of* $\nabla_{\boldsymbol{\mu}} J_1(\boldsymbol{\theta}_t), \ldots, \nabla_{\boldsymbol{\mu}} J_m(\boldsymbol{\theta}_t)$. *Further suppose that each gradient variance is bounded by* $\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t] = \mathbb{E}_{\boldsymbol{z}}[\|\hat{\boldsymbol{g}}_i^t - \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2] \leq \delta$, $i = 1, \ldots, m$ *and let* $\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] = \mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\lambda}^t - \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\|^2]$. *Then for any gradient descent algorithm updated with composite gradient* $\boldsymbol{q}_t = -\sum_{i=1}^m \lambda_i^t \hat{\boldsymbol{g}}_i^t$ *with* $\boldsymbol{\lambda}^t \in \Delta^{m-1}$, *we have following inequality in t-th iteration,*

(a) $\|\mathbb{E}_{\boldsymbol{z}}[-\boldsymbol{q}_t] - \mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)]\|^2 \leq \mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] \sum_{i=1}^m \mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]$.

(b) $\mathbb{E}_{\boldsymbol{z}}[(\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t))^\top \boldsymbol{q}_t] \leq 2H\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] \sum_{i=1}^m \mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} - \mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2]$.

(c) $\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{q}_t\|^2 - \|\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2] \leq \sum_{i=1}^m \mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t] + 4H\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] \sum_{i=1}^m \mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]}$.

A.2.3 Proof of the Result in Section 4.5

In this section, we provide the proof of the result in Section 4.5.

Theorem 4.2 can be directly obtained by Lemma A.5 (a).

A.2.3.1 Proof of the Proposition 4.3

From the definition of $J_i(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we know that $F_i(\boldsymbol{\mu}^*) = J_i(\boldsymbol{\mu}^*, \mathbf{0})$. Note that $F_i(\boldsymbol{x})$ is a convex function, we have that

$$F_i(\boldsymbol{\mu}) = F_i(\mathbb{E}_{\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})}[\boldsymbol{x}]) \leq \mathbb{E}_{\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})}[F_i(\boldsymbol{x})] = J_i(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{A.54}$$

It follows that

$$F_i(\boldsymbol{\mu}) - F_i(\boldsymbol{\mu}^*) \leq J_i(\boldsymbol{\mu}, \boldsymbol{\Sigma}) - J_i(\boldsymbol{\mu}^*, \mathbf{0}). \tag{A.55}$$

Then we have

$$\sum_{i=1}^m \lambda_i (F_i(\boldsymbol{\mu}) - F_i(\boldsymbol{\mu}^*)) \leq \sum_{i=1}^m \lambda_i (J_i(\boldsymbol{\mu}, \boldsymbol{\Sigma}) - J_i(\boldsymbol{\mu}^*, \mathbf{0})), \tag{A.56}$$

where we reach the conclusion.

A.2.3.2 Proof of the Proposition 4.4

Note that $\|\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}^*}J_i(\boldsymbol{\mu}^*)\|=0$, we have

$$\|\sum_{i=1}^{m}\lambda_i\nabla F_i(\boldsymbol{\mu}^*)\|^2=\left\|\sum_{i=1}^{m}\lambda_i\nabla F_i(\boldsymbol{\mu}^*)-\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}^*}J_i(\boldsymbol{\mu}^*)+\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}^*}J_i(\boldsymbol{\mu}^*)\right\|^2 \tag{A.57}$$

$$=\left\|\sum_{i=1}^{m}\lambda_i\nabla F_i(\boldsymbol{\mu}^*)-\sum_{i=1}^{m}\lambda_i\nabla_{\boldsymbol{\mu}^*}J_i(\boldsymbol{\mu}^*)\right\|^2. \tag{A.58}$$

It follows that

$$\|\sum_{i=1}^{m}\lambda_i\nabla F_i(\boldsymbol{\mu}^*)\|_2^2\leq\sum_{i=1}^{m}\lambda_i\left\|\nabla F_i(\boldsymbol{\mu}^*)-\nabla_{\boldsymbol{\mu}^*}J_i(\boldsymbol{\mu}^*)\right\|^2 \tag{A.59}$$

$$=\sum_{i=1}^{m}\lambda_i\left\|\nabla F_i(\boldsymbol{\mu}^*)-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\sigma})}\nabla F_i(\boldsymbol{x})\right\|^2 \tag{A.60}$$

$$\leq\sum_{i=1}^{m}\lambda_i\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\sigma})}\|\nabla F_i(\boldsymbol{x})-\nabla F_i(\boldsymbol{\mu}^*)\|^2 \tag{A.61}$$

$$\leq L_F^2\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\sigma})}\|\boldsymbol{x}-\boldsymbol{\mu}^*\|^2 \tag{A.62}$$

$$=L_F^2\|\mathrm{diag}(\boldsymbol{\Sigma})\|_1, \tag{A.63}$$

where the equality in Eq. (A.60) is due to $\nabla_{\boldsymbol{\mu}^*}J_i(\boldsymbol{\mu}^*)=\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\sigma})}\nabla F_i(\boldsymbol{x})$ in [98].

A.2.3.3 Proof of Theorem 4.3

We denote $\boldsymbol{q}_t=-\sum_{i=1}^{m}\lambda_i^t\hat{\boldsymbol{g}}_i^t$, then the update rule of $\boldsymbol{\mu}$ can be represented as $\boldsymbol{\mu}_{t+1}=\boldsymbol{\mu}_t+\beta_t\boldsymbol{\Sigma}_t\boldsymbol{q}_t$.

According to assumption 4.1, the function $J_i(\boldsymbol{\theta})$ is L-smooth w.r.t $\{\boldsymbol{\mu},\boldsymbol{\Sigma}\}$, then we have

$$\lambda_i^tJ_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t)\leq\lambda_i^t\left(J_i(\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t)+\beta_t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t,\boldsymbol{\Sigma}_t)^\top\boldsymbol{\Sigma}_t\boldsymbol{q}_t+\frac{L\beta_t^2}{2}\|\boldsymbol{\Sigma}_t\boldsymbol{q}_t\|^2\right). \tag{A.64}$$

Since $F_i(\boldsymbol{x})$ is convex function, we have $J_i(\boldsymbol{\theta})$ is convex w.r.t $\boldsymbol{\theta}=\{\boldsymbol{\mu},\boldsymbol{\Sigma}^{\frac{1}{2}}\}$ by Lemma A.4, together with $J_i(\boldsymbol{\theta})$ is c-strongly convex w.r.t $\boldsymbol{\mu}$ we obtain

$$J_i(\boldsymbol{\theta}_t)\leq J_i(\boldsymbol{\mu}^*,0)+\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*)+\nabla_{\boldsymbol{\Sigma}^{\frac{1}{2}}}J_i(\boldsymbol{\theta}_t)^\top\boldsymbol{\Sigma}_t^{\frac{1}{2}}-\frac{c}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2. \tag{A.65}$$

Note that $\nabla_{\boldsymbol{\Sigma}^{\frac{1}{2}}}J(\boldsymbol{\theta}_t)=\boldsymbol{\Sigma}_t^{\frac{1}{2}}\nabla_{\boldsymbol{\Sigma}}J(\boldsymbol{\theta}_t)+\nabla_{\boldsymbol{\Sigma}}J(\boldsymbol{\theta}_t)\boldsymbol{\Sigma}_t^{\frac{1}{2}}$, we have

$$J_i(\boldsymbol{\theta}_t)\leq J_i(\boldsymbol{\mu}^*,0)+\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*)+2\nabla_{\boldsymbol{\Sigma}}J_i(\boldsymbol{\theta}_t)\boldsymbol{\Sigma}_t-\frac{c}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2. \tag{A.66}$$

Substituting Eq. (A.66) into Eq. (A.64), we have

$$\begin{aligned}\lambda_i^t J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t) \le \lambda_i^t J_i(\boldsymbol{\mu}^*,0) + \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*) + 2\lambda_i^t \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)^\top \boldsymbol{\Sigma}_t \\ + \beta_t \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t,\boldsymbol{\Sigma}_t)^\top \boldsymbol{\Sigma}_t \boldsymbol{q}_t + \frac{L\lambda_i^t \beta_t^2}{2}\|\boldsymbol{\Sigma}_t \boldsymbol{q}_t\|^2 - \frac{c}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2. \end{aligned} \tag{A.67}$$

Let $A_t = \sum_{i=1}^m \lambda_i^t \big(J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t) - J_i(\boldsymbol{\mu}^*,0)\big)$ and $\beta_t \le \frac{1}{L}$, we have

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}}[A_t] \le \mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*)] + \beta_t \mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t,\boldsymbol{\Sigma}_t)^\top \boldsymbol{\Sigma}_t \boldsymbol{q}_t] \\ + \frac{\beta_t}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\Sigma}_t \boldsymbol{q}_t\|^2] + \mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)^\top \boldsymbol{\Sigma}_t] - \frac{c}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2. \end{aligned} \tag{A.68}$$

Note that

$$\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_{t+1}-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} \tag{A.69}$$

$$= \|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_t+\beta_t\boldsymbol{\Sigma}_t\boldsymbol{q}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} \tag{A.70}$$

$$= \|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \big(\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} + \beta_t\langle\boldsymbol{\mu}_t-\boldsymbol{\mu}^*,\boldsymbol{q}_t\rangle + \beta_t^2\langle\boldsymbol{\Sigma}_t\boldsymbol{q}_t,\boldsymbol{q}_t\rangle\big) \tag{A.71}$$

$$= -2\beta_t\boldsymbol{q}_t^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*) - \beta_t^2(\boldsymbol{\Sigma}_t\boldsymbol{q}_t)^\top\boldsymbol{q}_t. \tag{A.72}$$

