Noname manuscript No.
(will be inserted by the editor)

Adaptive multi-gradient methods for quasiconvex vector optimization and applications to multi-task learning

Tran Ngoc Thang · Nguyen Anh Minh

February 27, 2023

Abstract For solving a broad class of nonconvex multiobjective programming problems on an unbounded constraint set, we provide an adaptive step-size strategy that does not include line-search techniques and establish convergence of a generic approach under mild assumptions. Specifically, the objective function may not satisfy the convexity condition. It does not need a previously established Lipschitz constant for determining an initial step-size, as is the case with descent line-search algorithms. The crucial feature of this process is the steady reduction of the step size until a certain condition is fulfilled. In particular, it may be used to provide a novel multi-gradient projection approach to unbounded constrained optimization problems. The correctness of the method is verified by preliminary results from some computational examples. To demonstrate the effectiveness of the proposed technique for large scale problems, we apply it to some experiments on multi-task learning.

Keywords nonconvex multi-objective programming · gradient descent algorithms · quasiconvex functions · pseudoconvex functions · adaptive step-sizes

Mathematics Subject Classification (2000) 90C25 · 90C26 · 68Q32 · 93E35

1 Introduction

Gradient descent methods are a common tool for a wide range of programming problems, from convex to nonconvex, for both scalar and vector optimization, and have numerous practical applications (see [3], [6], [13] and references therein). At each iteration, gradient descent algorithms provide an iterative series of solutions based on gradient directions and step sizes. For a long time, researchers have focused on finding the direction to improve the

Tran Ngoc Thang
School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, 1st Dai Co Viet street, Hanoi, Viet Nam
E-mail: thang.tranngoc@hust.edu.vn
Nguyen Anh Minh
School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, 1st Dai Co Viet street, Hanoi, Viet Nam
E-mail: minh.na194117@hust.edu.vn

convergence rate of techniques, while the step-size was determined using one of the few well-known approaches (see [3], [17]).

Multicriteria optimization algorithms that do not scalarize have recently been developed (see, e.g., [?] for an overview on the subject). Some of these techniques are extensions of scalar optimization algorithms, such as notably the steepest descent algorithm [?] with at most linear convergence based on the gradient descent method, while others borrow heavily from ideas developed in heuristic optimization [?]. For the latter, no convergence proofs are known, and empirical results show that convergence generally is, as in the scalar case, quite slow [?].

Recently, new major areas of machine learning applications with high dimensionality and nonconvex objective functions have required the development of novel step-size choosing procedures to reduce the method's overall computing cost (see [6], [13]). The exact or approximate one-dimensional minimization line-search incurs significant computational costs per iteration, particularly when calculating the function value is nearly identical to calculating its derivative and requires the solution of complex auxiliary problems (see [3]). To avoid the line-search, the step-size value may be calculated using prior information such as Lipschitz constants for the gradient. However, this requires using just part of their inexact estimations, which leads to slowdown convergence. This is also true for the well-known divergent series rule (see [11], [17]).

In this research, we propose a novel and line-search-free adaptive step-size algorithm for a broad class of multiobjective programming problems where the objective function is nonconvex smooth and the constraint set is unbounded closed convex. A crucial component of this procedure is gradually decreasing the step size until a predetermined condition is fulfilled. Although the Lipschitz continuity of the gradient of the objective function is required for convergence, the approach does not make use of predetermined constants. The proposed change has been shown to be effective in preliminary computational tests. We perform various machine learning experiments, including multi-task learning and neural networks for classification, to show that the proposed method performs well on large-scale tasks.

The rest of this paper is structured as follows. Section 2 provides preliminaries and details the problem formulation. Section 3 summarizes the primary results, including the proposed algorithms. Section 4 depicts numerical experiments and analyzes computational outcomes. Section 5 presents applications to certain machine learning problems. The final section makes some conclusions.

2 Preliminaries

2.1 Notations and definitions

Denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}_+$the set of non-negative real numbers, and by $\mathbb{R}_{++}$the set of strictly positive real numbers. Assume that $U \subset \mathbb{R}^n$ is a nonempty, closed and convex set, and $F : U \longrightarrow \mathbb{R}^m$ is a given function.

Definition 1 The problem is to find *an efficient point* or *Pareto optimum* of F, i.e., a point $x^* \in U$ such that $\nexists y \in U, F(y) \leq F(x^*)$, and $F(y) \neq F(x^*)$, where the inequality sign $\leq$ between vectors is to be understood in a componentwise sense.

In effect, we are employing the partial order induced by $\mathbb{R}^m_+ = \mathbb{R}_+ \times \cdots \times \mathbb{R}_+$ (the nonnegative orthant or Paretian cone of $\mathbb{R}^m$) defined by

$$F(y) \leq F(z) \Longleftrightarrow F(z) - F(y) \in \mathbb{R}^m_+,$$

and we are searching for minimal points induced by such partial order.

Definition 2 A point x^* is *weakly efficient* or *weak Pareto optimum* if

$$\nexists y \in U, \quad F(y) < F(x^*),$$

where the vector strict inequality $F(y) < F(x^*)$ is to be understood componentwise too.

This relation is induced by $\mathbb{R}^m_{++} = \mathbb{R}_{++} \times \cdots \times \mathbb{R}_{++}$, the interior of the Paretian cone $\left(F(y) < F(z)\right.$ if, and only if, $\left.F(z) - F(y) \in \mathbb{R}^m_{++}\right)$. Later on, we will make use of the negative of $\mathbb{R}^m_{++}$, i.e,

$$-\mathbb{R}^m_{++} = \left\{-v : v \in \mathbb{R}^m_{++}\right\}.$$

Definition 3 A point $x^* \in U$ is *locally efficient* (respectively, *locally weakly efficient*) if there is a neighborhood $V \subseteq U$ of x^* such that the point x^* is efficient (respectively, weakly efficient) for F restricted to V. Locally efficient points are also called *local Pareto optimal*, and locally weakly efficient points are also called *local weak Pareto optimal*.

Note that if U is convex and F is $\mathbb{R}^m_+$-convex (i.e., if F is componentwise-convex), then each local Pareto optimal point is globally Pareto optimal. Clearly, every locally efficient point is locally weakly efficient.

Definition 4 We say that a point $x^* \in U$ is a *Pareto stationary* point (or a *Pareto critical* point) of F if

$$DF(x^*)(U - x^*) \cap \left(-\mathbb{R}^m_{++}\right) = \emptyset, \tag{1}$$

where $DF(x^*)$ is the Jacobian matrix of F at x^* given by $DF(x) = (\nabla f_1(x), \cdots, \nabla f_m(x))^\top$. This notion is necessary (but in general not sufficient) for a point to be weak Pareto efficient and was first used in [15] to investigate a steepest descent algorithm. Note that in the case when $m = 1$, (2) is reduced to the classical optimality condition in scalar optimization.

Definition 5 [16] The differentiable function $f : \mathbb{R}^m \to \mathbb{R}$ is said to be

i) convex on U if for all $x, y \in U$, $\lambda \in [0,1]$, it holds that

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

ii) pseudoconvex on U if for all $x, y \in U$, it holds that

$$\langle \nabla f(x), y - x \rangle \geq 0 \Rightarrow f(y) \geq f(x).$$

iii) quasiconvex on U if for all $x, y \in U$, $\lambda \in [0;1]$, it holds that

$$f(\lambda x + (1-\lambda)y) \leq \max\{f(x); f(y)\}.$$

Proposition 1 *[8] The differentiable function f is quasiconvex on U if and only if*

$$f(y) \leq f(x) \Rightarrow \langle \nabla f(x), y - x \rangle \leq 0.$$

It is worth mentioning that "f is convex" $\Rightarrow$ "f is pseudoconvex" $\Rightarrow$ "f is quasiconvex" [16].

Proposition 2 *When F is pseudoconvex, i.e. the components of f are pseudoconvex, then* (2) *is a necessary and sufficient condition for a point to be weakly efficient.*

Indeed, suppose to the contrary, that there exists $y \in U$ such that $F(y) \prec F(x)$. Since F_i is pseudoconvex for each $i = 1, \ldots, m$, it follows that $\langle \nabla F_i(x), (y-x) \rangle < 0$ and therefore $DF(x)(y-x) \in -\mathbb{R}^m_{++}$, contradicting (2). This result, in general, does not hold for quasi-convex functions; see [?].

The range, or image space, of a matrix $M \in \mathbb{R}^{m\times n}$ will be denoted by $R(M)$ and $I \in \mathbb{R}^{n\times n}$ will denote the unit matrix. For two matrices $A, B \in \mathbb{R}^{n\times n}, B \leq A (B < A)$ will mean $A - B$ positive semidefinite (definite). In what follows, the Euclidean norm in $\mathbb{R}^n$ will be denoted by $\|\cdot\|$, and $B[x, r]$ denotes the closed ball of radius r with center $x \in \mathbb{R}^n$. We will use the same notation $\|\cdot\|$ for the induced operator norms on the corresponding matrix spaces.

For $x \in \mathbb{R}^m$, denote by $P_U(x)$ the projection of x onto U, i.e.,

$$P_U(x) := \operatorname{argmin}\{\|z - x\| : z \in U\}.$$

Proposition 3 [1] *It holds that*

(i) $\|P_U(x) - P_U(y)\| \leq \|x - y\|$ *for all* $x, y \in \mathbb{R}^m$,
(ii) $\langle y - P_U(x), x - P_U(x) \rangle \leq 0$ *for all* $x \in \mathbb{R}^m$, $y \in U$.

Proposition 4 *[8] Suppose that ∇f is L-Lipschitz continuous on U. For all $x, y \in U$, it holds that*

$$|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \frac{L}{2}\|y - x\|^2.$$

Lemma 1 *[18] Let $\{a_k\}$; $\{b_k\} \subset (0; \infty)$ be sequences such that*

$$a_{k+1} \leq a_k + b_k \ \forall k \geq 0; \quad \sum_{k=0}^{\infty} b_k < \infty.$$

Then, there exists the limit $\lim_{k\to\infty} a_k = c \in \mathbb{R}$.

