ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240793

A First-Order Multi-Gradient Algorithm for Multi-Objective Bi-Level Optimization

Feiyang Ye[a,b], Baijiong Lin[c], Xiaofeng Cao[d], Yu Zhang[a,e,*] and Ivor W. Tsang[b,f,g,h]

[a]Department of Computer Science and Engineering, Southern University of Science and Technology
[b]Australian Artificial Intelligence Institute, University of Technology Sydney
[c]The Hong Kong University of Science and Technology (Guangzhou)
[d]School of Artificial Intelligence, Jilin University
[e]Shanghai Artificial Intelligence Laboratory
[f]Centre for Frontier AI Research, Agency for Science, Technology and Research
[g]Institute of High Performance Computing, Agency for Science, Technology and Research
[h]School of Computer Science and Engineering, Nanyang Technological University

Abstract. In this paper, we study the Multi-Objective Bi-Level Optimization (MOBLO) problem, where the upper-level subproblem is a multi-objective optimization problem and the lower-level subproblem is for scalar optimization. Existing gradient-based MOBLO algorithms need to compute the Hessian matrix, causing the computational inefficient problem. To address this, we propose an efficient first-order multi-gradient method for MOBLO, called FORUM. Specifically, we reformulate MOBLO problems as a constrained multi-objective optimization (MOO) problem via the value-function approach. Then we propose a novel multi-gradient aggregation method to solve the challenging constrained MOO problem. Theoretically, we provide the complexity analysis to show the efficiency of the proposed method and a non-asymptotic convergence result. Empirically, extensive experiments demonstrate the effectiveness and efficiency of the proposed FORUM method in different learning problems. In particular, it achieves state-of-the-art performance on three multi-task learning benchmark datasets. The code is available at https://github.com/Baijiong-Lin/FORUM.

1 Introduction

In this work, we study the Multi-Objective Bi-Level Optimization (MOBLO) problem, which is formulated as

$$\min_{\alpha\in\mathbb{R}^n,\omega\in\mathbb{R}^p} F(\alpha,\omega) \quad \text{s.t. } \omega\in\mathcal{S}(\alpha)=\arg\min_{\omega} f(\alpha,\omega), \quad (1)$$

where α and ω denote the Upper-Level (UL) and Lower-Level (LL) variables, respectively. The UL subproblem, $F := (F_1, F_2, \dots, F_m)^\top : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$, is a vector-valued jointly continuous function for m desired objectives. $\mathcal{S}(\alpha)$ denotes the optimal solution set (which is usually assumed to be a singleton set [11, 42]) of the LL subproblem by minimizing a continuous function $f(\alpha,\omega) : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ w.r.t. ω. In this work, we focus on MOBLO with a singleton set $\mathcal{S}(\alpha)$ and a non-convex UL subproblem, where F_i is a non-convex function for all i. MOBLO has demonstrated its superiority in various learning problems such as neural architecture search [26, 49], reinforcement learning [1, 41], multi-task learning [42], and meta-learning [42, 47].

* Corresponding Author. Email: yu.zhang.ust@gmail.com

Table 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 [42]	asymptotic	$\mathcal{O}(mp(n+p)T)$	$\mathcal{O}(mn+mpT)$
MoCo [10]	non-asymptotic	$\mathcal{O}(mp(n+p)T)$	$\mathcal{O}(2mn+mpT)$
FORUM	non-asymptotic	$\mathcal{O}(mn+p(m+T))$	$\mathcal{O}(mn+mp)$

Recently, MOML [42, 45] and MoCo [10] 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 [11, 12, 15]. 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 propose an efficient **F**irst-**O**rde**R** m**U**lti-gradient method for **M**OBLO (**FORUM**). Specifically, we reformulate MOBLO as an equivalent constrained multi-objective optimization (MOO) problem by the value-function-based approach [29, 25, 38]. 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. Theoretically, we provide the complexity analysis showing that FORUM is more efficient than MOML and MoCo in both time and memory costs, as summarised in Table 1. In addition, we provide a non-asymptotic convergence analysis for FORUM. Empirically, we evaluate the effectiveness and efficiency of FORUM on two learning problems, i.e., multi-objective data hyper-

cleaning and multi-task learning on three benchmark datasets.