Therefore we have

$$-\boldsymbol{q}_t^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*) = \frac{1}{2\beta_t}\big(\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_{t+1}-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}\big) + \frac{\beta_t}{2}(\boldsymbol{\Sigma}_t\boldsymbol{q}_t)^\top\boldsymbol{q}_t \tag{A.73}$$

$$\le \frac{1}{2\beta_t}\big(\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_{t+1}-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}\big) + \beta_t\|\boldsymbol{\Sigma}_t\|_F\|\boldsymbol{q}_t\|^2, \tag{A.74}$$

where the inequality is due to $\beta_t \ge 0$ and Lemma A.3. Note that we have

$$\mathbb{E}_{\boldsymbol{z}}\big[(\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t) + \boldsymbol{q}_t)^\top(\boldsymbol{\mu}_t-\boldsymbol{\mu}^*)\big] \le \|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|\sqrt{\mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t) + \boldsymbol{q}_t\|^2]} \tag{A.75}$$

$$\le D\sqrt{\mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^m \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t) + \boldsymbol{q}_t\|^2]} \tag{A.76}$$

$$\le D\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^m \mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]}, \tag{A.77}$$

where the first inequality is due to the Cauchy-Schwarz inequality, the second inequality

is due to $\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\| \leq D$ and the last inequality is due to Lemma A.8 (a). Then we have

(A.78)
$$\mathbb{E}_{\boldsymbol{z}}\big[\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)^\top (\boldsymbol{\mu}_t - \boldsymbol{\mu}^*)\big]$$

(A.79)
$$= \mathbb{E}_{\boldsymbol{z}}\big[-\boldsymbol{q}_t^\top (\boldsymbol{\mu}_t - \boldsymbol{\mu}^*) + (\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t) + \boldsymbol{q}_t)^\top (\boldsymbol{\mu}_t - \boldsymbol{\mu}^*)\big]$$

(A.80)
$$\leq \frac{1}{2\beta_t}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}] + \beta_t \|\boldsymbol{\Sigma}_t\|_F \mathbb{E}_{\boldsymbol{z}}\|\boldsymbol{q}_t\|^2] + D\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}(g_i^t)},$$

where the inequality is due to Eq. (A.74) and Eq. (A.77). Note that

(A.81)
$$\mathbb{E}_{\boldsymbol{z}}\big[\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)^\top \boldsymbol{\Sigma}_t\big] \leq \mathbb{E}_{\boldsymbol{z}}\big[\|\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\Sigma}} J_i(\boldsymbol{\theta}_t)\|\big]\|\boldsymbol{\Sigma}_t\|_F \leq H\|\boldsymbol{\Sigma}_t\|_F,$$

where the first inequality is due to Lemma A.3 and the second inequality is due to the Lipschitz continuous assumption of the function $J_i(\boldsymbol{\theta})$. By using Lemma A.3 and Lemma A.8 (b), we further have

(A.82)
$$\mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t, \boldsymbol{\Sigma}_t)^\top \boldsymbol{\Sigma}_t \boldsymbol{q}_t] \leq \|\boldsymbol{\Sigma}_t\|_F \left(2H\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} - \mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2]\right).$$

Then substituting Eqs. (A.80) (A.81) (A.82) into Eq. (A.68) and multiplying β_t on both sides of the inequality, we have

$$\begin{aligned}
\beta_t \mathbb{E}_{\boldsymbol{z}}[A_t] \leq{}& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}] - \frac{c\beta_t}{2}\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\|^2 + \beta_t^2 H\|\boldsymbol{\Sigma}_t\|_F \\
&+ (2H\beta_t^2\|\boldsymbol{\Sigma}_t\|_F + \beta_t D)\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} + \frac{\beta_t^2}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\Sigma}_t \boldsymbol{q}_t\|^2] \\
&+ \beta_t^2\|\boldsymbol{\Sigma}_t\|_F \mathbb{E}_{\boldsymbol{z}}\|\boldsymbol{q}_t\|^2] - \beta_t^2\|\boldsymbol{\Sigma}_t\|_F \mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^{m} \lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2] \qquad \text{(A.83)} \\
\leq{}& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}} - \|\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}] - \frac{c\beta_t}{2}\|\boldsymbol{\mu}_t - \boldsymbol{\mu}^*\|^2 + \beta_t^2 H\|\boldsymbol{\Sigma}_t\|_F \\
&+ (2H\beta_t^2\|\boldsymbol{\Sigma}_t\|_F + \beta_t D)\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} + \frac{\beta_t^2 H^2(d+4)^2}{8\xi^2(\sum_{k=1}^{t}\beta_k)^2} \\
&+ \beta_t^2\|\boldsymbol{\Sigma}_t\|_F \left(\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t] + 4H\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]}\right), \qquad \text{(A.84)}
\end{aligned}$$

where the second inequality is due to Lemma A.6 (b) and Lemma A.8 (c). We further obtain that

$$\sum_{t=0}^{T-1}\left[\frac{1}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}-\|\boldsymbol{\mu}_{t+1}-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}]-\frac{c\beta_t}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2\right] \tag{A.85}$$

$$\leq \frac{1}{2}\sum_{t=0}^{T-1}\left[\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_t^{-1}}-\|\boldsymbol{\mu}_{t-1}-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_{t-1}^{-1}}-\frac{c\beta_t}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2\right]$$

$$+\frac{1}{2}\left[\|\boldsymbol{\mu}_0-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_0^{-1}}-\|\boldsymbol{\mu}_T-\boldsymbol{\mu}^*\|^2_{\boldsymbol{\Sigma}_{T-1}^{-1}}\right] \tag{A.86}$$

$$\leq \frac{1}{2}\sum_{t=0}^{T-1}\left[\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2_{2\beta_t\sum_{i=1}^m\lambda_i^t\hat{G}_i^t}-\frac{c\beta_t}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2\right]+\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2 \tag{A.87}$$

$$\leq \frac{1}{2}\sum_{t=0}^{T-1}\left[\frac{c\beta_t}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2-\frac{c\beta_t}{2}\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|^2\right]+\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2 \tag{A.88}$$

$$=\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2, \tag{A.89}$$

where the second inequality is due to the update rule of $\boldsymbol{\Sigma}_t$ and $\|\boldsymbol{\mu}_t-\boldsymbol{\mu}^*\|\leq D$, and the third inequality is due to Cauchy-Schwarz inequality and $\hat{G}_i \preceq \frac{c}{4}\boldsymbol{I}$. Let $C=\max(\frac{c}{4\xi},\|\boldsymbol{\Sigma}_0^{-1}\|_\infty)$, we have

$$\sum_{t=0}^{T-1}\beta_t\mathbb{E}_{\boldsymbol{z}}[A_t]\leq\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2+\sum_{t=0}^{T-1}\left(\frac{\beta_t^2 H\sqrt{d}}{2\xi\sum_{k=1}^t\beta_k}+\frac{\beta_t^2H^2(d+4)^2}{8\xi^2(\sum_{k=1}^t\beta_k)^2}\right.$$

$$\left.+(6H\beta_t^2\|\boldsymbol{\Sigma}_t\|_F+\beta_t D)\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^m\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]}+\beta_t^2\|\boldsymbol{\Sigma}_t\|_F\sum_{i=1}^m\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]\right) \tag{A.90}$$

$$\leq\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2+\sum_{t=0}^{T-1}\left(\frac{\beta_t^2H\sqrt{d}}{2\xi\sum_{k=1}^t\beta_k}+\frac{\beta_t^2H^2(d+4)^2}{8\xi^2(\sum_{k=1}^t\beta_k)^2}\right.$$

$$\left.+(6H\beta_t^2\sqrt{d}R+\beta_t D)\gamma_t\sqrt{2\sum_{i=1}^m\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]}+\frac{\beta_t^2\sqrt{d}\sum_{i=1}^m\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]}{2\xi\sum_{k=1}^t\beta_k}\right) \tag{A.91}$$

$$\leq\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2+\sum_{t=0}^{T-1}\left(\frac{\beta_t^2H\sqrt{d}}{2\xi\sum_{k=1}^t\beta_k}+\frac{\beta_t^2H^2(d+4)^2}{8\xi^2(\sum_{k=1}^t\beta_k)^2}\right.$$

$$\left.+H\gamma_t\beta_t(d+4)(6H\beta_t\sqrt{d}R+D)\sqrt{\frac{2Cm}{N}}+\frac{\beta_t^2\sqrt{d}H^2C(d+4)^2m}{2N\xi\sum_{k=1}^t\beta_k}\right), \tag{A.92}$$

where the first inequality is due to Eq. (A.89) and Lemma A.5 (a), the second inequality is due to Lemma A.5 (b) and Lemma A.7, and the third inequality is due to Lemma A.6 (c).