2.2 Problem statements

2.2.1 Multi-objective optimization problem

We consider the following multi-objective optimization problem:

$$\operatorname{Min}_{x\in U} F(x), \qquad (\text{MOP}(F, U))$$

where the set $U \subset \mathbb{R}^n$ is nonempty, convex, and closed, and the vector function $F : U \longrightarrow \mathbb{R}^m$ is differentiable on an open set containing U. We denote the Jacobian matrix of F at x as follows:

$$DF(x) = \begin{bmatrix} \nabla F_1(x)^T \\ \vdots \\ \nabla F_m(x)^T \end{bmatrix} \in \mathbb{R}^{m\times n}$$

In this thesis, we consider the Jacobian matrix DF of F to be continuously Lipschitz, meaning that the component-wise derivatives $\nabla F_j \in \mathbb{R}^n$ of the function F_j are continuously Lipschitz for all $j = 1, ..., m$. We assume that the solution set of (MOP(F, U)) is nonempty. The following definitions are taken from [**?**].

Definition 6 A point $x^* \in U$ is called an *efficient point* or a Pareto optimal point of problem (MOP(F,U)) if $\nexists y \in U, F(y) \leq F(x^*)$ and $F(y) \neq F(x^*)$, where the $\leq$ sign between vectors is understood as the relationship between each component function. That is, $\nexists y \in U : F_i(y) \leq F_i(x^*), \forall i = 1,\dots,m$ and $\exists i_0 \in 1,\dots,m : F_{i_0}(y) < F_{i_0}(x^*)$. The set of Pareto optimal points forms the Pareto Front.

Considering $\mathbb{R}+^m := \mathbb{R}+ \times \cdots \times \mathbb{R}_+$, the partial order between vector function values is defined as:

$$F(y) \leq F(z) \Longleftrightarrow F(z) - F(y) \in \mathbb{R}^m_+,$$

Therefore, to solve problem (MOP(F,U)), we need to find a point that is a minimum in the partial order.

Definition 7 A point $x^* \in U$ is called a *weakly efficient* solution or a *weak Pareto optimal point* of problem (MOP(F,U)) if $\nexists y \in U$ such that $F(y) < F(x^*)$, where the inequality $F(y) < F(x^*)$ is also understood as the partial order.

Considering $\mathbb{R}{+}{+}^m := \mathbb{R}{+}{+} \times \cdots \times \mathbb{R}_{++}$, we also define

$$-\mathbb{R}{+}{+}^m = \{-v : v \in \mathbb{R}{+}{+}^m\}.$$

Definition 8 A point $x^* \in U$ is called a *locally efficient* point (or a weakly locally efficient point) of problem (MOP(F,U)) if there exists a neighborhood $V \subseteq U$ of x^* such that x^* is an efficient (or weakly efficient) point of problem (MOP(F,U)) restricted to V.

Note:

- When U is convex and F is $\mathbb{R}^m_+$-convex (i.e., each component function of F is convex), each locally efficient point is a globally efficient point.
- Each locally efficient point is also a weakly locally efficient point.

Definition 9 [7] Let $z \in U$ be a vector and $v \in \mathbb{R}^n$ be called a tangent direction of U at z if there exists a sequence $\left(z^k\right) k \subseteq U$ and a positive scalar $\lambda \in \mathbb{R}+$ such that

$$\lim_{k\to\infty} z^k = z \quad \text{and} \quad \lim_{k\to\infty} \lambda \frac{z^k - z}{\|z^k - z\|} = v$$

The set of all tangent directions of U at z is called the tangent cone of U at z and denoted by $T(U,z)$.

For a closed convex set U, we have $T(U,z) := U(z) := s \in \mathbb{R}^n | s = u - z$ for some $u \in U$.

Definition 10 A point $x^* \in U$ is called a *Pareto stationary point* (or *Pareto critical point*) of the vector function F over U if

$$DF\left(T(U,x^{)}\right) \cap \left(-\mathbb{R}^m_{++}\right) = \emptyset, \tag{2}$$

where $DF(x^*)$ is the Jacobian matrix of F at x^*.

This definition is a necessary condition (but not sufficient) for a point to be an effective Pareto point and was first used in [7] to define the steepest descent algorithm for multiobjective optimization. In the case of $m = 1$, (2) becomes the traditional optimization problem in single-objective optimization.

Lemma 2 *When the function F is pseudo-convex (i.e., its component functions $F_i, i = 1,\dots,m$ are pseudo-convex), then* (2) *is a necessary and sufficient condition for a point to be a weakly efficient point.*

Indeed, suppose the opposite, that there exists a point $y \in U$ such that $F(y) < F(x)$. Since F_i is pseudo-convex for each $i = 1,\dots,m$, it follows that $\langle \nabla F_i(x), (y-x)\rangle < 0$, and thus $DF(x)(y-x) \in -\mathbb{R}^m_{++}$, which contradicts (2) (note that we are considering the set U to be convex and closed).

2.2.2 Multi-Task Learning Problem

In this section, we model the MTL problem as an MOP problem based on the idea of Sener et al. [?].

The MTL problem involves performing m tasks with a vector loss function:

$$\min_{\theta} \mathscr{L}(\theta) = (\mathscr{L}_1(\theta), \mathscr{L}_2(\theta), \cdots, \mathscr{L}_m(\theta))^{\mathrm{T}}, \qquad \text{(MTL)}$$

where $\mathscr{L}_i(\theta)$ is the loss function for task i with parameter set θ for all tasks. MTL algorithms optimize all tasks simultaneously. Problem (MTL) is a multi-objective optimization problem, and there is no optimal solution or optimal parameter set that can simultaneously optimize all tasks. Therefore, we can only provide Pareto optimal solutions representing the trade-off between different tasks.

Definition of parameter dominance in MTL problem: Consider two parameter sets θ^a, θ^b in Ω, θ^a is said to be dominant over θ^b ($\theta^a \prec \theta^b$) if and only if $\mathscr{L}_i(\theta^a) \le \mathscr{L}_i(\theta^b), \forall i \in 1,\dots,m$ and $\mathscr{L}_j(\theta^a) < \mathscr{L}_j(\theta^b), \exists j \in 1,\dots,m$.

Definition of Pareto optimal point in MTL problem: θ^* is called the Pareto optimal solution and $\mathscr{L}(\theta^*)$ is called the Pareto optimal value if and only if there is no $\hat{\theta} \in \Omega$ such that $\hat{\theta} \prec \theta^*$. The set of all Pareto optimal points is called the Pareto optimal solution set, and their image is called the Pareto front.

3 The adaptive multi-gradient methods

3.1 Unconstraint case

Consider the following unconstrained multi-objective optimization problem:

$$\mathrm{Min}_{x\in\mathbb{R}^n} F(x). \qquad (\mathrm{MOP}(F,\mathbb{R}^n))$$

We now proceed by defining the adaptive steepest multi-gradient methods for the multi-objective problem with $U = \mathbb{R}^n$.

For $x \in \mathbb{R}^n$, we define $s(x)$, the steepest direction at x, as the optimal solution of

$$\begin{cases} \min\ \max_{j=1,\dots,m} \nabla F_j(x)^T s + \frac{1}{2} s^T s \\ \text{s.t.}\ \ s \in \mathbb{R}^n, \end{cases} \qquad (3)$$

First of all, observe that problem (3) always has a unique minimizer, since the functions $\nabla F_j(x)^T s + \frac{1}{2} s^T s$ are strongly convex for $j = 1,\dots,m$. Also note that for $m = 1$, the direction $s(x)$ is the "classical" steepest direction for scalar optimization, which is $-\nabla F(x)$.

Here, we are approximating

$$\max_{j=1,\dots,m} F_j(x+s) - F_j(x)$$

by the maximum of the local quadratic models at x of each F_j. Some comments are in order. First, note that in a standard scalarization approach, the multicriteria problem is replaced by a problem of minimizing a scalar function $x \mapsto \varphi(F_1(x),\dots,F_m(x))$, where φ might depend on some parameters. This is not the case in the approach taken here: We solve a scalar optimization problem to find a direction of descent for all objective functions involved. After a corresponding descent step, another, usually different, scalar optimization problem will be solved, and so on.

It is also important to remark that the idea of minimizing the maximum of quadratic approximations of the component functions variations in order to obtain a search direction has already been proposed in an algorithm for order-value optimization problems by Andreani et al. in [2], where at each iteration only the "ε-active" component functions are considered.

Search directions obtained by the minimization of the maximum of linear approximations (regularized by a quadratic term) of the components variations were previously considered for defining steepest descent-like methods for multiobjective optimization [7] and for vector optimization [**?**]. The optimal value of problem (3) will be denoted by $\Theta(x)$. Hence,

$$\Theta(x) = \inf_{s\in\mathbb{R}^n} \max_{j=1,\dots,m} \nabla F_j(x)^T s + \frac{1}{2} s^T s, \tag{4}$$

and

$$s(x) = \arg\min_{s\in\mathbb{R}^n} \max_{j=1,\dots,m} \nabla F_j(x)^T s + \frac{1}{2} s^T s \tag{5}$$

Although (3) is a nonsmooth problem, it can be framed as a convex quadratic optimization problem and so, it can be effectively solved. Indeed, (3) is equivalent to

$$\begin{cases} \min & g(t,s) = t \\ \text{s.t.} & \nabla F_j(x)^T s + \frac{1}{2} s^T s - t \le 0 \quad (1 \le j \le m) \\ & (t,s) \in \mathbb{R}\times\mathbb{R}^n. \end{cases} \tag{6}$$

The Lagrangian of this problem is

$$L((t,s),\lambda) = t + \sum_{j=1}^m \lambda_j \left(\nabla F_j(x)^T s + \frac{1}{2} s^T s - t \right).$$

Direct calculation of the Karush-Kuhn-Tucker conditions yields

$$\sum_{j=1}^m \lambda_j = 1, \quad \sum_{j=1}^m \lambda_j \left(\nabla F_j(x) + s\right) = 0, \tag{7}$$

$$\lambda_j \ge 0, \quad \nabla F_j(x)^T s + \frac{1}{2} s^T s \le t \quad (1 \le j \le m), \tag{8}$$

$$\lambda_j \left(\nabla F_j(x)^T s + \frac{1}{2} s^T s - t \right) = 0 \quad (1 \le j \le m). \tag{9}$$

Problem (6) has a unique solution, $(\Theta(x), s(x))$. As this is a convex problem and has a Slater point (e.g., $(1,0)$), there exists a KKT multiplier $\lambda = \lambda(x)$, which, together with $s = s(x)$ and $t = \Theta(x)$, satisfies conditions (7)-(9). In particular, from (7) we obtain

$$s(x) = -\sum_{j=1}^{m} \lambda_j(x) \nabla F_j(x). \tag{10}$$

So the steepest direction defined in this paper is a steepest direction for a standard scalar optimization problem, implicitly induced by weighting the given objective functions by the (nonnegative) a priori unknown KKT multipliers. As a consequence, the standard weighting factors [23], well known in multiobjective programming, do show up in our approach, albeit a posteriori and implicitly. In particular, it is not necessary to fix such weights in advance; every point $x \in U$ defines such weights by way of the KKT multipliers in the corresponding direction search program.