The main contributions of this work are three-fold:

- We propose the FORUM method, an efficient gradient-based algorithm for the MOBLO problem;
- We demonstrate that the proposed FORUM method is more efficient than existing MOBLO methods from the perspective of complexity analysis and provide a non-asymptotic convergence analysis;
- Extensive experiments demonstrate the effectiveness and efficiency of the proposed FORUM method. In particular, it achieves state-of-the-art performance on three benchmark datasets under the setting of multi-task learning.

2 Related Works

Multi-Objective Optimization. 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 [2], evolutionary-based [51, 5], and gradient-based [8, 30]. In this paper, we focus on the gradient-based category. One notable gradient-based MOO method is the MGDA [8] 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 [10, 52] have shown their superiority in many learning problems such as multi-task learning [35] and reinforcement learning [48], especially when some objectives are conflicting.

Bi-Level Optimization. BLO [28] is a type of optimization problem with a hierarchical structure, where one subproblem is nested within another subproblem. The MOBLO problem (1) reduces degrades to the BLO problem when m equals 1. One representative category of the BLO method is the ITD-based methods [11, 12, 15] 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 [25, 29, 38] 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.

Multi-Objective Bi-Level Optimization. MOML [42, 45] 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 [10] 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. [47] 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. MORBiT [16] studies a BLO problem with multiple objectives in its UL subproblem but it formulates the UL subproblem as a min-max problem, which is different from problem (1) we focus on in this paper.

3 The FORUM Algorithm

In this section, we introduce the proposed FORUM method. Firstly, we reformulate MOBLO as an equivalent constrained multi-objective problem via the value-function-based approach in Section 3.1. Next, we provide a novel multi-gradient aggregation method to solve the constrained multi-objective problem in Section 3.2.

3.1 Reformulation of MOBLO

Based on the value-function-based approach [29, 25, 38, 19], we reformulate MOBLO problem (1) as an equivalent single-level *constrained multi-objective optimization* problem:

$$\min_{\alpha\in\mathbb{R}^n,\omega\in\mathbb{R}^p} F(\alpha,\omega) \quad \text{s.t.} \quad f(\alpha,\omega)\le f^*(\alpha), \tag{2}$$

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 $\mathcal{Z}\equiv\mathbb{R}^n\times\mathbb{R}^p$. Then, we have $F(z)\equiv F(\alpha,\omega)$ and $f(z)\equiv f(\alpha,\omega)$. Thus, problem (2) can be rewritten as

$$\min_{z\in\mathcal{Z}} F(z) \quad \text{s.t.} \quad q(z)\le 0, \tag{3}$$

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

$$\nabla_\alpha f^*(\alpha)=\nabla_\alpha f(\alpha,\omega^*(\alpha))=\nabla_\alpha f(\alpha,\omega^*), \tag{4}$$

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

However, solving problem (3) is challenging for two reasons. One reason is that the Slater's condition [6], which is required for duality-based optimization methods, does not hold for problem (3), since the constraint $q(z)\le 0$ is ill-posed [29, 17] 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) cannot be treated as classic constrained optimization and we propose a novel gradient method to solve it in Section 3.2. 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

$$\tilde{\omega}^{t+1}=\tilde{\omega}^t-\eta\nabla_\omega f(\alpha,\tilde{\omega}^t),\quad t=0,\cdots,T-1, \tag{5}$$

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 $\nabla_z\widetilde{q}(z)$. We show that the approximation error of the gradient $\nabla_z\widetilde{q}(z)$ exponentially decays w.r.t. the LL iterations T in Appendix A.1 [44]. Hence, problem (3) is modified to

$$\min_{z\in\mathcal{Z}} F(z) \quad \text{s.t.} \quad \widetilde{q}(z)=f(z)-f(\alpha,\tilde{\omega}^T)\le 0. \tag{6}$$