Therefore, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}[A_t] \leq \frac{\|\boldsymbol{\Sigma}_0^{-1}\|_F D^2}{T\beta_t} + \frac{1}{T}\sum_{t=0}^{T-1}\Bigg(\frac{\beta_t H\sqrt{d}}{2\xi\sum_{k=1}^{t}\beta_k} + \frac{\beta_t H^2(d+4)^2}{8\xi^2(\sum_{k=1}^{t}\beta_k)^2}$$

$$+ H\gamma_t(d+4)(6H\beta_t\sqrt{d}R + D)\sqrt{\frac{2Cm}{N}} + \frac{\beta_t\sqrt{d}H^2C(d+4)^2m}{2N\xi\sum_{k=1}^{t}\beta_k}\Bigg). \tag{A.93}$$

Let $\beta_t = \beta$ and $\gamma_t = \gamma$. Since we have $\sum_{t=1}^{T}\frac{1}{t} \leq 1+\log(T)$, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}\left[\sum_{i=1}^{m}\lambda_i^t(J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t) - J_i(\boldsymbol{\mu}^*,0))\right] = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}[A_t] = \mathcal{O}\left(\frac{1}{\beta T} + \frac{\log T}{T} + \gamma\right), \tag{A.94}$$

where we reach the conclusion.

A.2.3.4 Proof of Theorem 4.4

We denote $\boldsymbol{q}_t = -\sum_{i=1}^{m}\lambda_i^t\hat{\boldsymbol{g}}_i^t$, then the update rule of $\boldsymbol{\mu}$ can be represented as $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \beta_t\boldsymbol{\Sigma}_t\boldsymbol{q}_t$.

According to assumption 4.1, the function $J_i(\boldsymbol{\theta})$ is L-smooth w.r.t $\{\boldsymbol{\mu},\boldsymbol{\Sigma}\}$, then we have

$$\lambda_i^t J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t) \leq \lambda_i^t\left(J_i(\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t) + \beta_t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t,\boldsymbol{\Sigma}_t)^\top\boldsymbol{\Sigma}_t\boldsymbol{q}_t + \frac{L\beta_t^2}{2}\|\boldsymbol{\Sigma}_t\boldsymbol{q}_t\|^2\right). \tag{A.95}$$

Let $B_t = \mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^{m}\lambda_i^t(J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t) - J_i(\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t))]$, then we have

$$B_t \leq \beta_t\|\boldsymbol{\Sigma}_t\|_F\mathbb{E}_{\boldsymbol{z}}[(\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t))^\top\boldsymbol{q}_t] + \frac{L\beta_t^2\|\boldsymbol{\Sigma}_t\|_F^2}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{q}_t\|^2] \tag{A.96}$$

$$\leq 2H\beta_t\sqrt{d}R\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} - \beta_t\sqrt{d}R\mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\|^2]$$

$$+ \frac{L\beta_t^2 dR^2}{2}\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{q}_t\|^2] \tag{A.97}$$

$$\leq (2H\beta_t\sqrt{d}R + 2HL\beta_t^2 dR^2)\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} + \frac{L\beta_t^2 dR^2}{2}\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]$$

$$+ \frac{L\beta_t^2 dR^2 - 2\beta_t\sqrt{d}R}{2}\mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\|^2], \tag{A.98}$$

where the first inequality is due to Lemma A.3, the second inequality is due to $\|\boldsymbol{\Sigma}_t\|_F \leq \|\boldsymbol{\Sigma}_0\|_F \leq \sqrt{d}R$ and Lemma A.8 (b), and the last inequality is due to Lemma A.8 (c). Let $\beta_t \leq \frac{1}{LR\sqrt{d}}$, and rearrange Eq. (A.98) we obtain

$$\frac{\beta_t\sqrt{d}R}{2}\mathbb{E}_{\boldsymbol{z}}\big[\|\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\|^2\big] \leq B_t + 4HR\sqrt{d}\beta_t\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} + \frac{L\beta_t^2 dR^2}{2}\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]. \tag{A.99}$$

So we have

(A.100)
$$\beta_t \mathbb{E}_{\boldsymbol{z}}\big[\|\sum_{i=1}^{m}\lambda_i^t \nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)\|^2\big] \le \frac{2B_t}{\sqrt{d}R} + 8H\beta_t\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} + \frac{L\beta_t^2\sqrt{d}R}{2}\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t].$$

Note that we have

(A.101)
$$\sum_{t=0}^{T-1}\sum_{i=1}^{m}\lambda_i^t(J_i(\boldsymbol{\theta}_{t+1}) - J_i(\boldsymbol{\theta}_t)) = \sum_{t=0}^{T-1}\sum_{i=1}^{m}(\lambda_i^t - \lambda_i^{t+1})J_i(\boldsymbol{\theta}_{t+1}) + \sum_{i=1}^{m}(\lambda_i^{T-1}J_i(\boldsymbol{\theta}_T) - \lambda_i^0 J_i(\boldsymbol{\theta}_0))$$

(A.102)
$$\le \sum_{t=0}^{T-1}\sum_{i=1}^{m}|\lambda_i^t - (1-\gamma_t)\lambda_i^t - \gamma_t\tilde{\lambda}_i^{t+1}|B + 2B$$

(A.103)
$$\le \sum_{t=0}^{T-1}\gamma_t\sum_{i=1}^{m}|\lambda_i^t - \tilde{\lambda}_i^{t+1}|B + 2B,$$

where the first inequality is due to the update rule of $\boldsymbol{\lambda}^t$ and $|J_i(\boldsymbol{\theta})| \le B$. Then we have

(A.104)
$$\sum_{t=0}^{T-1}B_t = \sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^{m}\lambda_i^t(J_i(\boldsymbol{\theta}_{t+1}) - J_i(\boldsymbol{\theta}_t))] + \sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^{m}\lambda_i^t(J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_t) - J_i(\boldsymbol{\mu}_{t+1},\boldsymbol{\Sigma}_{t+1}))]$$

(A.105)
$$\le \sum_{t=0}^{T-1}\gamma_t\sum_{i=1}^{m}|\lambda_i^t - \tilde{\lambda}_i^{t+1}|B + 2B + \sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}[\sum_{i=1}^{m}\lambda_i^t H\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F]$$

(A.106)
$$\le 2mB\sum_{t=0}^{T-1}\gamma_t + 2B + \sum_{t=0}^{T-1}H\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F,$$

where the first inequality is due to Eq. (A.103) and the Lipschitz continuous assumption of the function $J_i(\boldsymbol{\theta})$. Substituting Eq. (A.106) into Eq. (A.100), we have

(A.107)
$$\frac{1}{T}\sum_{t=0}^{T-1}\beta_t\mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\|^2] \le \frac{2mB\sum_{t=0}^{T-1}\gamma_t + 2B + \sum_{t=0}^{T-1}H\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F}{\sqrt{d}RT} + \frac{1}{T}\sum_{t=0}^{T-1}\left(8H\beta_t\sqrt{\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]} + \frac{L\beta_t^2\sqrt{d}R}{2}\sum_{i=1}^{m}\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t]\right).$$

According to Lemma A.5 (c), A.6 (c), and A.7. We know that $\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F \le \frac{b\beta_t d^{\frac{3}{2}}}{2\xi^2(\sum_{k=1}^{t}\beta_k)^2}$, $\mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t] \le \frac{H^2C(d+4)^2}{N}$, where $C = \max(\frac{b}{\xi}, \|\boldsymbol{\Sigma}_0^{-1}\|_\infty)$, and $\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] \le 2\gamma_t^2$. Then we have

(A.108)
$$\frac{1}{T}\sum_{t=0}^{T-1}\beta_t\mathbb{E}_{\boldsymbol{z}}\big[\|\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\|^2\big] \le \frac{2mB\sum_{t=0}^{T-1}\gamma_t + 2B + \sum_{t=0}^{T-1}H\frac{b\beta_t d^{\frac{3}{2}}}{2\xi^2(\sum_{k=1}^{t}\beta_k)^2}}{\sqrt{d}RT} + \frac{1}{T}\sum_{t=0}^{T-1}\left(8H^2C(d+4)\gamma_t\beta_t\sqrt{\frac{2m}{N}} + \frac{\beta_t^2H^2C(d+4)^2L\sqrt{d}Rm}{2N}\right).$$

Let $\beta_t = \beta$ and $\gamma_t = \gamma$, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\boldsymbol{z}}[\|\sum_{i=1}^{m}\lambda_i^t\nabla_{\boldsymbol{\mu}}J_i(\boldsymbol{\theta}_t)\|^2] = \mathcal{O}\left(\frac{\gamma}{\beta}+\frac{1}{\beta T}+\gamma+\beta\right), \tag{A.109}$$

where we reach the conclusion.

A.2.4 Proof of Technical Lemmas

In this section, we provide the proof of lemmas in Appendix A.2.2.

A.2.4.1 Proof of Lemma A.3

Since $\boldsymbol{\Sigma}$ and $\hat{\boldsymbol{\Sigma}}$ are both diagonal matrix. Denote $\boldsymbol{\sigma} = \text{diag}(\boldsymbol{\Sigma})$ and $\hat{\boldsymbol{\sigma}} = \text{diag}(\hat{\boldsymbol{\Sigma}})$. Then we have

$$\|\boldsymbol{\Sigma}\boldsymbol{z}\|^2 = \sum_{i=1}^{d}(\sigma_i z_i)^2 \le \sum_{i=1}^{d}(\sigma_i)^2\sum_{i=1}^{d}(z_i)^2 = \|\boldsymbol{\sigma}\|^2\|\boldsymbol{z}\|^2 = \|\boldsymbol{\Sigma}\|_F^2\|\boldsymbol{z}\|^2. \tag{A.110}$$

We further have

$$\|\boldsymbol{\Sigma}\hat{\boldsymbol{\Sigma}}\|_F^2 = \sum_{i=1}^{d}(\sigma_i\hat{\sigma}_i)^2 \le \sum_{i=1}^{d}(\sigma_i)^2\sum_{i=1}^{d}(\hat{\sigma}_i)^2 = \|\boldsymbol{\sigma}\|^2\|\hat{\boldsymbol{\sigma}}\|^2 = \|\boldsymbol{\Sigma}\|_F^2\|\hat{\boldsymbol{\Sigma}}\|_F^2. \tag{A.111}$$

Then we reach the conclusion.