Existence of the KKT multipliers for the convex problem (6) implies that there is no duality gap, and so apply (10) we have:

$$\begin{aligned} \Theta(x) &= \sup_{\lambda \geq 0} \inf_{s \in \mathbb{R}^n} L((t,s),\lambda) \\ &= \sup_{\substack{\lambda \geq 0 \\ \sum \lambda_j = 1}} \inf_{s \in \mathbb{R}^n} \sum_{j=1}^{m} \lambda_j \left(\nabla F_j(x)^T s + \frac{1}{2} s^T s \right) \\ &= \sup_{\substack{\lambda \geq 0 \\ \sum \lambda_j = 1}} \inf_{s \in \mathbb{R}^n} \left(\sum_{j=1}^{m} \lambda_j(x) \nabla F_j(x)^T . s + \frac{1}{2} s^T s \right) \\ &= \sup_{\substack{\lambda \geq 0 \\ \sum \lambda_j = 1}} -\frac{1}{2} \| \sum_{j=1}^{m} \lambda_j(x) \nabla F_j(x) \|^2 \end{aligned} \tag{11}$$

Let us now study some properties of function θ and analyze its relation with $s(x)$ and stationarity of x.

Lemma 3 *Take $x \in \mathbb{R}^n$, then*

$$\Theta(x) = -\frac{1}{2} s(x)^T s(x) \text{ and } \|\Theta(x)\| = \frac{1}{2} \|s(x)\|^2.$$

Lemma 4 *Under our general assumptions, we have:*

1. *For any $x \in U, \Theta(x) \leq 0$.*
2. *The following conditions are equivalent.*
   (a) The point x is noncritical.
   (b) $\Theta(x) < 0$.
   (c) $s(x) \neq 0$. In particular, $x \in U$ is Pareto stationary if and only if $\Theta(x) = 0$.
3. *The function $s : U \to \mathbb{R}^n$, given by (5), is bounded on compact sets and $\Theta : U \to \mathbb{R}$, given by (4), is continuous in U.*

Now we sketch the Steepest algorithm for multi-criteria optimization. At each step, at a non-stationary point, we minimize the maximum of all local models as in (3) to obtain the Steepest step (5), which is a descent direction. After that, the step length is determined by means of an Adaptive rule coupled with a backtracking procedure. Under suitable local assumptions, full Steepest steps are always accepted and the generated sequence converges super-linear (or quadratically) to a local solution. Formally, the algorithm for finding a Pareto point is the following.

Adaptive Steepest Algorithm for Multi-criteria Optimization

Algorithm 1 Steepest multi-gradient with adaptive step size version 1 No Constraint

```
Step 1. Choose $\kappa \in [0,1], \sigma \in [0,1], \alpha_1 \in (0,+\infty)$, and $x^1 \in \mathbb{R}^n$. Set $k := 1$.
Step 2. (Main Loop)
(a) Solve the direction search program of this problem:
        $\min_{\lambda_j} ||\sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k)||^2$
        s.t: $\lambda_j \geq 0, \sum_{j=1}^{m} \lambda_j^k = 1$
to obtain the optimal solution:
$\lambda^{\mathbf{k}} = (\lambda_j^k), s(x^k) = -\left(\sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k)\right)$ and $\Theta(x^k) = -\frac{1}{2}||s(x^k)||^2$.
(b) If $\Theta(x^k) = 0$ then Stop. Otherwise, proceed to Step 2(c).
(c) Set $x^{k+1} := x^k + \alpha_k s(x^k)$.
(d) Compute a step length:
If $\sum_{j=1}^{m} \lambda_j^k F_j(x^{k+1}) \leq \sum_{j=1}^{m} \lambda_j^k F_j(x^{k+1}) + \sigma \langle s(x^k), x_{temp} - x^k \rangle$
then $\alpha_{k+1} = \alpha_k$, otherwise set $\alpha_{k+1} := \kappa\alpha_k$
Step 3. Set $k := k+1$ and goto Step 2.
```

Remark 1 Define for $x \in U$

$$H(x) = \sum_{j=1}^{m} \lambda_j F_j(x).$$

We also have

$$H(x^k) = \sum_{j=1}^{m} \lambda_j^k F_j(x^k) \text{ and } H(x^{k+1}) = \sum_{j=1}^{m} \lambda_j^k F_j(x^{k+1}).$$

Then $s(x^k) = -\nabla H(x^k) = -\sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k)$. If Algorithm **??** stops at step k, then x^k is a Pareto stationary point of the problem MOP(F,U). Indeed, since $x^{k+1} = P_U\left(x^k - \alpha_k \nabla H(x^k)\right)$, applying Proposition 3-(ii), we have

$$\left\langle z - x^{k+1}, x^k - \alpha_k \nabla G(x^k) - x^{k+1} \right\rangle \leq 0 \; \forall z \in U. \tag{12}$$

If $x^{k+1} = x^k$, we get

$$\left\langle \nabla G(x^k), z - x^k \right\rangle \geq 0 \quad \forall z \in U. \tag{13}$$

If $U = \mathbb{R}^n$ **then** $s(x^k) = \nabla H(x^k) = 0$, which means x^k is a Pareto stationary point of the problem. If, in addition, F is pseudoconvex, from Proposition 2, it implies that x^k is a weakly efficient solution of MOP(F,U).

Now, suppose that the algorithms generates an infinite sequence. We will prove that this sequence converges to a Pareto solution of the problem MOP(F,U).

Theorem 1 *Assume that the sequence $\left\{x^k\right\}$ is generated by Algorithm* **??***. Then, each limit point (if any) of the sequence $\left\{x^k\right\}$ is a Pareto stationary point of the problem MOP(F,U). Moreover,*

- *if F is quasiconvex on U, then the sequence $\{x^k\}$ converges to a Pareto stationary point of the problem.*
- *if F is pseudoconvex on U, then the sequence $\{x^k\}$ converges to a weakly efficient solution of the problem.*

Proof Applying Proposition 4, we get

$$H(x^{k+1}) \le H(x^k) + \left\langle \nabla H(x^k), x^{k+1} - x^k \right\rangle + \frac{L}{2}\|x^{k+1} - x^k\|^2. \tag{14}$$

In (12), taking $z = x^k \in U$, we arrive at

$$\left\langle \nabla H(x^k), x^{k+1} - x^k \right\rangle \le -\frac{1}{\alpha_k}\|x^{k+1} - x^k\|^2. \tag{15}$$

Combining (14) and (15), we obtain

$$H(x^{k+1}) \le H(x^k) - \sigma \left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle - \left(\frac{1-\sigma}{\alpha_k} - \frac{L}{2}\right)\|x^{k+1} - x^k\|^2. \tag{16}$$

We now claim that $\{\alpha_k\}$ is bounded away from zero, or in other words, the step size changes finite times. Indeed, suppose, by contrary, that $\alpha_k \to 0$. From (16), there exists $k_0 \in \mathbb{N}$ satisfying

$$H(x^{k+1}) \le H(x^k) - \sigma \left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle \ \forall k \ge k_0.$$

According to the construction of α_k, the last inequality implies that $\alpha_k = \alpha_{k_0}$ for all $k \ge k_0$. This is a contradiction. And so, there exists $k_1 \in \mathbb{N}$ such that for all $k \ge k_1$, we have $\alpha_k = \alpha_{k_1}$ and

$$H(x^{k+1}) \le H(x^k) - \sigma \left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle. \tag{17}$$

Noting that $\left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle \ge 0$, we infer that the sequence $\{H(x^k)\}$ is convergent and

$$\sum_{k=0}^{\infty} \left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle < \infty; \quad \sum_{k=0}^{\infty} \|x^{k+1} - x^k\|^2 < \infty. \tag{18}$$

From (12), for all $z \in U$, we have

$$\begin{aligned} \|x^{k+1} - z\|^2 &= \|x^k - z\|^2 - \|x^{k+1} - x^k\|^2 + 2\left\langle x^{k+1} - x^k, x^{k+1} - z \right\rangle \\ &\le \|x^k - z\|^2 - \|x^{k+1} - x^k\|^2 + 2\alpha_k \left\langle \nabla H(x^k), z - x^{k+1} \right\rangle. \end{aligned} \tag{19}$$

Let $\bar{x}$ be a limit point of $\{x^k\}$. There exists a subsequence $\{x^{k_i}\} \subset \{x^k\}$ such that $\lim_{i\to\infty} x^{k_i} = \bar{x}$. In (19), let $k = k_i$ and take the limit as $i \to \infty$. Noting that $\|x^k - x^{k+1}\| \to 0$, ∇G is continuous, we get

$$\langle \nabla G(\bar{x}), z - \bar{x} \rangle \ge 0 \quad \forall z \in U.$$

If $U = \mathbb{R}^n$ **then** $s(x^k) = \nabla H(x^k) = 0$, which means $\bar{x}$ is a Pareto stationary point of the problem MOP(F,U).

