Steepest descent method using novel adaptive stepsizes for unconstrained nonlinear multiobjective programming

Pham Thi Hoai[a,*], Hoang Thi Hai Yen[a]

[a]*Faculty of Mathematics and Informatics, Hanoi University of Science and Technology, 1 Dai Co Viet Road, Hanoi, Vietnam*

Abstract

We propose new adaptive strategies to compute stepsizes for the steepest descent method to solve unconstrained nonlinear multiobjective optimization problems without employing any line search procedure. The resulting algorithms can be applied to a wide class of nonconvex unconstrained multi-criteria optimization problems satisfying a global Lipschitz continuity condition imposed on the gradients of all objectives. In a general setting, we prove that from some fixed iteration onwards, the sequences of our stepsizes are increasing to a positive number, and the sequences of the objective values are decreasing. Any accumulation point of the iterates is proved to be a Pareto critical point. In the convex setting of the objective functions, the iterates generated by our algorithms are proved to converge to a weakly efficient point. Additionally, we establish sublinear and linear convergence rates for the convex and strongly convex cases, respectively. To the best of our knowledge, there have been analyses of the global convergence rate for steepest descent algorithms with backtracking line search-based stepsizes; however, there have not been any for adaptive stepsizes proposed in the literature. Notably, although the global Lipschitz gradient condition is used for analyzing the convergence properties of our new methods, the computation of our new algorithms themselves do not require calculating or estimating the global Lipschitz constants of the gradients. This and the absence of line search backtracking procedures reduce the processing time significantly. The advantages of our new algorithms are further demonstrated by numerical experiments for various test instances when compared to recent methods.

Keywords: Nonconvex multiobjective programming, steepest descent method, gradient descent method, Lipschitz gradient, convex multi-objective programming

1. Introduction

Multiobjective programming has been received a lot of attention of researchers for a long time because of its applicative ability in many areas such as economics, engineering, environment, multitask learning, etc, (see e.g., Evans, 1984; Gravel et al., 1992; Fliege & Werner, 2014; Tapia & Coello, 2007; Marler & Arora, 2004; Leschine et al., 1992; Sener & Koltun, 2018). In this paper, we will consider

*Corresponding author

Email addresses: hoai.phamthi@hust.edu.vn (Pham Thi Hoai), hoangyen2002nb@gmail.com (Hoang Thi Hai Yen)

one of the typical problems in multiobjective programming that is unconstrained multiobjective optimization problems as follows

$$\min_{x\in\mathbb{R}^n} F(x) = (f_1(x), f_2(x), ..., f_m(x)), \tag{1.1}$$

where $f_i(x) : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$ are m objective functions. Unlike single objective optimization problems, multiobjective optimization problems concern finding optimal solutions for several objective functions simultaneously. The concept of their optimal solutions therefore changes adaptively with the order relation used for the multidimensional outcome space $\mathbb{R}^m$. In general, the order relation of the outcome space is induced by some convex cone and typically the nonnegative orthant cone $\mathbb{R}^m_+$. And throughout this paper, we will use the closed convex cone $\mathbb{R}^m_+$ for defining the order relation in the outcome space $\mathbb{R}^m$ as well as the convexity (strong convexity, resp.) of function F. Hence to solve Problem (1.1), we desire to find a vector $x^* \in \mathbb{R}^n$ called a *Pareto optimal solution* or an *efficient solution* of Problem (1.1) such that there does not exist $x \in \mathbb{R}^n$ satisfying $F(x) \leq F(x^*)$ and $F(x) \neq F(x^*)$. This definition can be relaxed to be *weak Pareto optimal solution* or *weakly efficient solution* if there does not exist $x \in \mathbb{R}^n$ satisfying $F(x) < F(x^*)$. The set of all efficient (weakly efficient, resp.) optimal solutions of Problem (1.1) is called the efficient (weakly efficient, resp.) set of Problem (1.1).