A.2.4.2 Proof of Lemma A.4

For $\lambda \in [0,1]$, we have

$$\lambda\bar{J}(\boldsymbol{\theta}_1)+(1-\lambda)\bar{J}(\boldsymbol{\theta}_2) = \lambda\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}[f(\boldsymbol{\mu}_1+\boldsymbol{\Sigma}_1^{\frac{1}{2}}\boldsymbol{z})]+(1-\lambda)\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}[f(\boldsymbol{\mu}_2+\boldsymbol{\Sigma}_2^{\frac{1}{2}}\boldsymbol{z})] \tag{A.112}$$

$$= \mathbb{E}[\lambda f(\boldsymbol{\mu}_1+\boldsymbol{\Sigma}_1^{\frac{1}{2}}\boldsymbol{z})+(1-\lambda)f(\boldsymbol{\mu}_2+\boldsymbol{\Sigma}_2^{\frac{1}{2}}\boldsymbol{z})] \tag{A.113}$$

$$\ge \mathbb{E}[f\left(\lambda\boldsymbol{\mu}_1+(1-\lambda)\boldsymbol{\mu}_2+(\lambda\boldsymbol{\Sigma}_1^{\frac{1}{2}}+(1-\lambda)\boldsymbol{\Sigma}_2^{\frac{1}{2}})\boldsymbol{z}\right)] \tag{A.114}$$

$$= \bar{J}(\lambda\boldsymbol{\theta}_1+(1-\lambda)\boldsymbol{\theta}_2), \tag{A.115}$$

where we reach the conclusion.

A.2.4.3 Proof of Lemma A.5

(a): Since we have $\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + 2\beta_t\sum_{i=1}^{m}\lambda_i^t\hat{G}_i^t$ and $\boldsymbol{\lambda}^t \in \Delta^{m-1}$. We obtain

$$\boldsymbol{\Sigma}_t^{-1} + 2b\beta_t\boldsymbol{I} \succeq \boldsymbol{\Sigma}_{t+1}^{-1} \succeq \boldsymbol{\Sigma}_t^{-1} + 2\xi\beta_t\boldsymbol{I}. \tag{A.116}$$

Summing up it over $t = 0, \ldots, T-1$, we have

$$\boldsymbol{\Sigma}_0^{-1} + 2b\sum_{t=1}^{T}\beta_t \boldsymbol{I} \succeq \boldsymbol{\Sigma}_T^{-1} \succeq \boldsymbol{\Sigma}_0^{-1} + 2\xi\sum_{t=1}^{T}\beta_t \boldsymbol{I}. \tag{A.117}$$

Therefore, we have

$$\frac{1}{2b\sum_{t=1}^{T}\beta_t \boldsymbol{I} + \boldsymbol{\Sigma}_0^{-1}} \preceq \boldsymbol{\Sigma}_T \preceq \frac{1}{2\xi\sum_{t=1}^{T}\beta_t \boldsymbol{I} + \boldsymbol{\Sigma}_0^{-1}}. \tag{A.118}$$

(b): We have

$$\|\boldsymbol{\Sigma}_t\|_F \leq \left\| \frac{1}{2\xi\sum_{t=1}^{T}\beta_t \boldsymbol{I} + \boldsymbol{\Sigma}_0^{-1}} \right\|_F \leq \left\| \frac{1}{2\xi\sum_{t=1}^{T}\beta_t \boldsymbol{I}} \right\|_F = \frac{\sqrt{d}}{2\xi\sum_{t=1}^{T}\beta_t}. \tag{A.119}$$

(c): We have

$$\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F = \|\frac{1}{\boldsymbol{\Sigma}_t^{-1} + 2\beta_t\sum_{i=1}^{m}\lambda_i^t\hat{G}_i^t} - \boldsymbol{\Sigma}_t\|_F \leq \|\frac{-2\beta_t\boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_t\sum_{i=1}^{m}\lambda_i^t\hat{G}_i^t}{\boldsymbol{I} + 2\beta_t\boldsymbol{\Sigma}_t\sum_{i=1}^{m}\lambda_i^t\hat{G}_i^t}\|_F \tag{A.120}$$

$$\leq 2\beta_t\|\boldsymbol{\Sigma}_t\|_F^2\|\sum_{i=1}^{m}\lambda_i^t\hat{G}_i^t\|_F. \tag{A.121}$$

Since $\|\boldsymbol{\Sigma}_t\| \leq \frac{\sqrt{d}}{2\xi\sum_{t=1}^{T}\beta_t}$ and $\|\sum_{i=1}^{m}\lambda_i^t\hat{G}_i^t\|_F \leq b\sqrt{d}$. Then we have

$$\|\boldsymbol{\Sigma}_{t+1} - \boldsymbol{\Sigma}_t\|_F \leq \frac{b\beta_t d^{\frac{3}{2}}}{2\xi^2(\sum_{t=1}^{T}\beta_t)^2}. \tag{A.122}$$

A.2.4.4 Proof of Lemma A.6

(a). We first show that $\hat{\boldsymbol{g}}_i^t$ is a unbiased estimator of $\nabla_{\boldsymbol{\mu}}\mathbb{E}_{p_{\boldsymbol{\theta}_t}}[F_i(\boldsymbol{x})]$.

$$\mathbb{E}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_i^t] = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\Sigma}_t^{-\frac{1}{2}}\boldsymbol{z}F_i(\boldsymbol{\mu}_t + \boldsymbol{\Sigma}_t^{\frac{1}{2}}\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\Sigma}_t^{-\frac{1}{2}}\boldsymbol{z}F_i(\boldsymbol{\mu}_t)] \tag{A.123}$$

$$= \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\Sigma}_t^{-\frac{1}{2}}\boldsymbol{z}F_i(\boldsymbol{\mu}_t + \boldsymbol{\Sigma}_t^{\frac{1}{2}}\boldsymbol{z})] \tag{A.124}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t)}[\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_t)F_i(\boldsymbol{x})] \tag{A.125}$$

$$= \nabla_{\boldsymbol{\mu}}\mathbb{E}_{p_{\boldsymbol{\theta}_t}}[F_i(\boldsymbol{x})]. \tag{A.126}$$

(b). Since the $\boldsymbol{\sigma}$ is the diagonal elements of $\boldsymbol{\Sigma}$, then we have

$$\|\boldsymbol{\Sigma}_t\hat{\boldsymbol{g}}_{ij}^t\|^2 = \|\boldsymbol{\sigma}_t \odot \boldsymbol{\sigma}_t^{-\frac{1}{2}} \odot \boldsymbol{z}_j(F_i(\boldsymbol{\mu}_t + \boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j) - F_i(\boldsymbol{\mu}_t))\|^2 \tag{A.127}$$

$$= \|\boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j\|^2(F_i(\boldsymbol{\mu}_t + \boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j) - F_i(\boldsymbol{\mu}_t))^2 \tag{A.128}$$

$$\leq \|\boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j\|^2 H^2\|\boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j\|^2 \tag{A.129}$$

$$\leq H^2\left\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\right\|_\infty^2 \times \left\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\right\|_\infty^2 \times \left\|\boldsymbol{z}_j\right\|^4 = H^2\left\|\boldsymbol{\sigma}_t\right\|_\infty^2\left\|\boldsymbol{z}_j\right\|^4. \tag{A.130}$$

It follows that

$$\|\boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t\|_2^2 = \left\| \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_{ij}^t \right\|^2 \leq \frac{1}{N} \sum_{j=1}^{N} \|\boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_{ij}^t\|^2 \leq H^2 \left\|\boldsymbol{\sigma}_t\right\|_\infty^2 \left\|\boldsymbol{z}_j\right\|^4. \tag{A.131}$$

Noticed that $\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{z}\|^4] \leq (d+4)^2$ and

$$\|\boldsymbol{\sigma}_t\|_\infty \leq \frac{1}{\|\boldsymbol{\sigma}_0^{-1}\|_{min} + 2(\sum_{k=1}^{t} \beta_k)\xi}, \tag{A.132}$$

where $\|\cdot\|_{min}$ denotes the minimum element in the input. Then we have

$$\mathbb{E}\|\boldsymbol{\Sigma}_t \hat{\boldsymbol{g}}_i^t\|_2^2 \leq H^2 \left\|\boldsymbol{\sigma}_t\right\|_\infty^2 (d+4)^2 \leq \frac{H^2(d+4)^2}{4\xi^2(\sum_{k=1}^{t} \beta_k)^2}, \tag{A.133}$$

where we reach the conclusion.