Now, suppose that F is quasiconvex on U. Denote

$$U := \left\{ x \in U : F(x) \le F(x^k) \quad \forall k \ge 0 \right\}.$$

Note that U contains the solution set of MOP(F,U), and hence, is not empty. Take $\hat{x} \in U$. Since $F(x^k) \geq F(\hat{x})$ for all $k \geq 0$, it implies that

$$\left\langle \nabla F(x^k), \hat{x} - x^k \right\rangle \leq 0, \ \forall k \geq 0. \tag{20}$$

Therefore,

$$\left\langle \nabla H(x^k), \hat{x} - x^k \right\rangle \leq 0 \quad \forall k \geq 0. \tag{21}$$

Combing (19) and (21), we get

$$\|x^{k+1} - \hat{x}\|^2 \leq \|x^k - \hat{x}\|^2 - \|x^{k+1} - x^k\|^2 + 2\alpha_k \left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle. \tag{22}$$

Applying Lemma 1 with $a_k = \|x^{k+1} - \hat{x}\|^2$, $b_k = 2\alpha_k \left\langle \nabla H(x^k), x^k - x^{k+1} \right\rangle$, we deduce that the sequence $\{\|x^k - \hat{x}\|\}$ is convergent for all $\hat{x} \in U$. Since the sequence $\{x^k\}$ is bounded, there exist a subsequence $\{x^{k_j}\} \subset \{x^k\}$ such that $\lim_{i\to\infty} x^{k_i} = \bar{x} \in U$. From (17) and (20), we know that the sequence $\left\{F(x^k)\right\}$ is nonincreasing and convergent. It implies that $\lim_{k\to\infty} F(x^k) = F(\bar{x})$ and $F(\bar{x}) \leq F(x^k)$ for all $k \geq 0$. This means $\bar{x} \in U$ and the sequence $\left\{\|x^k - \bar{x}\|\right\}$ is convergent. Thus,

$$\lim_{k\to\infty} \|x^k - \bar{x}\| = \lim_{i\to\infty} \|x^{k_i} - \bar{x}\| = 0.$$

Note that each limit point of $\left\{x^k\right\}$ is a Pareto stationary point of the problem. Then, the whole sequence $\left\{x^k\right\}$ converges to $\bar{x}$ - a Pareto stationary point of the problem.

Moreover, by Proposition 2, when F is pseudoconvex, this Pareto stationary point becomes a weakly efficient solution of MOP(F,U). □

Next, we estimate the convergence rate of Algorithm **??** in solving unconstrained optimization problems.

Corollary 1 *Assume that F is convex, $C = \mathbb{R}^m$ and $\{x^k\}$ is the sequence generated by Algorithm **??**. Then,*

$$H(x^k) - H(x^*) = O\left(\frac{1}{k}\right),$$

where x^* is a solution of the problem.

Proof Let x^* be a solution of the problem. Denote $\Delta_k := H(x^k) - H(x^*)$. From (17), noting that $x^k - x^{k+1} = \alpha_k \nabla G(x^k)$, we have

$$\Delta_{k+1} \leq \Delta_k - \sigma\alpha_{k_1} \|\nabla H(x^k)\|^2 \quad \forall k \geq k_1. \tag{23}$$

On the other hand, since the sequence $\left\{x^k\right\}$ is bounded and f is convex, it holds that

$$\begin{aligned} 0 \leq \Delta_k &\leq \left\langle \nabla H(x^k), x^k - x^* \right\rangle \\ &\leq M\|\nabla H(x^k)\|, \end{aligned} \tag{24}$$

where $M := \sup\left\{\|x^k - x^*\| : k \geq k_1\right\} < \infty$. From (23) and (24), we arrive at

$$\Delta_{k+1} \leq \Delta_k - Q\Delta_k^2 \quad \forall k \geq k_1, \tag{25}$$

where $Q := \frac{\sigma \alpha_{k_1}}{M^2}$. Noting that $\Delta_{k+1} \leq \Delta_k$, from (25), we obtain

$$\frac{1}{\Delta_{k+1}} \geq \frac{1}{\Delta_k} + Q \geq \ldots \geq \frac{1}{\Delta_{k_1}} + (k - k_1)Q,$$

which implies

$$H(x^k) - H(x^*) = O\left(\frac{1}{k}\right).$$

The experimental results for algorithm 1 will be presented in the section *Numerical experiments*. The solutions found by the algorithm will concentrate in a region on the Pareto front. Therefore, we use the idea of constructing linear constraints to separate the image space as in [2] to help evenly distribute the Pareto solutions.

3.2 Constraint case

We consider an optimization problem with F being a differentiable multi-objective function over an admissible set U, and the Jacobian matrix $DF \in \mathbb{R}^{m \times n}$ of F is continuously Lipschitz, meaning that the gradient $\nabla F_j \in \mathbb{R}^n$ of the component function F_j is Lipschitz continuous for all $j = 1, \ldots, m$. We consider the optimization problem:

$$\text{Min}_{x \in U} F(x), \qquad (MOP_{Cons}(F,U))$$

where $U := \{x \in \mathbb{R}^n | g_i(x) \leq 0, \forall i = 1, \ldots, l\}$ with the constraint functions g_i being differentiable for all $i = 1, \ldots, K$. We assume that the set of solutions to $(MOP_{Cons}(F,U))$ is nonempty.

The necessary condition for a point $x \in U$ to be a local Pareto optimal solution is:

$$DF(T(U,x)) \cap \left(-\mathbb{R}^m_{++}\right) = \emptyset, \tag{26}$$

which is equivalent to the system

$$(\nabla F_j(x))^\top s < 0, \quad \forall j = 1, \ldots, m, \quad s \in T(U,x), \tag{27}$$

having no solution s. The set of active constraints at x is denoted by $I_0(x)$,

$$I_0(x) := \{i \mid g_i(x) = 0\},$$

The linearized cone *(linearized cone)* at x, denoted by $C(x)$, is defined as:

$$C(x) := \left\{ s \in \mathbb{R}^n \mid (\nabla g_i(x))^\top s \leq 0, \forall i \in I_0(x) \right\}.$$

Assuming that the regular condition $C(x) = T(U,x)$ holds, then the condition (26) (and 27) is equivalent to the system

$$\begin{aligned} &\nabla F_j(x))^\top s < 0, \quad \forall j \\ &\nabla g_i(x))^\top s \leq 0 \quad \forall i \in I_0(x) \end{aligned}$$

having no solution $s \in \mathbb{R}^n$. Let $\hat{A}$ be the matrix obtained by appending the rows $(\nabla g_i(x))^\top$ for all binding constraints i underneath the Jacobian matrix $DF(x)$. The necessary condition for the system to have no solution is

$$R(\hat{A}) \cap -\mathbb{R}^m_{++} = \emptyset.$$

Consider the index set $I := 1,\dots,l$, $\varepsilon \in \mathbb{R}+$, and $x \in \mathbb{R}^n$. Using the idea from [7], we define the set I_ε as follows:

$$I_\varepsilon(x) := \{i \in I \mid g_i(x) \geq -\varepsilon\}.$$

Then, we proceed to solve the following optimization problem to find the descent direction for problem $(MOP_{Cons}(F,U))$:

$$\begin{aligned} &\min\ \alpha \\ &\text{s.t}\ (\nabla F_j(x))^\top s \leq \alpha, \quad j = 1,\dots,m, \\ &\quad (\nabla g_i(x))^\top s \leq \alpha, \quad j \in I_\varepsilon(x), \\ &\quad \|s\|_\infty \leq 1, \end{aligned} \tag{28}$$

The symbol $\Theta(x,\varepsilon)$ denotes the optimal value obtained. Using the Farkas lemma, we have $\Theta(x,0) = 0$ for a feasible point x if and only if there exist vectors $\lambda \in \mathbb{R}+^m \backslash 0$ and $\beta \in \mathbb{R}+^\ell$ such that

$$\sum_{j=1}^{m} \lambda_j \nabla F_j(x) + \sum_{i=1}^{\ell} \beta_i \nabla g_i(x) = 0$$

and

$$\sum_{i=1}^{\ell} \beta_i g_i(x) = 0$$

For $\lambda = (\lambda_j), \beta = (\beta_i) \in \mathbb{R}^m_{++}$, this is a sufficient condition for x to be a Pareto optimal solution.

3.2.1 *Solving problem* (MOP($F,\mathbb{R}^n$))

To evenly distribute the obtained solutions on the Pareto Front, we will build linear inequalities to decompose the image space similarly to the paper [2]. Let $\{u_1,u_2,\dots,u_K\} \in \mathbb{R}+^m$ be a set of K reference vectors. We decompose the image space into K subspaces Ω_k $(k = 1,\dots,K)$ using linear inequalities.

$$\Omega_k = \left\{ v \in R^m_+ \mid \langle u_i, v\rangle \leq \langle u_k, v\rangle, \forall i = 1,\dots,K \right\}.$$

We solve an unconstrained optimization problem on each region Ω_k by combining the inequality constraints $\mathscr{G}_p$ that separate the space. We need to solve the following optimization problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} F(x) = (F_1(x), F_2(x), \cdots, F_m(x)) \\ &\text{s.t}\ \mathscr{G}_p(x^*) = \langle u_p - u_k, F(x^*)\rangle \leq 0, \forall p = 1,\dots,K, \end{aligned} \tag{29}$$

Then, the problem (MOP($F,\mathbb{R}^n$)) will become problem (MOP(F,U)) with the set $U = \{x \in \mathbb{R}^n | \mathscr{G}_p(x) \leq 0,,, p = 1,\dots,K\}$.