To utilize the rich results of mathematical programming for the single objective case, one of the classical methods for solving Problem (1.1) is based on scalarization technique, see e.g., Ehrgott (2005). It means that we try to find the way to transform Problem (1.1) to be an optimization problem with only one objective. However, this approach commonly requires choosing parameters used for the process of scalarizing. And "for some problems, this choice can be problematic", as commented in Fliege et al. (2009b)[Section 7]. Therefore, one can be interested in parameter-free methods to solve Problem (1.1). Among them, *the steepest descent direction* approach is one of the most preferred methods (see e.g., Fliege et al. (2019)). This method raises the idea of the gradient descent method used for the case $m = 1$. Its advantage is firstly stated in the simple presentation as follows

$$x^{k+1} = x^k + t_k d^k, \tag{1.2}$$

where $t_k > 0$ is called *stepsize* and d^k is the steepest descent direction of F at x^k which defined by the unique minimizer of the convex optimization problem:

$$\min_{d\in\mathbb{R}^n} \max_{i=1,...,m} \left\{ \left\langle \nabla f_i(x^k), d \right\rangle + \frac{1}{2}\|d\|^2 \right\}. \tag{1.3}$$

Thus, this method does not require any predetermined parameters as other scalarization-based methods (see e.g., Fliege & Svaiter, 2000; Fliege et al., 2019; Morovati et al., 2016; Chen et al., 2023 for details). Note that when $m = 1$ the method (1.2) becomes *gradient descent scheme* - a traditional approach for solving unconstrained nonlinear optimization problems. And similarly to the single-objective case, we can control the performance of the **steepest descent** method (1.2) by choosing an

appropriate strategy of selecting the stepsize t_k.

In the literature, line search methods using backtracking procedures have been utilized commonly to create t_k since this way ensures the decreasing of all objectives simultaneously after each iteration. In particular, Fliege & Svaiter (2000) used Armijo backtracking procedure to compute the step length t_k. The obtained iterates $\{x^k\}$ are then proved to have at least one accumulation being a Pareto critical point of Problem (1.1) with nonconvex objective F satisfying the boundedness assumption on the lower level set of F. After that, in 2019, Fliege et al. (2019) continued studying the convergence rate of this method and obtained analogous results as gradient descent algorithm for single-objective case. Precisely, if all objectives are supposed to have global Lipschitz gradients and at least one objective is bounded from below then the computational complexity $O(1/\sqrt{k})$ is proved for $\min\limits_{1\leq i\leq k-1} \|d^i\|$. In the cases of convex and strongly convex F, the authors showed the sublinear and linear convergence rates of their proposed method, respectively. Other line search backtracking methods applied for steepest descent scheme (1.2) can be found in Mita et al. (2019). However, a common challenge of line search procedures is due to the expensive computation of backtracking calculations. Therefore, adaptive strategies may overcome this drawback. In 2016, utilizing Barzilai-Borwein method (see Barzilai & Borwein (1988)) which was created for single-objective cases, Morovati et al. (2016) proposed BBMO algorithm which uses an adaptive stepsize selection for steepest descent scheme (1.2). The local convergence of BBMO was analyzed when the objectives assumed to be twice continuously differentiable.

Very recently, Barzilai-Borwein strategy has been exploited again by Chen et al. (2023) for steepest descent method (1.2). The obtained algorithm was named as BBDMO. Unlike BBMO, BBDMO uses the idea of Barzilai-Borwein method in tuning the descent direction. More precisely, the gradient magnitudes are dynamically tuned with some multipliers generated by Barzilai-Borwein ratios before using in direction-finding subproblem (1.3). It is also worth noting by Chen et al. (2023) and the references therein that the line search stepsize strategies "can lead to a relatively small stepsize", especially in the imbalance situation of objectives. Therefore, although, the stepsize choices used in BBDMO are line search methods but their new proposed direction BBDMO can overcome the imbalance of the objectives and hence can reduce the backtracking computation. This advantage is not only able to decrease the running time but is also capable of giving bigger stepsizes. The convergence of BBDMO is then considered under the strictly and strongly convex assumptions on F. It is worth noting that, the study on the convergence rate was not given for both BBMO and BBDMO.