3.2 Multi-Gradient Aggregation Method

In this section, we introduce the proposed multi-gradient aggregation method for solving problem (6) iteratively. Specifically, at k-th iteration, assume z_k is updated by $z_{k+1}=z_k+\mu d_k$ 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 $\widetilde{q}(z) \leq 0$ 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

$$\begin{aligned}\max_{d} \min_{i\in\{1,\dots,m\}} &(F_i(z_k) - F_i(z_k+\mu d)) \\ &\approx -\mu \min_{d} \max_{i\in\{1,\dots,m\}} \langle \nabla F_i(z_k), d\rangle.\end{aligned} \tag{7}$$

To regularize the update direction, we add a regularization term $\frac{1}{2}\|d\|^2$ to problem (7) and compute d_k by solving $\min_d \max_{i\in\{1,\dots,m\}} \langle \nabla F_i(z_k), d\rangle + \frac{1}{2}\|d\|^2$.

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 [14], 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:

$$\begin{aligned}\min_{d} \max_{i\in\{1,\dots,m\}} &\langle \nabla F_i(z_k), d\rangle + \frac{1}{2}\|d\|^2, \\ \text{s.t. } &\langle \nabla\widetilde{q}(z_k), d\rangle \leq -\phi_k.\end{aligned} \tag{8}$$

Based on the Lagrangian multiplier method, problem (8) has a solution as

$$d_k = -\left(\sum_{i=1}^m \lambda_i^k \nabla F_i(z_k) + \nu(\lambda^k)\nabla\widetilde{q}(z_k)\right), \tag{9}$$

where Lagrangian multipliers $\lambda^k = (\lambda_1^k, \dots, \lambda_m^k) \in \Delta^{m-1}$ (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

$$\begin{aligned}\nu(\lambda) &= \max\left(\sum_{i=1}^m \lambda_i \pi_i(z_k), 0\right), \\ \text{with } \pi_i(z_k) &= \frac{2\phi_k - \langle \nabla\widetilde{q}(z_k), \nabla F_i(z_k)\rangle}{\|\nabla\widetilde{q}(z_k)\|^2}.\end{aligned} \tag{10}$$

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

$$\lambda^k = \arg\min_{\lambda\in\Delta^{m-1}} \frac{1}{2}\left\|\sum_{i=1}^m \lambda_i \nabla F_i(z_k) + \nu(\lambda)\nabla\widetilde{q}(z_k)\right\|^2 - \nu(\lambda)\phi_k. \tag{11}$$

The detailed derivations of the above procedure are put in Appendix A.2 [44]. Problem (11) can be reformulated as

$$\begin{aligned}\min_{\lambda\in\Delta^{m-1},\gamma} &\frac{1}{2}\left\|\sum_{i=1}^m \lambda_i \nabla F_i(z_k) + \gamma\nabla\widetilde{q}(z_k)\right\|^2 - \gamma\phi_k \\ \text{s.t. } &\gamma \geq 0, \gamma \geq \sum_{i=1}^m \lambda_i \pi_i(z_k).\end{aligned} \tag{12}$$

The first term of the objective function in problem (12) can be simplified to $R^\top \Lambda^\top \Lambda R$, where $R = (\lambda_1, \dots, \lambda_m, \gamma)^\top$ and $\Lambda = (\nabla F_1, \dots, \nabla F_m, \nabla\widetilde{q})$. Note that the dimension of the matrix $\Lambda^\top\Lambda$ is $(m+1)\times(m+1)$, which is independent with the dimension of z.

Algorithm 1 The FORUM Method

Require: 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 $\nabla_z F_i(z_k)$, $i = 1, \dots, m$;
  Compute λ^k by solving problem (12);
  Update $\tilde{\lambda}^k$ by $\tilde{\lambda}^k = (1-\beta_k)\tilde{\lambda}^{k-1} + \beta_k\lambda^k$;
  Compute $\nu(\tilde{\lambda}^k)$ via Eq. (10);
  Compute d_k via Eq. (9);
  Update z as $z_{k+1} = z_k + \mu d_k$;
end for
return z_K.