Finding the initial point

To solve problem (29), we need to find an feasible initial point that satisfies all constraints. We use the method of finding an feasible starting point based on the descent direction using the idea of [2]. Before solving the optimization problem, we will find the initial point based on the following steps:

- **Step 1:** Randomly initialize a point $x_0 \in \mathbb{R}^n$.
- **Step 2:** If x_0 is an feasible point, Stop. Otherwise, proceed to **Step 3**.
- **Step 3:** Determine the set of tight constraints $I(x_0) = \{p \mid \mathscr{G}_p(x_0) \geq 0, p = 1, \ldots, K\}$. We can find a descent direction d_r as follows:

$$(d_r, \alpha_r) = \arg \min_{d \in \mathbb{R}^n, \alpha \in R} \alpha + \frac{1}{2} \|d\|^2, \text{ s.t } \nabla \mathscr{G}_p(x_0)^T d \leq \alpha, p \in I(x_0).$$

- **Step 4:** Update $x_0 = x_0 + \eta_r d_r$ and go back to **Step 2**.

After finding the initialization point, we will proceed with the main algorithm to solve the multi-objective optimization problem in the cases presented in the following section. Based on the theoretical content presented on the steepest descent direction for multi-objective optimization problems, we propose the following algorithm:

Algorithm 2 Steepest multi-gradient with adaptive step size version 2 No Constraints

Step 1. Set $\kappa \in [0,1]$, $\sigma \in [0,1]$, $\alpha_1 \in (0, +\infty)$, and $x_r^1 \in \mathbb{R}^n$. Set $k := 1$.
Step 2. Obtain an feasible initialization point x^1 from x_r^1.
Step 3. (Main Loop)
(a) Find the steepest descent direction by solving the following problem:

$$\min_{\lambda_j, \gamma_p} || \sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k) + \sum_{i \in I_\varepsilon(x^k)} \gamma_p^k \nabla \mathscr{G}_p(x^k) ||^2$$

$$\text{s.t: } \lambda_j \geq 0, \gamma_p \geq 0, \ \sum_{j=1}^{m} \lambda_j^k + \sum_{p \in I_\varepsilon(x^k)} \gamma_p^k = 1$$

where $I_\varepsilon(x^k) := \{i \in 1, \ldots, K | \mathscr{G}_p(x^k) \geq -\varepsilon\}$.
From this, we obtain the optimal solution:
$\lambda^{\mathbf{k}} = (\lambda_j^k)$, $\gamma^{\mathbf{k}} = (\gamma_p^k)$, $s(x^k) = -\left(\sum_{j=1}^m \lambda_j^k \nabla F_j(x^k) + \sum_{p \in I_\varepsilon(x)} \gamma_p^k \nabla \mathscr{G}_p(x^k)\right)$ and $\Theta(x^k) = -\frac{1}{2} ||s(x^k)||^2$.
(b) If $\Theta(x^k) = 0$, go to **Step 4**. Otherwise, continue to **Step 3(c).**
(c) Set $x_{temp} := x^k + \alpha_k s(x^k)$.
(d) Compute the step size:
Set $\eta_j^k := \lambda_j^k + \sum_{p \in I_\varepsilon(x)} \gamma_p^k$
If $\sum_{j=1}^m \eta_j^k F_j(x_{temp}) \leq \sum_{j=1}^m \eta_j^k F_j(x_{temp}) + \sigma \left\langle s(x^k), x_{temp} - x^k \right\rangle$
then $\alpha_{k+1} = \alpha_k$, otherwise set $\alpha_{k+1} := \kappa \alpha_k$
Step 3. Set $x^{k+1} := x^k + \alpha_{k+1} s(x^k), k := k+1$ and go to **Step 2.**
Step 4. Use the projection operator P_U, **and conclude** $x_{out} = P_U(x^k)$**.**

3.2.2 Solving problem ($MOP_{Cons}(F,U)$)

Version 1 of the proposed algorithm for solving constrained multi-objective optimization problems

We combine the steepest descent multi-objective method with adaptive step size and an initialization process to solve problem ($MOP_{Cons}(F,U)$), where $U := x \in \mathbb{R}^n | g_i(x) \leq 0, , \forall i = 1,\dots,l$ with differentiable constraint functions $g_i, , \forall i = 1,\dots,l$.

3.2.3 Initialization

Similar to Section 3.2.1, we propose an acceptable initialization process as follows:

- **Step 1:** Randomly initialize a point $x_0 \in \mathbb{R}^n$.
- **Step 2:** If x_0 is feasible, stop. Otherwise, go to **Step 3**.
- **Step 3:** Determine the set of tight constraints $I_\varepsilon(x_0) = \{i \mid g_i(x_0) \geq -\varepsilon\}$. We can find a descent direction d_r as follows:

$$(d_r, \alpha_r) = \arg \min_{d \in \mathbb{R}^n, \alpha \in R} \alpha + \frac{1}{2}\|d\|^2, \text{ s.t } g_i(x_0)^T d \leq \alpha, i \in I(x_0).$$

- **Step 4:** Update $x_0 = P_U(x_0 + \eta_r d_r)$ and go back to **Step 2**.

After obtaining a feasible starting point, we proceed with the main algorithm as follows:

Algorithm 3 Steepest multi-gradient with adaptive size version 1

```
Step 1. Set $\kappa \in [0,1], \sigma \in [0,1], \alpha_1 \in (0,+\infty)$, and $x_r^1 \in \mathbb{R}^n$. Set $k := 1$.
Step 2. Find an acceptable initial point $x^1$ from $x_r^1$.
Step 3. (Main loop)
(a) Find the descent direction by solving the following problem:
        $\min_{\lambda_j,\beta_i} || \sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k) + \sum_{i \in I_\varepsilon(x^k)} \beta_i^k \nabla g_i(x^k) ||^2$
        s.t: $\lambda_j \geq 0, \beta_i \geq 0, \sum_{j=1}^{m} \lambda_j^k + \sum_{i \in I_\varepsilon(x^k)} \beta_i^k = 1$
    where $I_\varepsilon(x^k) = \{i \in 1,\dots,l \mid g_i(x^k) \geq -\varepsilon\}$.
    From this, we determine the optimal solution:
    $\lambda^{\mathbf{k}} = (\lambda_j^k)$, $\beta^{\mathbf{k}} = (\beta_i^k)$, $s(x^k) = -\left(\sum_{j=1}^m \lambda_j^k \nabla F_j(x^k) + \sum_{i \in I_\varepsilon(x^k)} \beta_i^k \nabla g_i(x^k)\right)$
    and $\Theta(x^k) = -\frac{1}{2}||s(x^k)||^2$.
(b) If $\Theta(x^k) = 0$, go to Step 5. Otherwise, continue with Step 3(c).
(c) Set $x_{temp} := x^k + \alpha_k s(x^k)$.
(d) Compute the step size:
    If $\sum_{j=1}^{m} \lambda_j^k F_j(x_{temp}) + \sum_{i \in I_\varepsilon(x^k)} \beta_i^k g_i(x^k) \leq \sum_{j=1}^{m} \lambda_j^k F_j(x^k) + \sum_{i \in I_\varepsilon(x^k)} \beta_i^k g_i(x^k) + \sigma \left\langle s(x^k), x_{temp} - x^k \right\rangle$
    then $\alpha_{k+1} = \alpha_k$, otherwise set $\alpha_{k+1} := \kappa \alpha_k$.
Step 4. Set $x^{k+1} := x^k + \alpha_{k+1} s(x^k), k := k+1$ and go to Step 3.
Step 5. Use the projection operator $P_U$ to obtain the solution $x_{out} = P_U(x^k)$.
```

3.2.4 Proof of convergence

For $x \in U$, we define

$$H(x) = \sum_{j=1}^{m} \lambda_j F_j(x) + \sum_{i \in I_\varepsilon(x)} \beta_i^k g_i(x).$$

So, we have:

$$H(x^k) = \sum_{j=1}^{m} \lambda_j^k F_j(x^k) + \sum_{i \in I_\varepsilon(x^k)} \beta_i^k g_i(x^k)$$

$$s(x^k) = -\nabla H(x^k) = -\sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k) - \sum_{i \in I_\varepsilon(x^k)} \beta_i^k \nabla g_i(x^k).$$

Version 2 of the proposed algorithm for solving constrained multi-objective optimization problems

3.2.5 Theoretical foundation of the algorithm

Similar to Section 3.2.1, we propose adding inequality constraints $\mathscr{G}_p$ to separate the image space and find an effective initial point before solving problem $(MOP_{Cons}(F,U))$. Then, the problem we need to solve is formulated as follows:

$$\begin{aligned} &\min_{x \in U} F(x) = (F_1(x), F_2(x), \cdots, F_m(x)) \\ &\text{s.t}\, \mathscr{G}_p(x) = \langle u_p - u_k, F(x) \rangle \leq 0, \forall p = 1, \ldots, K, \end{aligned}$$

We will add constraints $\mathscr{G}_p, p = 1, \ldots, K$ to the feasible set U (defined in Section 3.2).

3.2.6 Finding an initial point

Similar to Section 3.2.1, we propose the following approach to finding an acceptable initial point:

- **Step 1:** Randomly initialize a point $x_0 \in \mathbb{R}^n$.
- **Step 2:** If x_0 is a feasible point, stop. Otherwise, go to **Step 3**.
- **Step 3:** We determine the set of regular constraints $I_\varepsilon(x_0) = \{i, p \mid \mathscr{G}_p(x_0) \geq 0, p = 1, \ldots, K \text{ and } g_i(x) \geq -\varepsilon\}$. We can find a descent direction d_r as follows:

$$\begin{aligned} (d_r, \alpha_r) = \arg &\min_{d \in \mathbb{R}^n, \alpha \in R} \alpha + \frac{1}{2}\|d\|^2, \\ &\text{s.t}\, \nabla \mathscr{G}_p(x_0)^T d \leq \alpha, p \in I(x_0) \\ &\nabla g_i(x_0)^T d \leq \alpha, i \in I(x_0) \end{aligned}$$

- **Step 4:** Update $x_0 = P_U(x_0 + \eta_r d_r)$ and go back to **Step 2**.