Contributions: In this paper, by utilizing the recent adaptive stepsize NGD proposed by Hoai et al. (2024) for single-objective unconstrained nonlinear programming, we propose some new adaptive strategies of choosing t_k used in (1.2) for solving Problem (1.1). Our new stepsizes can be quickly computed by explicit formulas without using any line search backtracking procedures. Moreover, we highlight that:

(i) From some fixed iteration $\hat{k}$, the sequence of stepsizes is increasing to a finite limit.

(ii) Our new methods are proved to be descent after the iteration $\hat{k}$.

(iii) If the objectives have global Lipschitz gradients and at least one objective is lower bounded, we prove that $\{x^k\}$ is bounded and any accumulation point of $\{x^k\}$ is a Pareto critical point. We also prove the computational complexity $O(1/\sqrt{K})$ of $\min\limits_{\hat{k}\leq i\leq K-1}\|d^i\|$ for a class of nonconvex objectives which include convex, concave and indefinite quadratic functions.

(iv) Additionally, if we complement the convex or strongly convex condition on F then we can establish the sublinear or linear convergence rates of our proposed steepest descent algorithms.

To the best of our knowledge, the four properties mentioned above have not been provided for the other *adaptive* strategies of stepsize selection applied for steepest descent scheme to solve Problem (1.1) in the literature. The efficiency of our new method is also demonstrated by computational experiments for numerous examples in comparison with the recent methods.

The rest of the paper is organized as follows. After giving some preliminaries in Section 2, we propose four strategies for finding adaptive stepsizes and present the corresponding steepest descent algorithms in Section 3. The convergence results for the nonconvex and convex cases are established in Section 3 and Section 4, respectively. We then present the numerical results in Section 5 and close the paper with some conclusions in Section 6.

2. Preliminaries

In this paper, we denote $\mathbb{R}_+$ and $\mathbb{R}_{++}$ for the sets of nonnegative and positive real numbers, respectively. The notations $\mathbb{R}^m_+ := \mathbb{R}_+ \times \cdots \times \mathbb{R}_+$ and $\mathbb{R}^m_{++} := \mathbb{R}_{++} \times \cdots \times \mathbb{R}_{++}$ correspond to the nonnegative orthant and the positive orthant of $\mathbb{R}^m$. We consider the partial order induced by $\mathbb{R}^m_+$: with $u, v \in \mathbb{R}^m, u \leq v$ if $v - u \in \mathbb{R}^m_+$, and a stronger relation: $u < v$ if $v - u \in \mathbb{R}^m_{++}$. In other words, $u \leq v$ means that $u_i \leq v_i$ for all $i = 1, \ldots, m$ and $u < v$ also represents $u_i < v_i$ for all $i = 1, \ldots, m$. The unit simplex in outcome space $\mathbb{R}^m$ is defined by

$$\Delta_m = \left\{ \lambda \in \mathbb{R}^m \mid \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0, i = 1, ..., m \right\}.$$

As usual, we use $JF(x) \in \mathbb{R}^{m\times n}$ for the Jacobi matrix of F at x with entries $JF(x))_{i,j} = \dfrac{\partial f_i(x)}{\partial x_j}$. We recall some important definitions in multiobjective optimization in the sequel.

Definition 2.1. (Fliege & Svaiter, 2000; Fliege et al., 2019; Fukuda & Drummond, 2014)

(i) A vector $x^* \in \mathbb{R}^n$ is called a *Pareto optimal* solution of Problem (1.1) if there does not exist $x \in \mathbb{R}^n$ such that $F(x) \leq F(x^*)$ and $F(x) \neq F(x^*)$.

(ii) A vector $x^* \in \mathbb{R}^n$ is called a *weakly Pareto optimal* solution of Problem (1.1) if there does not exist $x \in \mathbb{R}^n$ such that $F(x) < F(x^*)$.

(iii) A vector $x^* \in \mathbb{R}^n$ is called a *Pareto critical point* of Problem (1.1) if $range(JF(x^*)) \cap \left[-\mathbb{R}^m_{++}\right] = \emptyset$. where $range(JF(x^*)) = \{JF(x^*)v : v \in \mathbb{R}^n\}$. In other words, $x^* \in \mathbb{R}^n$ is a Pareto critical point of F if and only if, for all vectors $v \in \mathbb{R}^n$, we have $JF(x^*)v \not< 0$.