As the number of UL objectives m is usually small compared with the dimension of z, solving problem (12) does not incur too much computational cost. In practice, we can use the open-source `CVXPY` library [9] to solve problem (12).

To ensure convergence, the sequence of $\{\lambda^k\}_{k=1}^K$ should be a convergent sequence (refer to the discussion in Appendix A.3 [44]). However, $\{\lambda^k\}_{k=1}^K$ obtained by directly solving the problem (12) in each iteration cannot ensure such properties. Therefore, we apply a momentum strategy [52, 46] to λ to generate a stable sequence and further guarantee the convergence. Specifically, in k-th iteration, we first solve the problem (12) to obtain λ^k, then update the weights by $\tilde{\lambda}^k = (1-\beta_k)\tilde{\lambda}^{k-1} + \beta_k\lambda^k$, 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 $\nu(\tilde{\lambda}^k)$ via Eq. (10). Then we obtain the update direction d_k by Eq. (9) and update the variable z_k as $z_{k+1} = z_k + \mu d_k$. The entire FORUM algorithm is shown in Algorithm 1.

4 Analysis

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

4.1 Complexity Analysis

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 (12), 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.

We provide a complexity analysis for the existing MOBLO methods (i.e., MOML and MoCo). 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.

4.2 Convergence Analysis

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

Assumption 1. *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 1 are widely adopted in non-convex multi-objective optimization [52, 10]. Then we make an assumption for the LL subproblem.

Assumption 2. *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 2 is commonly used in the analysis for the BLO [11, 12] and MOBLO problems [10, 42]. The proposed FORUM algorithm focuses on generating one Karush-Kuhn-Tucker (KKT) stationary point of the original constrained multi-objective optimization problem (3). Following [14, 25], we measure the convergence of problem (3) by both its KKT stationary condition and the feasibility condition, where detailed definitions are provided in Appendix B.1 [44]. Specifically, we denote by $\mathcal{K}(z_k) = \left\| \sum_{i=1}^m \tilde{\lambda}_i^k \nabla F_i(z_k) + \nu_k \nabla q(z_k) \right\|^2$ 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), 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 1. *Suppose that Assumptions 1 and 2 hold, and the sequence $\{z_k\}_{k=0}^K$ generated by Algorithm 1 satisifes $q(z_k) \leq B$, 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*

$$\max \left\{ \min_{k<K} \mathcal{K}(z_k), q(z_k) \right\} = \mathcal{O}(K^{-1/4} + \Gamma(T)), \qquad (13)$$

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

The proof is put in Appendix B.3 in the supplementary material [44]. Theorem 1 gives a non-asymptotic convergence result for Algorithm 1 based on the KKT stationary condition and the feasibility condition of the problem (3). 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).

5 Experiments

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

5.1 Data Hyper-Cleaning

Setup. Data hyper-cleaning [3, 11, 25, 36] 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-objective 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

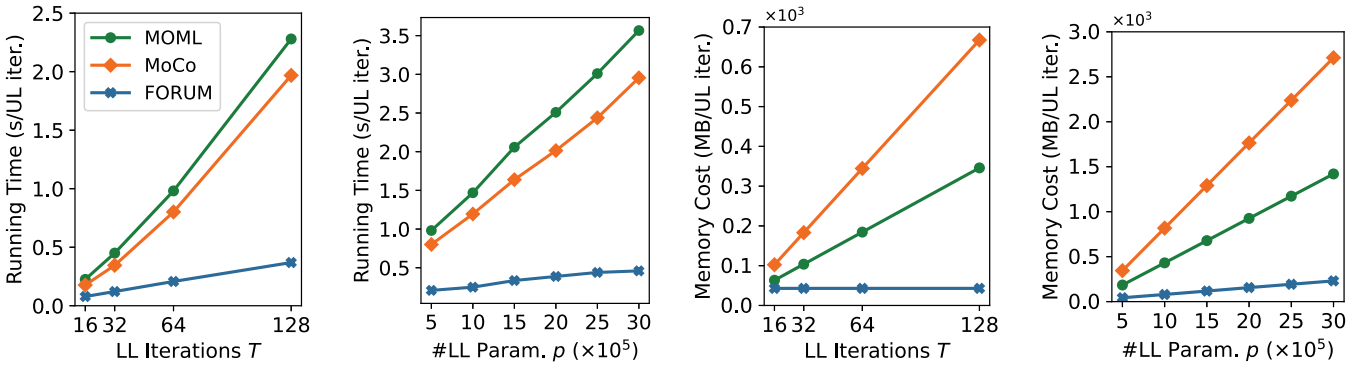
$$\mathcal{L}_i^{\text{tr}}(\alpha, \omega; \mathcal{D}_i^{\text{tr}}) = \frac{1}{N_i} \sum_{j=1}^{N_i} \sigma(\alpha_{i,j}) \ell(\omega; x_{i,j}, y_{i,j})$$