(c): We have

$$\|\hat{\boldsymbol{g}}_{ij}^t\|^2 = \|\boldsymbol{\sigma}_t^{-\frac{1}{2}} \odot \boldsymbol{z}_j (F_i(\boldsymbol{\mu}_t + \boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j) - F_i(\boldsymbol{\mu}_t))\|^2 \tag{A.134}$$

$$= \|\boldsymbol{\sigma}_t^{-\frac{1}{2}} \odot \boldsymbol{z}_j\|^2 (F_i(\boldsymbol{\mu}_t + \boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j) - F_i(\boldsymbol{\mu}_t))^2 \tag{A.135}$$

$$\leq \|\boldsymbol{\sigma}_t^{-\frac{1}{2}} \odot \boldsymbol{z}_j\|^2 H^2 \|\boldsymbol{\sigma}_t^{\frac{1}{2}} \odot \boldsymbol{z}_j\|^2 \tag{A.136}$$

$$\leq H^2 \left\|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\right\|_\infty^2 \times \left\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\right\|_\infty^2 \times \left\|\boldsymbol{z}_j\right\|^4. \tag{A.137}$$

Then we obtain

$$\mathbb{E}_{\boldsymbol{z}}[\|\hat{\boldsymbol{g}}_{ij}^t\|^2] \leq H^2 \left\|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\right\|_\infty^2 \times \left\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\right\|_\infty^2 \times \mathbb{E}[\left\|\boldsymbol{z}_j\right\|^4] \tag{A.138}$$

$$\leq H^2 \left\|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\right\|_\infty^2 \times \left\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\right\|_\infty^2 (d+4)^2. \tag{A.139}$$

Note that for N i.i.d samples $\boldsymbol{z}_j$, we have

$$\mathbb{V}_{\boldsymbol{z}}[\frac{1}{N} \sum_{j=1}^{N} \hat{\boldsymbol{g}}_{ij}^t] = \frac{1}{N} \mathbb{V}_{\boldsymbol{z}}[\hat{\boldsymbol{g}}_{ij}^t] \leq \frac{1}{N} \mathbb{E}_{\boldsymbol{z}}[\|\hat{\boldsymbol{g}}_{ij}^t\|_2^2] \tag{A.140}$$

$$\leq \frac{H^2(d+4)^2 \|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\|_\infty^2 \|\boldsymbol{\sigma}_t^{\frac{1}{2}}\|_\infty^2}{N}. \tag{A.141}$$

Note that we have

$$\boldsymbol{\sigma}_0^{-1} + 2b \sum_{k=1}^{t} \beta_t \mathbf{1} \geq \boldsymbol{\sigma}_t^{-1} \geq \boldsymbol{\sigma}_0^{-1} + 2\xi \sum_{k=1}^{t} \beta_t \mathbf{1}. \tag{A.142}$$

Then, we have

$$\|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\|_\infty^2 = \|\boldsymbol{\sigma}_t^{-1}\|_\infty \leq \|\boldsymbol{\sigma}_0^{-1}\|_\infty + 2(\sum_{k=1}^{t} \beta_k) b. \tag{A.143}$$

And

$$\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\|_\infty^2 = \|\boldsymbol{\sigma}_t\|_\infty \leq \frac{1}{\|\boldsymbol{\sigma}_0^{-1}\|_{min} + 2(\sum_{k=1}^t \beta_k)\xi}, \tag{A.144}$$

where $\|\cdot\|_{min}$ denotes the minimum element in the input.

we then have

$$\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\|_\infty^2 \|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\|_\infty^2 \leq \frac{\|\boldsymbol{\sigma}_0^{-1}\|_\infty + 2(\sum_{k=1}^t \beta_k)b}{\|\boldsymbol{\sigma}_0^{-1}\|_{min} + 2(\sum_{k=1}^t \beta_k)\xi} \tag{A.145}$$

$$= \frac{b}{\xi} + \frac{\|\boldsymbol{\sigma}_0^{-1}\|_\infty - \frac{b}{\xi}\|\boldsymbol{\sigma}_0^{-1}\|_{min}}{\|\boldsymbol{\sigma}_0^{-1}\|_{min} + 2(\sum_{k=1}^t \beta_k)\xi}. \tag{A.146}$$

If $\|\boldsymbol{\sigma}_0^{-1}\|_\infty - \frac{b}{\xi}\|\boldsymbol{\sigma}_0^{-1}\|_{min} \geq 0$, we have

$$\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\|_\infty^2 \|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\|_\infty^2 \leq \frac{b}{\xi} + \frac{\|\boldsymbol{\sigma}_0^{-1}\|_\infty - \frac{b}{\xi}\|\boldsymbol{\sigma}_0^{-1}\|_{min}}{\|\boldsymbol{\sigma}_0^{-1}\|_{min}} \leq \|\boldsymbol{\sigma}_0^{-1}\|_\infty. \tag{A.147}$$

If $\|\boldsymbol{\sigma}_0^{-1}\|_\infty - \frac{b}{\xi}\|\boldsymbol{\sigma}_0^{-1}\|_{min} < 0$, we have

$$\|\boldsymbol{\sigma}_t^{\frac{1}{2}}\|_\infty^2 \|\boldsymbol{\sigma}_t^{-\frac{1}{2}}\|_\infty^2 \leq \frac{b}{\xi}. \tag{A.148}$$

Therefore, let $C = \max(\frac{b}{\xi}, \|\boldsymbol{\sigma}_0^{-1}\|_\infty)$, we have

$$\mathbb{V}_{\boldsymbol{z}}[\|\frac{1}{N}\sum_{j=1}^N \hat{\boldsymbol{g}}_{ij}^t\|^2] \leq \frac{H^2(d+4)^2 C}{N}, \tag{A.149}$$

where we reach the conclusion.

A.2.4.5 Proof of Lemma A.7

We have

$$\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t] = \mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\lambda}^t - \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{\lambda}^t]\|^2]] \leq \mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-1}\|^2] \tag{A.150}$$

$$= \mathbb{E}_{\boldsymbol{z}}[\|\gamma_t(\tilde{\boldsymbol{\lambda}}^t - \boldsymbol{\lambda}^{t-1})\|^2] \leq \gamma_t^2 \mathbb{E}_{\boldsymbol{z}}[\|\tilde{\boldsymbol{\lambda}}^t - \boldsymbol{\lambda}^{t-1}\|^2] \leq 2\gamma_t^2, \tag{A.151}$$

where we reach the conclusion.

A.2.4.6 Proof of Lemma A.8

According to Lemma A.6 (a) and (c), we know that $\hat{\boldsymbol{g}}_i^t$ is an unbiased estimator of the gradient $\nabla_{\boldsymbol{\mu}} J_i(\boldsymbol{\theta}_t)$ and the variance of $\hat{\boldsymbol{g}}_i^t$ is bounded. Therefore let $\boldsymbol{q}_t = -\sum_{i=1}^m \lambda_i^t \hat{\boldsymbol{g}}_i^t$, the results in Lemma A.8 can be directly obtained by Lemma 1, 7, and 8 in [149].

A.2.5 Updated Rule Under Transformation

To avoid the scaling problem, we can employ monotonic transformation for the aggregated objective, i.e. $h(\boldsymbol{\lambda}^\top F(\boldsymbol{x}_j)) = \frac{\boldsymbol{\lambda}^\top F(\boldsymbol{x}_j) - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}}$, where $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ denote mean and stand deviation of aggregated function values $\boldsymbol{\lambda}^\top F(\boldsymbol{x}_j) = \sum_{i=1}^m \lambda_i F_i(\boldsymbol{x}_j)$, $j = 1, \ldots, N$. Then by applying this rescaling strategy, the update rule for $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ in t-th iteration can be written as

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \frac{\beta_t}{N} \sum_{j=1}^{N} (\boldsymbol{x}_j - \boldsymbol{\mu}_t) \frac{\sum_{i=1}^m \lambda_i^t F_i(\boldsymbol{x}_j) - \hat{\boldsymbol{\mu}}^t}{\hat{\boldsymbol{\sigma}}^t}, \tag{A.152}$$

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + \frac{\beta_t}{N} \sum_{j=1}^{N} \operatorname{diag}\Big[\boldsymbol{\Sigma}_t^{-1}\Big[\operatorname{diag}\big((\boldsymbol{x}_j - \boldsymbol{\mu}_t)(\boldsymbol{x}_j - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1}\big) \frac{\sum_{i=1}^m \lambda_i^t F_i(\boldsymbol{x}_j) - \hat{\boldsymbol{\mu}}^t}{\hat{\boldsymbol{\sigma}}^t}\Big]\Big]. \tag{A.153}$$

BIBLIOGRAPHY

[1] D. ANGUS, *Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem*, in IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, 2007.

[2] A. ARRIETA, S. WANG, A. ARRUABARRENA, U. MARKIEGI, G. SAGARDUI, AND L. ETXEBERRIA, *Multi-objective black-box test case selection for cost-effectively testing simulation models*, in Proceedings of the genetic and evolutionary computation conference, 2018, pp. 1411–1418.

[3] T. BACK, *A survey of evolution strategies*, in Proc. of Fourth Internal. Conf. on Genetic Algorithms, 1991.

[4] F. BAO, G. WU, C. LI, J. ZHU, AND B. ZHANG, *Stability and generalization of bilevel programming in hyperparameter optimization*, in Neural Information Processing Systems, 2021.

[5] R. G. BARTLE AND D. R. SHERBERT, *Introduction to real analysis*, vol. 2, Wiley New York, 2000.

[6] P. L. BARTLETT AND S. MENDELSON, *Rademacher and gaussian complexities: Risk bounds and structural results*, Journal of Machine Learning Research, 3 (2002), pp. 463–482.

[7] J. M. BORWEIN, *A very complicated proof of the minimax theorem*, Minimax Theory and its Applications, 1 (2016), pp. 21–27.

[8] T. BROWN, B. MANN, N. RYDER, M. SUBBIAH, J. D. KAPLAN, P. DHARIWAL, A. NEELAKANTAN, P. SHYAM, G. SASTRY, A. ASKELL, ET AL., *Language models are few-shot learners*, Advances in neural information processing systems, 33 (2020), pp. 1877–1901.