(iv) A vector $d \in \mathbb{R}^n$ is called a descent direction for F at x, if $JF(x)d \in -\mathbb{R}^m_{++}$.

Following by (Fliege & Svaiter, 2000; Fliege et al., 2019; Chen et al., 2023) and the references therein, for a fixed $x \in \mathbb{R}^n$, the *steepest descent direction* $d(x)$ of F at x is defined as the unique minimizer of the convex optimization problem:

$$\min_{d\in\mathbb{R}^n} \max_{i=1,\dots,m} \left\{ \langle \nabla f_i(x), d\rangle + \frac{1}{2}\|d\|^2 \right\}. \tag{2.1}$$

In other words,

$$d(x) = \arg\min_{d\in\mathbb{R}^n} \max_{i=1,\dots,m} \langle \nabla f_i(x), d\rangle + \frac{1}{2}\|d\|^2. \tag{2.2}$$

Denoting $\theta(x)$ as the optimal value of the subproblem (2.1) then

$$\theta(x) = \min_{d\in\mathbb{R}^n} \max_{i=1,\dots,m} \left\{ \langle \nabla f_i(x), d\rangle + \frac{1}{2}\|d\|^2 \right\} = \max_{i=1,\dots,m} \left\{ \langle \nabla f_i(x), d(x)\rangle + \frac{1}{2}\|d(x)\|^2 \right\}. \tag{2.3}$$

The following lemma plays an important role in designing steepest descent method for solving Problem (1.1).

Lemma 2.2 (Fliege & Svaiter, 2000, Lemma 1)**.** *For differentiable objective f_i, $i = 1, ..., m$, let $\theta(x)$ and $d(x)$ be the optimal value and the optimal solution of Problem (2.1), respectively. Then.*

(a) The following conditions are equivalent:

(i) The point x is Pareto critical;

(ii) $\theta(x) = 0$;

(iii) $d(x) = 0$.

(b) The following conditions are equivalent:

(i) The point x is non-critical Pareto;

(ii) $\theta(x) < 0$.

(c) The mappings $x \mapsto d(x)$ and $x \mapsto \theta(x)$ are continuous.

In the view of computational aspect, one can find $d(x)$ by solving an equivalent optimization problem of (2.1) as follows

$$\begin{aligned} &\min\ t + \frac{1}{2}\|d\|^2 \\ &\text{s.t.}\ \ \langle \nabla f_i(x), d\rangle \le t, \quad i = 1, ..., m. \end{aligned} \tag{2.4}$$

Problem (2.4) is indeed a convex quadratic programming with linear constraints in $(n+1)-$ dimensional space $\mathbb{R}^{n+1}$. Therefore, applying KKT theorem we obtain Lagrange function corresponding to Problem (2.4)

$$L((t,d),\lambda) = t + \frac{1}{2}\|d\|^2 + \sum_{i=1}^{m} \lambda_i(\langle \nabla f_i(x), d\rangle - t)$$

and the existence of $\lambda_i(x) \geq 0, i = 1, ..., m$ such that

$$\sum_{i=1}^{m} \lambda_i(x) = 1, \tag{2.5}$$

$$d(x) + \sum_{i=1}^{m} \lambda_i(x)\nabla f_i(x) = 0, \tag{2.6}$$

$$\langle \nabla f_i(x), d(x)\rangle \leq t(x), \quad i = 1, ..., m, \tag{2.7}$$

$$\lambda_i(x)(\langle \nabla f_i(x), d(x) - t(x)\rangle) = 0, \quad i = 1, ..., m. \tag{2.8}$$

On the other hand, the strong duality property is ensured for Problem (2.4) then the optimal value of (2.4) equals to the optimal value of its dual problem, i.e.,

$$\theta(x) = \sup_{\lambda\in\mathbb{R}^m_+} \inf\{L((t,d),\lambda) \mid (t,d) \in \mathbb{R}\times\mathbb{R}^n\} = \max_{\lambda\in\Delta_m} \; -\frac{1}{2}\left\|\sum_{i=1}^{m} \lambda_i \nabla f_i(x)\right\|^2. \tag{2.9}$$