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-objective data hyper-cleaning is formulated as

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

Datasets. We conduct experiments on the MNIST [20] and Fashion-MNIST [40] datasets. Each dataset corresponds to a 10-class image classification problem. All the images have the same size of 28×28. Following [3], 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. The proposed FORUM method is compared with two *MOBLO methods*: MOML [42, 45] and MoCo [10]. The same configuration is used for both the MOML, MoCo, and FORUM methods. Specifically, the hard-parameter sharing architecture [50] 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



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

Figure 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 $T = 64$.

Table 2: Performance of different methods on the MNIST and FashionMNIST datasets for the multi-objective data hyper-cleaning problem. Each experiment is repeated over 3 random seeds, and the mean and the standard deviation are reported. The best result is marked in **bold**.

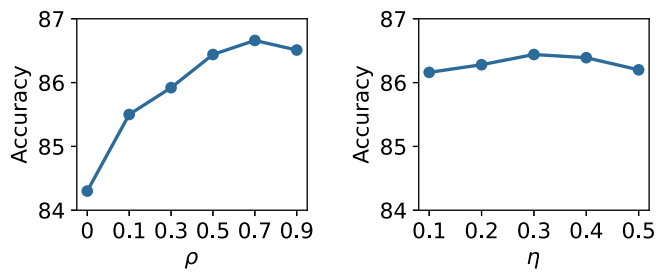
Methods	MNIST		FashionMNIST	
	Accuracy (%)	F1 Score	Accuracy (%)	F1 Score
MOML [42]	$88.64_{\pm 0.94}$	$88.61_{\pm 0.98}$	$80.64_{\pm 0.35}$	$80.60_{\pm 0.49}$
MoCo [10]	$88.05_{\pm 1.21}$	$88.03_{\pm 1.27}$	$80.94_{\pm 0.19}$	$80.67_{\pm 0.25}$
FORUM	$\mathbf{90.81}_{\pm 0.14}$	$\mathbf{90.81}_{\pm 0.15}$	$\mathbf{82.07}_{\pm 0.38}$	$\mathbf{81.72}_{\pm 0.57}$

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 1(b) and 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 [32] to compute the GPU memory cost in Figures 1(c) and 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 2. As can be seen, the proposed FORUM method outperforms the MOML and MoCo in both datasets, which demonstrates its effectiveness.

Figures 1(a) and 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 1(c) and 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 1 match the complexity analysis in Section 4.1 and demonstrate that FORUM is more efficient than MOML and MoCo in terms of both time and space complexity.

Effects of η and ρ. We conduct additional experiments to study the effects of hyperparameters η and ρ in the data hyper-cleaning problem. The results are shown in Figure 2. FORUM is insensitive with η and a large ρ (e.g, $\rho = 0.5, 0.7, 0.9$). Besides, FORUM with a positive ρ performs better than $\rho = 0$, which shows the effectiveness of ϕ_k introduced in Section 3.2.



(a) Accuracy vs. ρ. (b) Accuracy vs. η.

Figure 2: Effects of ρ (**Left**) and η (**Right**) in the multi-objective data hyper-cleaning problem. "Accuracy" denotes the average accuracy on MNIST and FashionMNIST datasets.