[9] D. BRUGGEMANN, M. KANAKIS, S. GEORGOULIS, AND L. VAN GOOL, *Automated search for resource-efficient branched multi-task networks*, arXiv preprint arXiv:2008.10292, (2020).

[10] R. CARUANA, *Multitask learning*, Machine learning, 28 (1997), pp. 41–75.

[11] G. CHEN, X. HAN, G. LIU, C. JIANG, AND Z. ZHAO, *An efficient multi-objective optimization method for black-box functions using sequential approximate technique*, Applied Soft Computing, 12 (2012), pp. 14–27.

[12] L. CHEN, H.-L. LIU, K. LI, AND K. C. TAN, *Evolutionary bi-level optimization via multi-objective transformation-based lower level search*, IEEE Transactions on Evolutionary Computation, (2023).

[13] L. CHEN, J. XU, AND J. ZHANG, *On bilevel optimization without lower-level strong convexity*, arXiv preprint arXiv:2301.00712, (2023).

[14] L. CHEN, Y. ZHU, G. PAPANDREOU, F. SCHROFF, AND H. ADAM, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, in Proceedings of the 14th European Conference on Computer Vision, vol. 11211, 2018, pp. 833–851.

[15] R. T. CHEN, Y. RUBANOVA, J. BETTENCOURT, AND D. K. DUVENAUD, *Neural ordinary differential equations*, Advances in neural information processing systems, 31 (2018).

[16] Z. CHEN, V. BADRINARAYANAN, C.-Y. LEE, AND A. RABINOVICH, *Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks*, in International conference on machine learning, PMLR, 2018, pp. 794–803.

[17] Z. CHEN, J. NGIAM, Y. HUANG, T. LUONG, H. KRETZSCHMAR, Y. CHAI, AND D. ANGUELOV, *Just pick a sign: Optimizing deep multitask models with gradient sign dropout*, in Proceedings of the 33rd Advances in Neural Information Processing Systems, 2020.

[18] C. CUI, Z. SHEN, J. HUANG, M. CHEN, M. XU, M. WANG, AND Y. YIN, *Adaptive feature aggregation in deep multi-task convolutional neural networks*, IEEE Transactions on Circuits and Systems for Video Technology, 32 (2021), pp. 2133–2144.

[19] A. B. DA SILVA AND M. GAZEAU, *A general system of differential equations to model first-order adaptive algorithms.*, J. Mach. Learn. Res., 21 (2020).

[20] K. DEB, K. SINDHYA, AND J. HAKANEN, *Multi-objective optimization*, in Decision sciences, CRC Press, 2016, pp. 161–200.

[21] K. DEB AND A. SINHA, *Solving bilevel multi-objective optimization problems using evolutionary algorithms*, in Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2009, pp. 110–124.

[22] J.-A. DÉSIDÉRI, *Multiple-gradient descent algorithm (mgda) for multiobjective optimization*, Comptes Rendus Mathematique, 350 (2012), pp. 313–318.

[23] J. DEVLIN, M.-W. CHANG, K. LEE, AND K. TOUTANOVA, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805, (2018).

[24] S. DIAMOND AND S. BOYD, *CVXPY: A Python-embedded modeling language for convex optimization*, Journal of Machine Learning Research, 17 (2016), pp. 1–5.

[25] J. DOMKE, *Generic methods for optimization-based modeling*, in Proceedings of the 15th Artificial Intelligence and Statistics, 2012, pp. 318–326.

[26] E. DUPONT, A. DOUCET, AND Y. W. TEH, *Augmented neural odes*, Advances in Neural Information Processing Systems, 32 (2019).

[27] G. EICHFELDER, *Twenty years of continuous multiobjective optimization in the twenty-first century*, EURO Journal on Computational Optimization, 9 (2021), p. 100014.

[28] D. ERIKSSON, M. PEARCE, J. GARDNER, R. D. TURNER, AND M. POLOCZEK, *Scalable global optimization via local bayesian optimization*, Advances in neural information processing systems, 32 (2019).

[29] A. FALLAH, A. MOKHTARI, AND A. OZDAGLAR, *On the convergence theory of gradient-based model-agnostic meta-learning algorithms*, in Proceedings of the International Conference on Artificial Intelligence and Statistics, 2020, pp. 1082–1092.

[30] M. FENG AND S. LI, *An approximate strong KKT condition for multiobjective optimization*, TOP, 26 (2018), pp. 489–509.

[31] H. D. FERNANDO, H. SHEN, M. LIU, S. CHAUDHURY, K. MURUGESAN, AND T. CHEN, *Mitigating gradient bias in multi-objective learning: A provably convergent approach*, in International Conference on Learning Representations, 2023.

[32] C. FINLAY, J.-H. JACOBSEN, L. NURBEKYAN, AND A. OBERMAN, *How to train your neural ode: the world of jacobian and kinetic regularization*, in International conference on machine learning, PMLR, 2020, pp. 3154–3164.

[33] J. FLIEGE, A. I. F. VAZ, AND L. N. VICENTE, *Complexity of gradient descent for multiobjective optimization*, Optimization Methods and Software, 34 (2019), pp. 949–959.

[34] L. FRANCESCHI, M. DONINI, P. FRASCONI, AND M. PONTIL, *Forward and reverse gradient-based hyperparameter optimization*, in International Conference on Machine Learning, 2017.

[35] L. FRANCESCHI, P. FRASCONI, S. SALZO, R. GRAZZI, AND M. PONTIL, *Bilevel programming for hyperparameter optimization and meta-learning*, in Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 1568–1577.

[36] K. GAO AND O. SENER, *Generalizing gaussian smoothing for random search*, in International Conference on Machine Learning, PMLR, 2022, pp. 7077–7101.

[37] Y. GAO, H. BAI, Z. JIE, J. MA, K. JIA, AND W. LIU, *Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning*, in Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, 2020, pp. 11543–11552.

[38] Y. GAO, J. MA, M. ZHAO, W. LIU, AND A. L. YUILLE, *Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 3205–3214.

[39] J. GILMER, S. S. SCHOENHOLZ, P. F. RILEY, O. VINYALS, AND G. E. DAHL, *Neural message passing for quantum chemistry*, in International Conference on Machine Learning, 2017.

[40] C. GONG, X. LIU, AND Q. LIU, *Automatic and harmless regularization with constrained and lexicographic optimization: A dynamic barrier approach*, in Neural Information Processing Systems, 2021.

[41] W. GRATHWOHL, R. T. CHEN, J. BETTENCOURT, AND D. DUVENAUD, *Scalable reversible generative models with free-form continuous dynamics*, in International Conference on Learning Representations, 2019.

[42] R. GRAZZI, L. FRANCESCHI, M. PONTIL, AND S. SALZO, *On the iteration complexity of hypergradient computation*, in International Conference on Machine Learning, 2020.

[43] P. GUO, C.-Y. LEE, AND D. ULBRICHT, *Learning to branch for multi-task learning*, in International conference on machine learning, PMLR, 2020, pp. 3854–3863.

[44] T. HAARNOJA, A. ZHOU, K. HARTIKAINEN, G. TUCKER, S. HA, J. TAN, V. KUMAR, H. ZHU, A. GUPTA, P. ABBEEL, ET AL., *Soft actor-critic algorithms and applications*, arXiv preprint arXiv:1812.05905, (2018).

[45] C. HAN, L. CUI, R. ZHU, J. WANG, N. CHEN, Q. SUN, X. LI, AND M. GAO, *When gradient descent meets derivative-free optimization: A match made in black-box scenario*, arXiv preprint arXiv:2305.10013, (2023).

[46] N. HANSEN, *The cma evolution strategy: a comparing review*, Towards a new evolutionary computation: Advances in the estimation of distribution algorithms, (2006), pp. 75–102.

[47] X. HE, Z. ZHENG, AND Y. ZHOU, *Mmes: Mixture model-based evolution strategy for large-scale optimization*, IEEE Transactions on Evolutionary Computation, 25 (2020), pp. 320–333.

[48] T. HOSPEDALES, A. ANTONIOU, P. MICAELLI, AND A. STORKEY, *Meta-learning in neural networks: A survey*, IEEE transactions on pattern analysis and machine intelligence, 44 (2021), pp. 5149–5169.

[49] A. JAVALOY AND I. VALERA, *Rotograd: Gradient homogenization in multitask learning*, in Proceedings of the 10th International Conference on Learning Representations, 2022.

[50] K. JI, J. YANG, AND Y. LIANG, *Bilevel optimization: Convergence analysis and enhanced design*, in International conference on machine learning, PMLR, 2021, pp. 4882–4892.

[51] Y. JI, S. QU, AND Z. YU, *A new method for solving multiobjective bilevel programs*, Discrete Dynamics in Nature and Society, 2017 (2017).

[52] R. JIANG, N. ABOLFAZLI, A. MOKHTARI, AND E. Y. HAMEDANI, *A conditional gradient-based method for simple bilevel optimization with convex lower-level problem*, in International Conference on Artificial Intelligence and Statistics, 2023.

[53] A. KENDALL, Y. GAL, AND R. CIPOLLA, *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, 2018, pp. 7482–7491.

[54] M. KONAKOVIC LUKOVIC, Y. TIAN, AND W. MATUSIK, *Diversity-guided multi-objective bayesian optimization with batch evaluations*, Advances in Neural Information Processing Systems, 33 (2020), pp. 17708–17720.