And, moreover $\lambda_i(x), i = 1, ..., m$ obtained from KKT equations are actually taken from an optimal solution of subproblem (2.9). Hence, we have

$$d(x) = -\sum_{i=1}^{m} \lambda_i(x)\nabla f_i(x) \quad \text{(from (2.6))} \tag{2.10}$$

and

$$\theta(x) = -\frac{1}{2}\left\|\sum_{i=1}^{m} \lambda_i(x)\nabla f_i(x)\right\|^2 = -\frac{1}{2}\|d(x)\|^2. \tag{2.11}$$

Similarly to Chen et al. (2023), by (2.7) and (2.8), we have the following lemma.

Lemma 2.3 (Chen et al., 2023)**.** *Suppose that f_i is differentiable for all $i = 1, ..., m$. Let $\theta(x)$ be the optimal value of Problem (2.4) (also, Problems (2.1) and (2.9)); $d(x)$ be an optimal solution of Problem (2.4) (Problem (2.1) as well); $\lambda(x) = (\lambda_1(x), ..., \lambda_m(x))$ be an optimal solution of Problem (2.9). Then the following assertions hold.*

(i)

$$\langle \nabla f_i(x), d(x)\rangle \leq t(x) = \theta(x) - \frac{1}{2}\|d(x)\|^2 = -\|d(x)\|^2, \quad i = 1, ..., m. \tag{2.12}$$

(ii) If $\lambda_i(x) \neq 0$,

$$\langle \nabla f_i(x), d(x)\rangle = t(x) = -\|d(x)\|^2. \tag{2.13}$$

(iii)

$$\left\|\sum_{i=1}^{m}\lambda_i(x)\nabla f_i(x)\right\|^2 = t(x) = -\|d(x)\|^2. \tag{2.14}$$

We also need the concept of quasi-Fejér sequence which will be useful for deriving the convergence results of our new algorithms in the upcoming sections.

Definition 2.4 (Burachik et al., 2010). *A sequence $\{y^k\}$ is quasi-Fejér convergent to a set $U \subseteq \mathbb{R}^n$ if for every $u \in U$ there exists a sequence $\{\epsilon_k\} \subseteq \mathbb{R}_+$ such that $\sum\limits_{k=0}^{+\infty}\epsilon_k < +\infty$ and*

$$\|y^{k+1} - u\|^2 \leq \|y^k - u\|^2 + \epsilon_k \quad \textit{for all } k \geq 0.$$

Lemma 2.5 (Burachik et al., 2010). *If $\{y^k\}$ is quasi-Fejér convergent to a nonempty set $U \subseteq \mathbb{R}^n$, then $\{y^k\}$ is bounded. If furthermore, a cluster point y of $\{y^k\}$ belongs to U, then*

$$\lim_{k\to+\infty} y^k = y.$$

3. Our new proposed algorithms based on steepest descent method for solving a class of nonconvex multiobjective optimization problems

In this section, we propose new steepest descent algorithms with novel adaptive stepsize selections for solving a class of nonconvex problems (1.1) satisfying Assumption 1 and Assumption 2 below.

Assumption 1. *For all $i = 1, ..., m$, f_i is differentiable and ∇f_i is L_i- Lipschitz continuous. Setting $L_{\max} = \max\{L_1, ..., L_m\}$.*

Assumption 2. *For $u, v \in \mathbb{R}^n$ and $\lambda \in \triangle_m$, the function $g_{uv\lambda} : \mathbb{R} \to \mathbb{R}$ defined by*

$$g_{uv\lambda}(t) = \sum_{i=1}^{m}\lambda_i(f_i)'_t(u + t(v-u)) = \sum_{i=1}^{m}\lambda_i\langle\nabla f_i(u + t(v-u)), v-u\rangle$$

is quasiconvex on $[0, 1]$.