After finding an effective initial point, we proceed to the main algorithm

Algorithm 4 Steepest multi-gradient with adaptive step size version 2

```
Step 1. Set $\kappa \in [0,1], \sigma \in [0,1], \alpha_1 \in (0,+\infty)$, and $x^1 \in \mathbb{R}^n$. Let $k := 1$.
Step 2. (Main loop)
(a) Find the descent direction by solving the following problem:
    $\min_{\lambda_j,\beta_i,\gamma_p} ||\sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k) + \sum_{i\in I_\varepsilon(x^k)} \beta_i^k \nabla g_i(x^k) + \sum_{p\in I_\varepsilon(x^k)} \gamma_p^k \nabla \mathscr{G}_p(x^k)||^2$
    s.t: $\lambda_j \geq 0, \beta_i \geq 0, \gamma_p \geq 0,\ \sum_{j=1}^{m} \lambda_j^k + \sum_{i\in I_\varepsilon(x^k)} \beta_i^k + \sum_{p\in I_\varepsilon(x^k)} \gamma_p^k = 1$
where $I_\varepsilon(x^k) := i \in 1,\dots,l; p \in 1,\dots,K \mid \mathscr{G}_p\left(x^k\right) \geq 0$ and $g_i(x^k) \geq -\varepsilon$.
From this, the optimal solution is obtained:
$\lambda^{\mathbf{k}} = (\lambda_j^k),\ \beta^{\mathbf{k}} = (\beta_i^k),\ s(x^k) = -\left(\sum_{j=1}^{m} \lambda_j^k \nabla F_j(x^k) + \sum_{i\in I_\varepsilon(x^k)} \beta_i^k \nabla g_i(x^k) + \sum_{p\in I_\varepsilon(x^k)} \gamma_p^k \nabla \mathscr{G}_p(x^k)\right)$
and $\Theta(x^k) = -\frac{1}{2}||s(x^k)||^2$.
(b) If $\Theta(x^k) = 0$, go to Step 4. Otherwise, continue to Step 2(c).
(c) Set $x_{temp} := x^k + \alpha_k s(x^k)$.
(d) Compute the step size:
Let $\eta_j^k := \lambda_j^k + \sum_{p\in I_\varepsilon(x)} \gamma_p^k$
If $\sum_{j=1}^{m} \eta_j^k F_j(x_{temp}) + \sum_{i\in I_\varepsilon(x^k)} \beta_i^k g_i(x^k) \leq \sum_{j=1}^{m} \eta_j^k F_j(x^k) + \sum_{i\in I_\varepsilon(x^k)} \beta_i^k g_i(x^k) + \sigma \left\langle s(x^k), x_{temp} - x^k \right\rangle$
then $\alpha_{k+1} = \alpha_k$, else $\alpha_{k+1} := \kappa \alpha_k$.
Step 3. Set $x^{k+1} := x^k + \alpha_{k+1} s(x^k)$, $k := k+1$ and go back to Step 2.
Step 4. Use the projection operator $P_U$, output the solution $x_{out} = P_U(x^k)$.
```

3.3 Proposed algorithm for multitask learning

Similar to [2], we use K reference vectors $\mathbf{u_1}, \mathbf{u_2}, \dots, \mathbf{u_K}$ to represent the trade-off between the priority of tasks in MTL. We also use linear inequality constraints $\mathscr{G}_p(\theta_t) = \langle u_p - u_k, \mathscr{L}(\theta_t)\rangle \leq 0, \forall p = 1,\dots,K$ to partition the image space into sub-problems. Then, we propose using an adaptive step size to update the learning rate of the neural network. The proposed algorithm for MTL is presented in detail below.

Algorithm 5 Proposed algorithm for MTL

```
Input: Set of reference vectors $\{\mathbf{u_1}, \mathbf{u_2}, \dots, \mathbf{u_K}\}$.
 1: Set $\kappa \in [0,1], \sigma \in [0,1], \alpha_1 \in (0,+\infty)$.
 2: for k = 1 to K do
 3:     Initialize hyperparameters $\theta_r^k$ for the neural network.
 4:     Find effective hyperparameters $\theta_0^k$ from $\theta_r^k$ using descent methods.
 5:     for t =1 to T do
 6:         Find the descent direction by solving the following problem:
            $$\min_{\lambda_j,\gamma_p} || \sum_{j=1}^{m} \lambda_j^k \nabla \mathscr{L}_j(\theta^k)^T + \sum_{p \in I_\varepsilon(x^k)} \gamma_p^k \nabla \mathscr{G}_p(x^k)||^2$$
            $$\text{s.t: } \lambda_j \geq 0, \gamma_p \geq 0, \sum_{j=1}^{m} \lambda_j^k + \sum_{p \in I_\varepsilon(x^k)} \gamma_p^k = 1$$
 7:         Where $I_\varepsilon(\theta^k) := \{p \in 1,\dots,K | \mathscr{G}_p(\theta^k) \geq -\varepsilon\}$.
 8:         Determine the descent direction as : $s(\theta^k) = -\left(\sum_{j=1}^{m} \lambda^k j \nabla \mathscr{L} j(\theta^k) + \sum_{p \in I_\varepsilon(x^k)} \gamma_p^k \nabla \mathscr{G}_p(x^k)\right)$.
 9:         Set $\theta_{temp} := \theta^k + \alpha_k s(\theta^k)$.
10:         Compute the step size:
11:         Set $\eta_j^k := \lambda_j^k + \sum_{p \in I_\varepsilon(\theta)} \gamma_p^k$
12:         If $\sum_{j=1}^m \eta_j^k \mathscr{L}_j(\theta_{temp}) \leq \sum_{j=1}^m \eta_j^k \mathscr{L}_j(\theta^k) + \sigma \left\langle s(\theta^k), \theta_{temp} - \theta^k \right\rangle$
13:         then $\alpha_{k+1} = \alpha_k$, else $\alpha_{k+1} := \kappa \alpha_k$.
14:         Update hyper parameter $\theta^{k+1} := \theta^k + \alpha_{k+1} s(\theta^k)$, $k := k+1$.
15:     end for
16: end for
Output: The optimized hyperparameters for each sub-problem $\{\theta_T^k | k = 1,\dots,K\}$.
```

4 Numerical experiments

Example 1 (Toy example (Lin et al [2])) Consider the non-convex unconstrained multi-objective optimization problem:

$$\begin{aligned} &\text{Min } F(x) = \left\{ 1 - \exp^{-\sum_{i=1}^{d}\left(x_i - \frac{1}{d}\right)^2}, 1 - \exp^{-\sum_{i=1}^{d}\left(x_i + \frac{1}{d}\right)^2} \right\} \\ &\text{s.t: } x \in \mathbb{R}^2 \end{aligned}$$

We conducted experiments with $d = 20$. We apply Algorithm 1 and 2 to solve the above problem with 10 initial data points, and we obtain the following results:

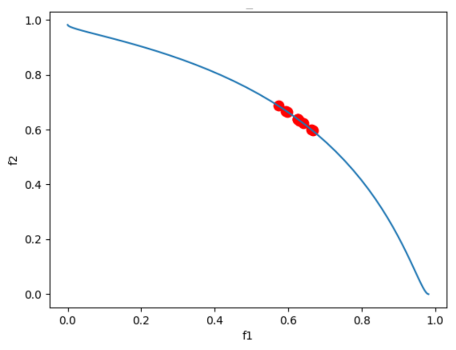


Fig. 1 Result of Algo 1

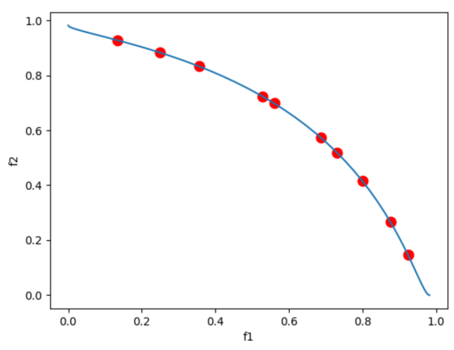
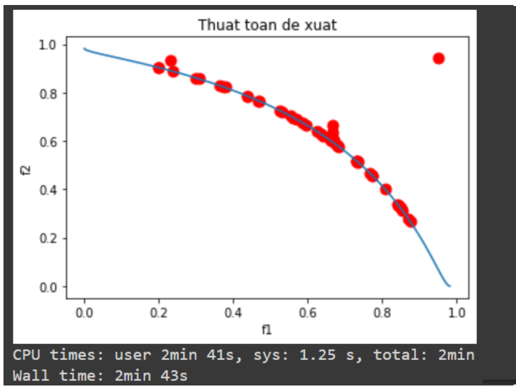


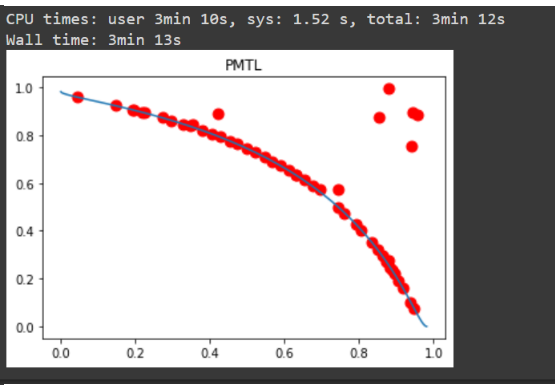
Fig. 2 Result of Algo 2

The solutions obtained by Algorithm 1 concentrate in a region on the Pareto Front, while using Algorithm 2, we obtain solutions that are all on the Pareto Front and spread evenly across the surface. This is an improvement compared to version 1, or the MGDA algorithm, where the solutions found only concentrate in a region on the Pareto Front [2]. Comparing the running time of algorithm 2 with the PMTL algorithm [2] and set with the maximum number of iterations to solve example is 50, we obtain the following results:



```
CPU times: user 2min 41s, sys: 1.25 s, total: 2min
Wall time: 2min 43s
```

Fig. 3 The time to solve the problem of Algorithm 2



```
CPU times: user 3min 10s, sys: 1.52 s, total: 3min 12s
Wall time: 3min 13s
```

Fig. 4 The time to solve the problem of Algorithm PMTL

We can see that the proposed algorithm has a faster running time than the PMTL algorithm (2 minutes and 43 seconds compared to 3 minutes and 13 seconds), and at the same time, the obtained solutions converge better to the Pareto Front than the PMTL algorithm.