[55] V. KONDA AND J. TSITSIKLIS, *Actor-critic algorithms*, Advances in neural information processing systems, 12 (1999).

[56] A. KUMAR AND H. DAUME III, *Learning task grouping and overlap in multi-task learning*, arXiv preprint arXiv:1206.6417, (2012).

[57] A. KURAKIN, I. J. GOODFELLOW, AND S. BENGIO, *Adversarial examples in the physical world*, in Proceedings of the 5th International Conference on Learning Representations, 2017.

[58] J. KWON, D. KWON, S. WRIGHT, AND R. D. NOWAK, *A fully first-order method for stochastic bilevel optimization*, in International Conference on Machine Learning, 2023.

[59] M. LAUMANNS, L. THIELE, E. ZITZLER, E. WELZL, AND K. DEB, *Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem*, in Parallel Problem Solving from Nature-PPSN VII: 7th International Conference Granada, Spain, September 7–11, 2002 Proceedings 7, Springer, 2002, pp. 44–53.

[60] M. LECHNER AND R. HASANI, *Learning long-term dependencies in irregularly-sampled time series*, arXiv preprint arXiv:2006.04418, (2020).

[61] Y. LECUN, L. BOTTOU, Y. BENGIO, AND P. HAFFNER, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[62] J. LIANG, E. MEYERSON, AND R. MIIKKULAINEN, *Evolutionary architecture search for deep multitask networks*, in Proceedings of the genetic and evolutionary computation conference, 2018, pp. 466–473.

[63] B. LIN, W. JIANG, F. YE, Y. ZHANG, P. CHEN, Y.-C. CHEN, S. LIU, AND J. T. KWOK, *Dual-balancing for multi-task learning*, arXiv preprint arXiv:2308.12029, (2023).

[64] B. LIN, F. YE, Y. ZHANG, AND I. TSANG, *Reasonable effectiveness of random weighting: A litmus test for multi-task learning*, Transactions on Machine Learning Research, (2022).

[65] B. LIN AND Y. ZHANG, *LibMTL: A Python library for multi-task learning*, Journal of Machine Learning Research, 24 (2023), pp. 1–7.

[66] B. LIU, X. LIU, X. JIN, P. STONE, AND Q. LIU, *Conflict-averse gradient descent for multi-task learning*, Advances in Neural Information Processing Systems, 34 (2021), pp. 18878–18890.

[67] B. LIU, M. YE, S. WRIGHT, P. STONE, ET AL., *Bome! bilevel optimization made easy: A simple first-order approach*, in Neural Information Processing Systems, 2022.

[68] G. LIU, L. ZHAO, F. YANG, J. BIAN, T. QIN, N. YU, AND T.-Y. LIU, *Trust region evolution strategies*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4352–4359.

[69] H. LIU, K. SIMONYAN, AND Y. YANG, *DARTS: differentiable architecture search*, in Proceedings of the 7th International Conference on Learning Representations, 2019.

[70] L. LIU, Y. LI, Z. KUANG, J. XUE, Y. CHEN, W. YANG, Q. LIAO, AND W. ZHANG, *Towards impartial multi-task learning*, in International Conference on Learning Representations, 2021.

[71] R. LIU, J. GAO, J. ZHANG, D. MENG, AND Z. LIN, *Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2021).

[72] R. LIU, X. LIU, X. YUAN, S. ZENG, AND J. ZHANG, *A value-function-based interior-point method for non-convex bi-level optimization*, in International Conference on Machine Learning, 2021.

[73] R. LIU, P. MU, X. YUAN, S. ZENG, AND J. ZHANG, *A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton*, in International Conference on Machine Learning, PMLR, 2020, pp. 6305–6315.

[74] S. LIU, E. JOHNS, AND A. J. DAVISON, *End-to-end multi-task learning with attention*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1871–1880.

[75] S. LIU AND L. N. VICENTE, *The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning*, Annals of Operations Research, (2021), pp. 1–30.

[76] Y. LIU, Y. LU, H. LIU, Y. AN, Z. XU, Z. YAO, B. ZHANG, Z. XIONG, AND C. GUI, *Hierarchical prompt learning for multi-task learning*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 10888–10898.

[77] Z. LIU, P. LUO, X. WANG, AND X. TANG, *Deep learning face attributes in the wild*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 3730–3738.

[78] Y. LU, A. KUMAR, S. ZHAI, Y. CHENG, T. JAVIDI, AND R. FERIS, *Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5334–5343.

[79] R. LUCCHETTI, *Convexity and well-posed problems*, Springer Science & Business Media, 2006.

[80] R. LUCCHETTI AND E. MIGLIERINA, *Stability for convex vector optimization problems*, Optimization, 53 (2004), pp. 517–528.

[81] Y. LYU AND I. W. TSANG, *Black-box optimizer with stochastic implicit natural gradient*, in Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21, Springer, 2021, pp. 217–232.

[82] Y. LYU, Y. YUAN, AND I. W. TSANG, *Efficient batch black-box optimization with deterministic regret bounds*, arXiv preprint arXiv:1905.10041, (2019).

[83] D. MAHAPATRA AND V. RAJAN, *Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization*, in Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 6597–6607.

[84] K.-K. MANINIS, I. RADOSAVOVIC, AND I. KOKKINOS, *Attentive single-tasking of multiple tasks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1851–1860.

[85] A. MAURER, M. PONTIL, AND B. ROMERA-PAREDES, *The benefit of multitask representation learning*, Journal of Machine Learning Research, 17 (2016), pp. 1–32.

[86] N. MILOJKOVIC, D. ANTOGNINI, G. BERGAMIN, B. FALTINGS, AND C. MUSAT, *Multi-gradient descent for multi-objective recommender systems*, arXiv preprint arXiv:2001.00846, (2019).

[87] I. MISRA, A. SHRIVASTAVA, A. GUPTA, AND M. HEBERT, *Cross-stitch networks for multi-task learning*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 3994–4003.

[88] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. RIEDMILLER, A. K. FIDJELAND, G. OSTROVSKI, ET AL., *Human-level control through deep reinforcement learning*, nature, 518 (2015), pp. 529–533.

[89] A. NAVON, A. SHAMSIAN, I. ACHITUVE, H. MARON, K. KAWAGUCHI, G. CHECHIK, AND E. FETAYA, *Multi-task learning as a bargaining game*, in International Conference on Machine Learning, PMLR, 2022, pp. 16428–16446.

[90] S. J. PAN AND Q. YANG, *A survey on transfer learning*, IEEE Transactions on knowledge and data engineering, 22 (2009), pp. 1345–1359.

[91] A. PASZKE, S. GROSS, F. MASSA, A. LERER, AND ET AL, *PyTorch: An imperative style, high-performance deep learning library*, in Neural Information Processing Systems, 2019.

[92] F. PEDREGOSA, *Hyperparameter optimization with approximate gradient*, in International Conference on Machine Learning, 2016.

[93] H. PHAM, M. GUAN, B. ZOPH, Q. LE, AND J. DEAN, *Efficient neural architecture search via parameters sharing*, in International conference on machine learning, PMLR, 2018, pp. 4095–4104.

[94] S. QU, M. GOH, AND F. T. CHAN, *Quasi-newton methods for solving multiobjective optimization*, Operations Research Letters, 39 (2011), pp. 397–399.

[95] A. RADFORD, J. W. KIM, C. HALLACY, A. RAMESH, G. GOH, S. AGARWAL, G. SASTRY, A. ASKELL, P. MISHKIN, J. CLARK, ET AL., *Learning transferable visual models from natural language supervision*, in International conference on machine learning, PMLR, 2021, pp. 8748–8763.

[96] C. RAFFEL, N. SHAZEER, A. ROBERTS, K. LEE, S. NARANG, M. MATENA, Y. ZHOU, W. LI, AND P. J. LIU, *Exploring the limits of transfer learning with a unified text-to-text transformer*, The Journal of Machine Learning Research, 21 (2020), pp. 5485–5551.

[97] R. RAMAKRISHNAN, P. O. DRAL, M. RUPP, AND O. A. VON LILIENFELD, *Quantum chemistry structures and properties of 134 kilo molecules*, Scientific Data, 1 (2014), pp. 1–7.

[98] D. J. REZENDE, S. MOHAMED, AND D. WIERSTRA, *Stochastic backpropagation and approximate inference in deep generative models*, in International conference on machine learning, PMLR, 2014, pp. 1278–1286.

[99] C. ROSENBAUM, T. KLINGER, AND M. RIEMER, *Routing networks: Adaptive selection of non-linear functions for multi-task learning*, in International Conference on Learning Representations, 2018.

[100] S. RUUSKA AND K. MIETTINEN, *Constructing evolutionary algorithms for bilevel multiobjective optimization*, in Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–7.

[101] K. SAENKO, B. KULIS, M. FRITZ, AND T. DARRELL, *Adapting visual category models to new domains*, in European Conference on Computer Vision, 2010.

[102] T. SALIMANS, J. HO, X. CHEN, S. SIDOR, AND I. SUTSKEVER, *Evolution strategies as a scalable alternative to reinforcement learning*, arXiv preprint arXiv:1703.03864, (2017).

[103] O. SENER AND V. KOLTUN, *Multi-task learning as multi-objective optimization*, in Advances in Neural Information Processing Systems, 2018, pp. 525–536.

[104] A. SHABAN, C.-A. CHENG, N. HATCH, AND B. BOOTS, *Truncated back-propagation for bilevel optimization*, in Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, 2019, pp. 1723–1732.