5.2 *Multi-Task Learning*

Setup. Multi-Task Learning (MTL) [50, 23] aims to train a single model to solve several tasks simultaneously. Following MOML [42], we aim to learn the loss weights to balance different tasks and improve the generalization performance by casting MTL as a MOBLO problem. Specifically, suppose there are m tasks and the i-th task has its corresponding dataset $\mathcal{D}_i$ that contains a training set $\mathcal{D}_i^{\text{tr}}$ and a validation set $\mathcal{D}_i^{\text{val}}$. The MTL model is parameterized by ω and $\alpha \in \Delta^{m-1}$ denotes the loss weights for the m tasks. Let $\mathcal{L}(\omega; \mathcal{D})$ represent the average loss of model ω on the dataset $\mathcal{D}$. The MOBLO formulation for MTL is as

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

We conduct experiments on three benchmark datasets among three different task categories, i.e., the Office-31 [34] dataset for the image classification task, the NYUv2 [37] dataset for the scene understanding task, and the QM9 dataset [33] for the molecular property prediction task.

Datasets. The Office-31 dataset [34] 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 [22]: 60% for training, 20% for validation, and 20% for testing.

Table 3: Classification accuracy (%) on Office-31. Each experiment is repeated over 3 random seeds and the average 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 [50]	83.53	97.27	96.85	92.55	-0.61
UW [18]	83.82	97.27	96.67	92.58	-0.56
MGDA [35]	85.47	95.90	97.03	92.80	-0.27
PCGrad [48]	83.59	96.99	96.85	92.48	-0.68
GradDrop [7]	84.33	96.99	96.30	92.54	-0.59
GradVac [39]	83.76	97.27	96.67	92.57	-0.58
CAGrad [24]	83.65	95.63	96.85	92.04	-1.13
IMTL [27]	83.70	96.44	96.29	92.14	-1.02
Nash-MTL [31]	85.01	97.54	97.41	93.32	+0.24
RLW [22]	83.82	96.99	96.85	92.55	-0.59
first-order bi-level optimization methods					
BVFIM [29]	84.84	96.99	97.78	93.21	+0.11
BOME [25]	85.53	96.72	**98.15**	93.47	+0.41
multi-objective bi-level optimization methods					
MOML [42]	84.67	96.72	96.85	92.75	-0.36
MoCo [10]	84.38	97.26	97.03	92.89	-0.22
FORUM	**85.64**	**98.63**	97.96	**94.07**	**+0.96**

The NYUv2 dataset [37], an indoor scene understanding dataset, has 795 and 654 images for training and testing, respectively. It has three tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction.

The QM9 dataset [33], a molecular property prediction dataset. We use the same data split as in Nash-MTL [31]: $110,000$ 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 FORUM method is 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) [50], UW [18], MGDA [35], PCGrad [48], GradDrop [7], GradVac [39], CAGrad [24], IMTL[27], Nash-MTL [31], and RLW [22]; two *first-order BLO methods*: BVFIM [29] and BOME [25], 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; two *gradient-based MOBLO methods*: MOML [42] and MoCo [10].

Evaluation Metrics. For the Office-31 dataset, following RLW [22], 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 [22], 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 [31], we use mean absolute error (MAE) as the evaluation metric.

Following [22, 23, 43], 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 $\Delta_{\mathrm{p}} = 100\% \times \frac{1}{m}\sum_{i=1}^{m}\frac{1}{N_i}\sum_{j=1}^{N_i}\frac{(-1)^{s_{i,j}}(M_{i,j}-M_{i,j}^{\mathrm{STL}})}{M_{i,j}^{\mathrm{STL}}}$, 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.

Implementation Details. For the Office-31 dataset, following RLW [22], we use a pre-trained ResNet-18 network as a shared backbone among tasks and a fully connected layer as the task-specific head. All input images are resized to 224×224. The batch size is set to 64. The cross-entropy loss is used for all tasks. The number of UL epochs K is set to 100. An Adam optimizer with the learning rate as 10^{-4} and the weight decay as 10^{-5} is used for updating model parameters ω in the UL subproblem.