Example 2 (Andreani et al [**?**]) Consider a convex multi-objective optimization problem without constraints:

$$\text{Min } F(x) = \left\{ \frac{1}{25}x_1^2 + \frac{1}{100}\left(x_2 - \frac{9}{2}\right)^2, \frac{1}{25}x_2^2 + \frac{1}{100}\left(x_1 - \frac{9}{2}\right)^2 \right\}$$
$$\text{s.t: } x \in \mathbb{R}^n$$

The results obtained by using Algorithm 1 and Algorithm 2 to solve Example 2 with 10 initialization points are as follows:

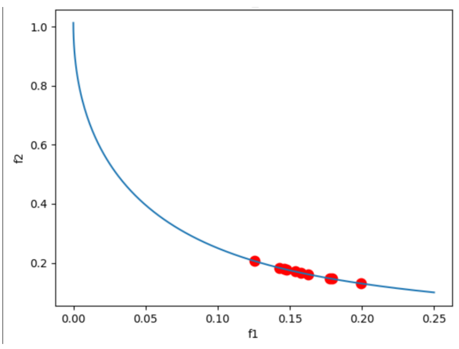


Fig. 5 Result of Algo 1

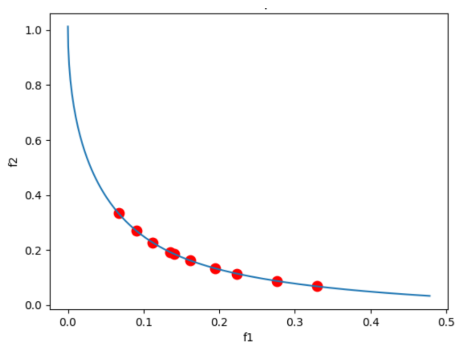
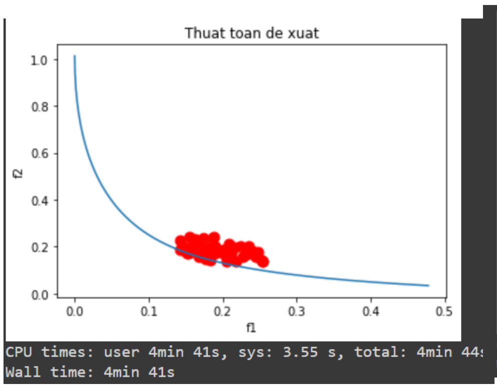


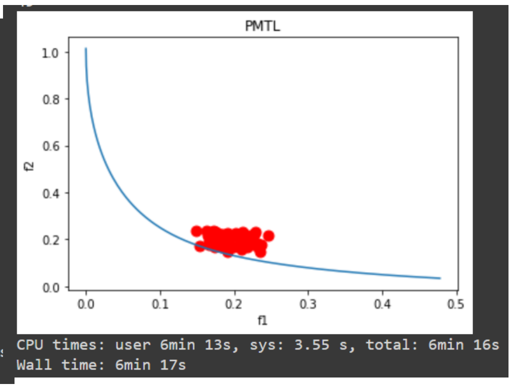
Fig. 6 Result of Algorithm 2

For Algorithm 1, we can see that the obtained Pareto solutions are concentrated in a region on the Pareto Front. This drawback has been overcome by using constraints to divide the image space in Algorithm 2. Comparing the running time of Algorithm 2 with the PMTL algorithm [2] with the maximum number of iterations to solve example is 50, we obtain the following results:



```
CPU times: user 4min 41s, sys: 3.55 s, total: 4min 44
Wall time: 4min 41s
```

Fig. 7 The time to solve example 2 of Algorithm 2



```
CPU times: user 6min 13s, sys: 3.55 s, total: 6min 16s
Wall time: 6min 17s
```

Fig. 8 The time to solve example 2 of Algorithm PMTL

We can see that the proposed algorithm has a faster running time compared to the PMTL algorithm (4 minutes and 41 seconds versus 6 minutes and 17 seconds), and the obtained solutions converge better to the Pareto Front compared to the PMTL algorithm.

In summary, the solutions found by the proposed algorithm converge better to the Pareto Front than PMTL. The proposed algorithm can be seen as an improved version of PMTL with the ability to search for Pareto solutions more evenly and faster on unconstrained multi-objective optimization problems.

Example 3 We design a convex constrained multi-objective optimization problem as follows:

$$\begin{aligned}&\text{Min } F(x) = \{\frac{x_1^2 + x_2^2}{50}, \frac{(x_1-5)^2 + (x_2-5)^2}{50}\}\\&\text{s.t: } x_i \in [0,5],\ i = 1,2\end{aligned}$$

The test result of algorithm 4 with 10 randomly initialized points for the Pareto solutions obtained as follows:

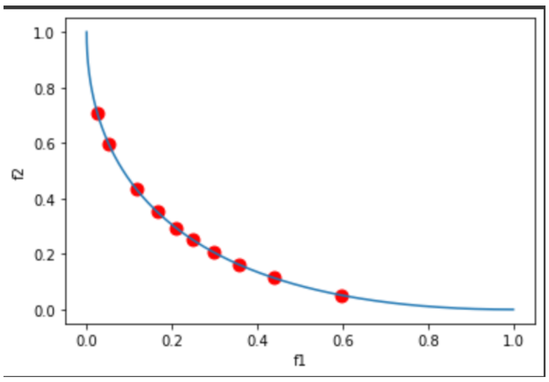


Fig. 9 Result of Algorithm 4 solving Problem 3

With the Pareto Front shown in blue, we can see that the obtained solutions are all on the Pareto Front and are evenly distributed over the entire surface.

Example 4 We consider the following non-convex constrained multi-objective optimization problem:

$$\begin{aligned}&\text{Min } F(x) = \{\frac{x_1^2 + x_2^2 + 3}{1 + 2x_1 + 8x_2}, \frac{x_2^2 + x_1^2 + 3}{1 + 2x_2 + 8x_1}\}\\&\text{s.t} \quad x \in U,\end{aligned}$$

consider $U = \{x = (x_1, x_2)^\top \in \mathbb{R}^2 | g_1(x) = -x_1^2 - 2x_1x_2 \leq -4;\ x_1, x_2 \geq 0\}$.

We perform the experiment with Algorithm 3 and Algorithm 4 using 20 random initial points, and we obtain the following results:

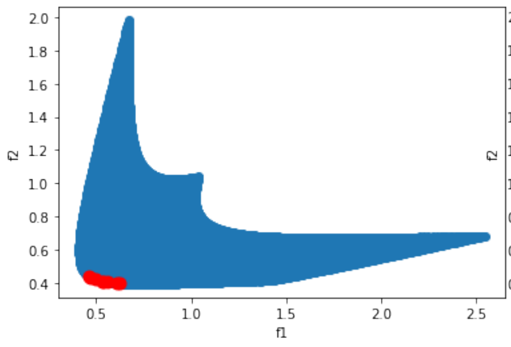


Fig. 10 Result of solving Problem 4 of Algorithm 3

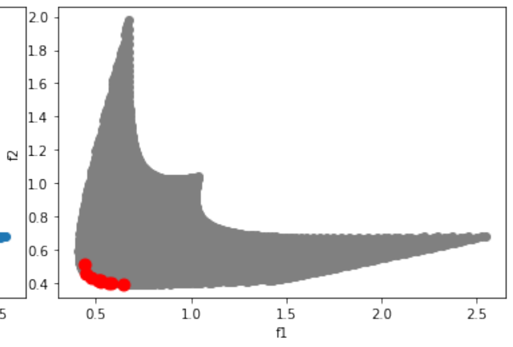


Fig. 11 Result of solving Problem 4 of Algorithm 4

We can see that both algorithms are able to find solutions on the Pareto Front, but with the linear constraints that help divide the image space, algorithm 4 finds solutions that are more evenly spread out on the Pareto Front.

Example 5 Consider a convex multi-objective optimization problem with the following fractional objective function and constraints:

$$\operatorname{Min} F(x) = \{\frac{x_1+1}{-x_1^2+3x_1-x_2^2+3x_2+3.50}, \frac{x_1^2-2x_1+x_2^2-8x_2+20.00}{x_2}\}$$

$$\text{s.t } x_1, x_2 \in U,$$

consider $U = \{2x_1+x_2 \leq 6, 3x_1+x_2 \leq 8, x_1-x_2 \leq 1, x_1, x_2 \geq 1\}$

The result of testing algorithm 4 with 20 randomly initialized points for the Pareto solution is shown as follows:

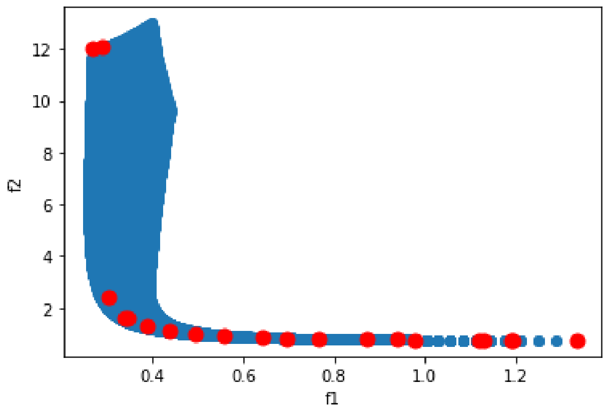


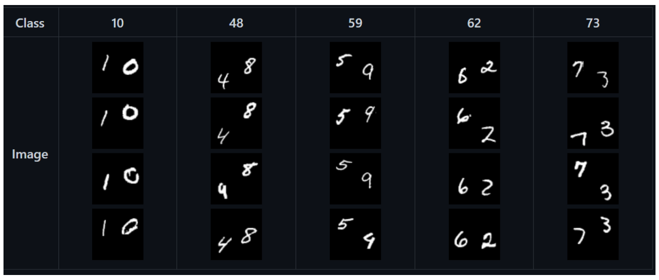
Fig. 12 Result of solving Problem 5 of Algorithm 4

With the blue Pareto Front line in the figure, we can see that most of the solutions found belong to the Pareto Front, and the other points are close to the Pareto Front.