[105] N. SILBERMAN, D. HOIEM, P. KOHLI, AND R. FERGUS, *Indoor segmentation and support inference from rgbd images*, in European conference on computer vision, Springer, 2012, pp. 746–760.

[106] A. SINHA, *Bilevel multi-objective optimization problem solving using progressively interactive emo*, in Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2011, pp. 269–284.

[107] A. SINHA, P. MALO, AND K. DEB, *Towards understanding bilevel multi-objective optimization with deterministic lower level decisions*, in International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 426–443.

[108] S. SODHANI, A. ZHANG, AND J. PINEAU, *Multi-task reinforcement learning with context-based representations*, in International Conference on Machine Learning, PMLR, 2021, pp. 9767–9779.

[109] D. SOW, K. JI, Z. GUAN, AND Y. LIANG, *A constrained optimization approach to bilevel optimization with multiple inner minima*, arXiv preprint arXiv:2203.01123, (2022).

[110] M. SRINIVAS AND L. M. PATNAIK, *Genetic algorithms: A survey*, computer, 27 (1994), pp. 17–26.

[111] N. SRINIVAS, A. KRAUSE, S. M. KAKADE, AND M. SEEGER, *Gaussian process optimization in the bandit setting: No regret and experimental design*, arXiv preprint arXiv:0912.3995, (2009).

[112] G. STREZOSKI, N. V. NOORD, AND M. WORRING, *Many task learning with task routing*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1375–1384.

[113] G. SUN, T. PROBST, D. P. PAUDEL, N. POPOVIĆ, M. KANAKIS, J. PATEL, D. DAI, AND L. VAN GOOL, *Task switching network for multi-task learning*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 8291–8300.

[114] T. SUN, Z. HE, H. QIAN, Y. ZHOU, X.-J. HUANG, AND X. QIU, *Bbtv2: towards a gradient-free future with large language models*, in Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 2022, pp. 3916–3930.

[115] T. SUN, Z. HE, Q. ZHU, X. QIU, AND X.-J. HUANG, *Multitask pre-training of modular prompt for chinese few-shot learning*, in Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 11156–11172.

[116] T. SUN, Y. SHAO, H. QIAN, X. HUANG, AND X. QIU, *Black-box tuning for language-model-as-a-service*, in Proceedings of ICML, 2022.

[117] X. SUN, R. PANDA, R. FERIS, AND K. SAENKO, *Adashare: Learning what to share for efficient deep multi-task learning*, Advances in Neural Information Processing Systems, 33 (2020), pp. 8728–8740.

[118] K. SWERSKY, J. SNOEK, AND R. P. ADAMS, *Multi-task bayesian optimization*, Advances in neural information processing systems, 26 (2013).

[119] H. TANABE, E. H. FUKUDA, AND N. YAMASHITA, *Proximal gradient methods for multiobjective optimization and their applications*, Computational Optimization and Applications, 72 (2019), pp. 339–361.

[120] S. VANDENHENDE, S. GEORGOULIS, B. DE BRABANDERE, AND L. VAN GOOL, *Branched multi-task networks: Deciding what layers to share*, Proceedings BMVC 2020, (2019).

[121] S. VANDENHENDE, S. GEORGOULIS, W. VAN GANSBEKE, M. PROESMANS, D. DAI, AND L. VAN GOOL, *Multi-task learning for dense prediction tasks: A survey*,

IEEE transactions on pattern analysis and machine intelligence, 44 (2021), pp. 3614–3633.

[122] H. VENKATESWARA, J. EUSEBIO, S. CHAKRABORTY, AND S. PANCHANATHAN, *Deep hashing network for unsupervised domain adaptation*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5018–5027.

[123] G. G. WANG AND S. SHAN, *An efficient pareto set identification approach for multi-objective optimization on black-box functions*, in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 46946, 2004, pp. 279–291.

[124] Y. WANG, Q. YAO, J. T. KWOK, AND L. M. NI, *Generalizing from a few examples: A survey on few-shot learning*, ACM Computing Surveys, 53 (2020), pp. 1–34.

[125] Z. WANG, R. PANDA, L. KARLINSKY, R. FERIS, H. SUN, AND Y. KIM, *Multitask prompt tuning enables parameter-efficient transfer learning*, arXiv preprint arXiv:2303.02861, (2023).

[126] Z. WANG, Y. TSVETKOV, O. FIRAT, AND Y. CAO, *Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models*, in International Conference on Learning Representations, 2021.

[127] D. WIERSTRA, T. SCHAUL, T. GLASMACHERS, Y. SUN, J. PETERS, AND J. SCHMIDHUBER, *Natural evolution strategies*, The Journal of Machine Learning Research, 15 (2014), pp. 949–980.

[128] J. WON, J. PARK, K. KIM, AND J. LEE, *How to train your dragon: example-guided control of flapping flight*, ACM Transactions on Graphics (TOG), 36 (2017), pp. 1–13.

[129] H. XIA, V. SULIAFU, H. JI, T. NGUYEN, A. BERTOZZI, S. OSHER, AND B. WANG, *Heavy ball neural ordinary differential equations*, Advances in Neural Information Processing Systems, 34 (2021), pp. 18646–18659.

[130] H. XIAO, K. RASUL, AND R. VOLLGRAF, *Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms*, arXiv preprint arXiv:1708.07747, (2017).

[131] Y. YAO, J. CAO, AND H. CHEN, *Robust task grouping with representative tasks for clustered multi-task learning*, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1408–1417.

[132] F. YE, B. LIN, X. CAO, Y. ZHANG, AND I. TSANG, *A first-order multi-gradient algorithm for multi-objective bi-level optimization*, arXiv preprint arXiv:2401.09257, (2024).

[133] F. YE, B. LIN, Z. YUE, P. GUO, Q. XIAO, AND Y. ZHANG, *Multi-objective meta learning*, Advances in Neural Information Processing Systems, 34 (2021), pp. 21338–21351.

[134] F. YE, B. LIN, Z. YUE, Y. ZHANG, AND I. W. TSANG, *Multi-objective meta-learning*, Artificial Intelligence, 335 (2024), p. 104184.

[135] F. YE, Y. LYU, X. WANG, Y. ZHANG, AND I. TSANG, *Adaptive stochastic gradient algorithm for black-box multi-objective learning*, in International Conference on Learning Representations, 2024.

[136] F. YE, X. WANG, Y. ZHANG, AND I. W. TSANG, *Multi-task learning via time-aware neural ODE*, in International Joint Conference on Artificial Intelligence, 2023.

[137] C. YILDIZ, M. HEINONEN, AND H. LAHDESMAKI, *Ode2vae: Deep generative second order odes with bayesian neural networks*, Advances in Neural Information Processing Systems, 32 (2019).

[138] F. YU, V. KOLTUN, AND T. FUNKHOUSER, *Dilated residual networks*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 472–480.

[139] L. YU, W. JIANG, H. SHI, J. YU, Z. LIU, Y. ZHANG, J. T. KWOK, Z. LI, A. WELLER, AND W. LIU, *Metamath: Bootstrap your own mathematical questions for large language models*, arXiv preprint arXiv:2309.12284, (2023).

[140] R. YU, W. CHEN, X. WANG, AND J. KWOK, *Enhancing meta learning via multi-objective soft improvement functions*, in International Conference on Learning Representations, 2023.

[141] T. YU, S. KUMAR, A. GUPTA, S. LEVINE, K. HAUSMAN, AND C. FINN, *Gradient surgery for multi-task learning*, Advances in Neural Information Processing Systems, 33 (2020), pp. 5824–5836.

[142] T. YU, D. QUILLEN, Z. HE, R. JULIAN, K. HAUSMAN, C. FINN, AND S. LEVINE, *Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning*, in Conference on robot learning, PMLR, 2020, pp. 1094–1100.

[143] A. R. ZAMIR, A. SAX, W. SHEN, L. J. GUIBAS, J. MALIK, AND S. SAVARESE, *Taskonomy: Disentangling task transfer learning*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3712–3722.

[144] M.-L. ZHANG AND Z.-H. ZHOU, *A review on multi-label learning algorithms*, IEEE transactions on knowledge and data engineering, 26 (2013), pp. 1819–1837.

[145] R. ZHANG AND D. GOLOVIN, *Random hypervolume scalarizations for provable multi-objective black box optimization*, in International Conference on Machine Learning, PMLR, 2020, pp. 11096–11105.

[146] Y. ZHANG AND Q. YANG, *A survey on multi-task learning*, IEEE Transactions on Knowledge and Data Engineering, 34 (2022), pp. 5586–5609.

[147] A. ZHOU, B.-Y. QU, H. LI, S.-Z. ZHAO, P. N. SUGANTHAN, AND Q. ZHANG, *Multiobjective evolutionary algorithms: A survey of the state of the art*, Swarm and Evolutionary Computation, 1 (2011), pp. 32–49.

[148] K. ZHOU, J. YANG, C. C. LOY, AND Z. LIU, *Conditional prompt learning for vision-language models*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 16816–16825.

[149] S. ZHOU, W. ZHANG, J. JIANG, W. ZHONG, J. GU, AND W. ZHU, *On the convergence of stochastic multi-objective gradient manipulation and beyond*, Advances in Neural Information Processing Systems, 35 (2022), pp. 38103–38115.

[150] A. ŽILINSKAS, *A statistical model-based algorithm for 'black-box' multi-objective optimisation*, International Journal of Systems Science, 45 (2014), pp. 82–93.