For the NYUv2 dataset, following RLW [22], we use the DeepLabV3+ architecture [4], which contains a ResNet-50 network with dilated convolutions as the shared encoder among all tasks and three Atrous Spatial Pyramid Pooling (ASPP) [4] modules as task-specific heads. All input images are resized to 288×384. The batch size is set to 8. The cross-entropy loss, L_1 loss, and cosine loss are used as the loss function of the three tasks, respectively. The total number of UL epochs K is set to 200. An Adam optimizer with the learning rate as 10^{-4} and the weight decay as 10^{-5} is used for updating model parameters ω in the UL subproblem. The learning rate of ω is halved after 100 epochs.

For the QM9 dataset, following Nash-MTL [31], we use a graph neural network [13] 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 [32] is used to reduce the learning rate of ω once Δ_{p} on the validation dataset stops improving.

All methods are implemented based on the open-source `LibMTL` library [21]. 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 proposed 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 shows the results on Office-31. We can see that FORUM outperforms all baselines from different categories in terms of average classification accuracy and Δ_{p}, highlighting its effectiveness. The results on NYUv2 dataset are shown in Table 4. As can be seen, only FORUM achieves better performance than STL in terms of Δ_{p}. Moreover, FORUM performs well in the depth estimation and surface normal prediction tasks. The results on QM9 dataset are shown in Table 5. The QM9 dataset is a challenging dataset in MTL and none of the MTL methods performs better than STL, as observed in previous research [31]. We can see that FORUM again outperforms all the baselines in terms of Δ_{p}. Those results consistently demonstrate FORUM achieves state-of-the-art performance and is more effective than previous MOBLO methods such as MOML and MoCo.

Table 4: Results on the NYUv2 dataset. Each experiment is repeated over 3 random seeds and the average 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					$\Delta_{\mathrm{p}}\uparrow$
					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 [50]	53.93	75.53	0.3825	0.1577	23.5691	17.0149	35.04	60.99	72.05	-1.78
UW [18]	**54.29**	75.64	0.3815	0.1583	23.4805	16.9206	35.26	61.17	72.21	-1.52
MGDA [35]	53.52	74.76	0.3852	0.1566	22.7400	16.0000	37.12	63.22	73.84	-0.64
PCGrad [48]	53.94	75.62	0.3804	0.1578	23.5226	16.9276	35.19	61.17	72.19	-1.57
GradDrop [7]	53.73	75.54	0.3837	0.1580	23.5392	16.9587	35.17	61.06	72.07	-1.85
GradVac [39]	54.21	75.67	0.3859	0.1583	23.5804	16.9055	35.34	61.15	72.10	-1.75
CAGrad [24]	53.97	75.54	0.3885	0.1588	22.4701	15.7139	**37.77**	63.82	74.30	-0.27
IMTL [27]	53.63	75.44	0.3868	0.1592	22.5800	15.8500	37.44	63.52	74.09	-0.57
Nash-MTL [31]	53.41	74.95	0.3867	0.1612	22.5662	15.9365	37.30	63.40	74.09	-1.01
RLW [22]	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 [29]	53.29	75.07	0.3981	0.1632	22.3552	15.9710	37.15	63.44	74.27	-1.68
BOME [25]	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 [42]	53.59	75.48	0.3839	0.1577	23.1487	16.5319	36.06	62.05	72.89	-1.26
MoCo [10]	53.73	75.63	0.3838	0.1560	23.1922	16.5737	36.02	61.93	72.82	-1.06
FORUM	54.04	75.64	**0.3795**	**0.1555**	**22.1870**	**15.6815**	37.71	**64.04**	**74.67**	**+0.65**

Table 5: Mean absolute error (MAE) on the QM9 dataset. Each experiment is repeated over 3 random seeds and the average 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	$\Delta_{\mathrm{p}}\uparrow$
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 [50]	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 [18]	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 [35]	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 [48]	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 [7]	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 [39]	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 [24]	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 [27]	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 [31]	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 [22]	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 [29]	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 [25]	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 [42]	**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 [10]	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	0.104	0.266	85.37	82.15	2.126	6.49	96.97	97.53	97.69	95.88	**0.097**	**-73.36**

6 Conclusion

In this paper, we propose FORUM, an efficient fully first-order gradient-based method for solving the multi-objective bi-level optimization problem. Specifically, we reformulate the original MOBLO problem to a constrained MOO problem and we propose a novel multi-gradient aggregation method to solve it. Compared with the existing MOBLO methods, FORUM does not require any hypergradient computation and thus is efficient. Theoretically, we provide a complexity analysis to show the efficiency of the proposed method and a non-asymptotic convergence guarantee for FORUM with a non-convex multi-objective UL subproblem. 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.