5 Applications to multi-task learning

In this section, we will implement the proposed algorithm and evaluate its effectiveness on a two-task image classification problem using the Multi-MNIST dataset provided at [2]. At the same time, we will compare the effectiveness of the proposed algorithm with the PMTL algorithm.

Multi-MNIST is a dataset constructed by combining two digits from the MNIST dataset [**?**] into one image. The image classification MTL problem will simultaneously learn two loss functions representing the prediction of two digits, one in the top left corner and the other in the bottom right corner. An illustration of the dataset is shown below.



We construct a LeNet model similar to the one in [**?**] with the following layers:

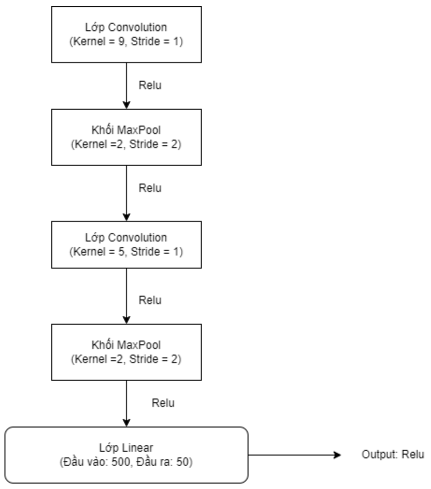

Lớp Convolution (Kernel = 9, Stride = 1)
Relu
Khối MaxPool (Kernel =2, Stride = 2)
Relu
Lớp Convolution (Kernel = 5, Stride = 1)
Relu
Khối MaxPool (Kernel =2, Stride = 2)
Relu
Lớp Linear (Đầu vào: 500, Đầu ra: 50)
Output: Relu

Fig. 13 Implementation of Model

We carry out the training process with 5 reference vectors $[\cos(\frac{k\pi}{2K}), \sin(\frac{k\pi}{2K})], k = 0,1,2,3,4$ for every 50 epochs. We illustrate the results with the reference vector $u_1 = (1,0)$ as follows:

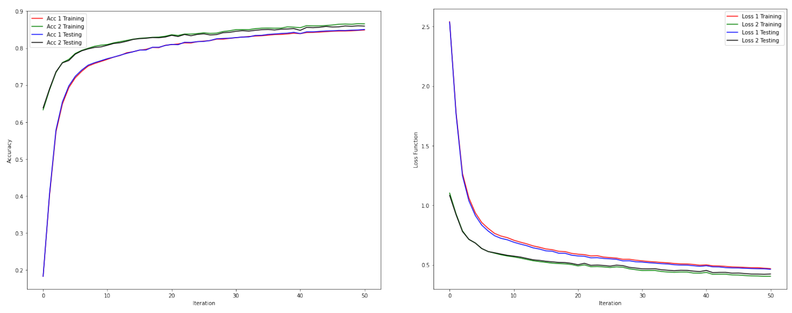


Fig. 14 The test result

The experimental results show that both image classification tasks are learned simultaneously and relatively effectively. The table comparing the accuracy values on the Test set between the proposed algorithm and PMTL will be illustrated below:

Vector preference	Accuracy Task 1 Proposal	Accuracy Task 2 Proposal	Accuracy Task 1 PMTL	Accuracy Task 2 PMTL
u_1	$\mathbf{0.85 \pm 0.05}$	0.87 ± 0.05	0.84 ± 0.05	0.87 ± 0.05
u_2	0.88 ± 0.05	0.88 ± 0.05	0.88 ± 0.05	$\mathbf{0.89 \pm 0.05}$
u_3	0.90 ± 0.05	0.88 ± 0.05	$\mathbf{0.91 \pm 0.05}$	0.88 ± 0.05
u_4	0.90 ± 0.05	$\mathbf{0.88 \pm 0.05}$	$\mathbf{0.91 \pm 0.05}$	0.87 ± 0.05
u_5	0.91 ± 0.05	0.84 ± 0.05	$\mathbf{0.92 \pm 0.05}$	$\mathbf{0.86 \pm 0.05}$

We can see that the effectiveness of the proposed algorithm is quite similar to the PMTL algorithm. The comparison of the training process between the proposed algorithm (blue line for task 1 and black line for task 2) and the PMTL algorithm (red line for task 1 and green line for task 2) for the reference vector $\mathbf{u_1}$ is shown in the following image:

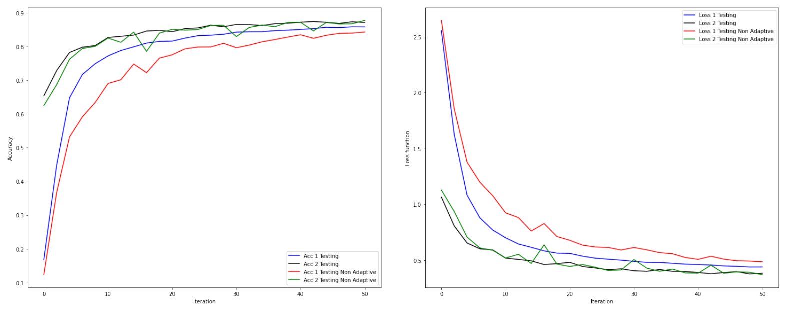


Fig. 15 Compare the proposed algorithm and PMTL

The proposed algorithm and PMTL algorithm are two multi-task learning methods for image classification tasks. The experimental results show that both algorithms can learn two tasks simultaneously with relatively similar effectiveness, as seen in the comparison of accuracy values on the Test set. However, in terms of the training process, the proposed algorithm has an advantage over PMTL in that it uses an Adaptive step size, which leads to a more consistent decrease in the loss function and therefore a continuous increase in accuracy during training. Meanwhile, PMTL uses a fixed step size, which may lead to slower convergence or oscillation during training. In summary, while both algorithms can achieve similar performance in multi-task learning for image classification, the proposed algorithm has the advantage of a more stable and consistent training process thanks to its Adaptive step size.

6 Conclusion

We proposed a novel easy adaptive step-size process in a wide family of solution methods for optimization problems with non-convex objective functions. This approach does not need any line-searching or prior knowledge, but rather takes into consideration the iteration sequence's behavior. As a result, as compared to descending line-search approaches, it significantly reduces the implementation cost of each iteration. We demonstrated technique convergence under simple assumptions. We demonstrated that this new process produces a generic foundation for optimization methods. The preliminary results of computer experiments demonstrated the new procedure's efficacy.

References

1. Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in hilbert spaces. Springer, New York (2011)
2. Lin, Xi and Zhen, Hui-Ling and Li, Zhenhua and Zhang, Qing-Fu and Kwong, Sam: Pareto multi-task learning, Advances in neural information processing systems (2019)
3. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2009)
4. Rockafellar, R.T.: Convex analysis. Princeton University Press, Princeton, New Jersey (1970)
5. W. Bian, L. Ma, S. Qin, X. Xue: Neural network for nonsmooth pseudoconvex optimization with general convex constraints, Neural Networks 101, 1-14 (2018).
6. Cevher, V., Becker, S., Schmidt, M.: Convex optimization for big data. Signal Process. Magaz. 31, 32–43 (2014).
7. Fliege and Svaiter, Benar Fux: Steepest descent methods for multicriteria optimization, Mathematical methods of operations research (2000).
8. J.E. Dennis, R.B. Schnabel: Numerical methods for unconstrained optimization and nonlinear equations, Prentice-Hall, New Jersey, 1983.
9. O. P. Ferreira, W. S. Sosa, On the Frank–Wolfe algorithm for non-compact constrained optimization problems, Optimization, 71:1, 197-211 (2022).
10. Y. Hu , J. Li, C. K. Yu, Convergence Rates of Subgradient Methods for Quasiconvex Optimization Problems, Computational Optimization and Applications, 77, 183–212 (2020).
11. K. C. Kiwiel, Convergence and efficiency of subgradient methods for quasiconvex minimization, Math. Program., Ser. A 90: 1–25 (2001).
12. I. V. Konnov, Simplified versions of the conditional gradient method, Optimization, 67(12), 2275-2290 (2018).
13. Guanghui Lan, First-order and Stochastic Optimization Methods for Machine Learning, Springer Series in the Data Sciences, Springer Nature Switzerland (2020)
14. N. Liu, J. Wang, S. Qin: A one-layer recurrent neural network for nonsmooth pseudoconvex optimization with quasiconvex inequality and affine equality constraints, Neural Networks 147, 1-9 (2022).
15. Yura Malitsky, Konstantin Mishchenko, Adaptive Gradient Descent without Descent, Proceedings of Machine Learning Research, 119:6702-6712 (2020).
16. O. Mangasarian, Pseudo-convex functions, Siam Control, 8, 281-290 (1965)
17. Y. Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2013.

18. Xu, H.K.: Iterative algorithms for nonlinear operators. J. London Math. Soc. 66, 240-256 (2002)
19. C.K. Yu, Y. Hu, X. Yang & S. K. Choy, Abstract convergence theorem for quasi-convex optimization problems with applications, Optimization, 68(7), 1289-1304, 2019.
20. Yura Malitsky, Konstantin Mishchenko, Adaptive Gradient Descent without Descent, Proceedings of Machine Learning Research, 119:6702-6712 (2020).
21. O. Mangasarian, Pseudo-convex functions, Siam Control, 8, 281-290 (1965)
22. Y. Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2013.
23. Xu, H.K.: Iterative algorithms for nonlinear operators. J. London Math. Soc. 66, 240-256 (2002)
24. C.K. Yu, Y. Hu, X. Yang & S. K. Choy, Abstract convergence theorem for quasi-convex optimization problems with applications, Optimization, 68(7), 1289-1304, 2019.