Acknowledgements

This work is supported by National Key R&D Program of China (No. 2022ZD0160300), NSFC key grant 62136005, NSFC general grant 62076118, NSFC young scientists fund 62206108, and Shenzhen fundamental research program JCYJ20210324105000003.

References

[1] A. Abdolmaleki, S. Huang, L. Hasenclever, M. Neunert, F. Song, M. Zambelli, M. Martins, N. Heess, R. Hadsell, and M. Riedmiller. A distributional view on multi-objective policy optimization. In *International Conference on Machine Learning*, 2020.

[2] 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.

[3] 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.

[4] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision*, 2018.

[5] 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.

[6] L. Chen, J. Xu, and J. Zhang. On bilevel optimization without lower-level strong convexity. *arXiv preprint arXiv:2301.00712*, 2023.

[7] 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 *Neural Information Processing Systems*, 2020.

[8] J.-A. Désidéri. Multiple-gradient descent algorithm (MGDA) for multi-objective optimization. *Comptes Rendus Mathematique*, 350(5):313–318, 2012.

[9] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[10] 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.

[11] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, 2017.

[12] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 2018.

[13] 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.

[14] 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.

[15] R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, 2020.

[16] A. Gu, S. Lu, P. Ram, and T.-W. Weng. Min-max multi-objective bilevel optimization with applications in robust machine learning. In *International Conference on Learning Representations*, 2023.

[17] 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.

[18] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[19] 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.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

[21] B. Lin and Y. Zhang. LibMTL: A Python library for multi-task learning. *Journal of Machine Learning Research*, 24(209):1–7, 2023.

[22] 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.

[23] 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.

[24] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu. Conflict-averse gradient descent for multi-task learning. In *Neural Information Processing Systems*, 2021.

[25] 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.

[26] J. Liu and Y. Jin. Multi-objective search of robust neural architectures against multiple types of adversarial attacks. *Neurocomputing*, 453: 73–84, 2021.

[27] 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.

[28] 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*, 44(12):10045–10067, 2021.

[29] 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.

[30] D. Mahapatra and V. Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*, 2020.

[31] 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*, 2022.

[32] 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.

[33] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):1–7, 2014.

[34] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, 2010.

[35] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In *Neural Information Processing Systems*, 2018.

[36] A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots. Truncated back-propagation for bilevel optimization. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[37] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012.

[38] 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.

[39] 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.

[40] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[41] R. Yang, X. Sun, and K. Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Neural Information Processing Systems*, 2019.

[42] F. Ye, B. Lin, Z. Yue, P. Guo, Q. Xiao, and Y. Zhang. Multi-objective meta learning. In *Neural Information Processing Systems*, 2021.

[43] 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.

[44] 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.

[45] F. Ye, B. Lin, Z. Yue, Y. Zhang, and I. W. Tsang. Multi-objective meta-learning. *Artificial Intelligence*, 335:104184, 2024.

[46] 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.

[47] 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.

[48] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. In *Neural Information Processing Systems*, 2020.

[49] Z. Yue, B. Lin, Y. Zhang, and C. Liang. Effective, efficient and robust neural architecture search. In *International Joint Conference on Neural Networks*, 2022.

[50] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2022.

[51] 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(1):32–49, 2011.

[52] S. Zhou, W. Zhang, J. Jiang, W. Zhong, J. Gu, and W. Zhu. On the convergence of stochastic multi-objective gradient manipulation and beyond. In *Neural Information Processing Systems*, 2022.