We recall from Hoai et al. (2024) that the class of functions $f_i, i = 1, ..., m$ matching Assumption 2 includes convex and indefinite quadratic functions. Interestingly, in the following example, one can see that this class includes even concave functions. Based on these typical types of functions, from now on, we call functions that satisfy Assumption 2, are **quacavex** where, "qua" is taken from **qua**dratic, "cave" is taken from con**cave** and "vex" is taken from con**vex**.

Example 3.1. (i) Firstly, if each objective is an indefinite quadratic function, i.e., for any $i = 1, ..., m$; $f_i(x) = \frac{1}{2}x^TA^ix + (b^i)^Tx$ with A^i is a symmetric matrix in $\mathbb{R}^{n\times n}$ and $b^i \in \mathbb{R}^n$, then we have

$$g_{uv\lambda}(t) = \left\langle\left(\sum_{i=1}^{m}\lambda_iA^i\right)(u + t(v-u)) + \sum_{i=1}^{m}\lambda_ib^i, v-u\right\rangle$$

which is obvious linear with respect to t and therefore quasiconvex on $[0,1]$ for any $u, v \in \mathbb{R}^n$, $\lambda \in \triangle_m$. Hence every quadratic function f_i is quacavex, i.e., satisfying Assumption 2.

(ii) Secondly, if the differentiable function f_i is either convex or concave then the convexity (concavity, resp.) of f_i follows the convexity (concavity, resp.) of $f_i(u + t(v-u))$ on the set $\{t \in \mathbb{R} \mid u + t(v-u) \in \mathbb{R}^n\} \supset [0,1]$. Thus, for $\lambda \in \triangle_m$, $\sum_{i=1}^{m} \lambda_i (f_i)'_t(u + t(v-u))$ is increasing (decreasing, resp.) monotone over $[0,1]$ and therefore quasiconvex on $[0,1]$.

We continue to illustrate the practical applicability of this assumption via an example in machine learning.

Example 3.2. One of the basic models in machine learning is *linear regression* that we want to estimate the parameters $w \in \mathbb{R}^p$ to minimize the loss function

$$L(w) = \frac{1}{N}\|Y - Xw\|^2, \tag{3.1}$$

where N is the number of data used for training, $X \in \mathbb{R}^{N\times p}$ is a matrix including input data and $Y \in \mathbb{R}^N$ is an output data. To avoid overfitting phenomenon, one can use regularization technique like Ridge regularization by changing the loss function to be

$$L_{Ridge}(w) = \frac{1}{N}\|Y - Xw\|^2 + \lambda\|w\|^2,\ \lambda > 0. \tag{3.2}$$

However, it is hard to choose a suitable parameter λ. Another approach to avoid the process of finding a suitable λ is based on multiobjective optimization. In particular, we solve the following bi-objective optimization problem

$$\min_{w\in\mathbb{R}^p} \left(\frac{1}{N}\|Y - Xw\|^2, \|w\|^2\right). \tag{3.3}$$

The trade-off between two objectives will be described via the Pareto front of Problem (3.3). The owner of the model will get the desired solution according to their target. Problem (3.3) obviously satisfies Assumptions 1 and 2.

Definition 3.3. For any $x, y \in \mathbb{R}^n$ and $\lambda \in \triangle_m$, we construct a function

$$A_\ell(x, y, \lambda) : \mathbb{R}^n \times \mathbb{R}^n \times \triangle_m \to \mathbb{R},\ \ell = 1, ..., 4, \tag{3.4}$$

which is defined by one of the following options

(i) Option 1:

$$A_1(x, y, \lambda) = \sum_{i=1}^{m} \lambda_i \langle \nabla f_i(x) - \nabla f_i(y), x - y\rangle. \tag{Opt1}$$

(ii) Option 2:

$$A_2(x, y, \lambda) = \max_{i=1,...,m} |\langle \nabla f_i(x) - \nabla f_i(y), x - y\rangle|. \tag{Opt2}$$

(iii) Option 3:

$$A_3(x,y,\lambda) = \left\| \sum_{i=1}^{m} \lambda_i \left(\nabla f_i(x) - \nabla f_i(y) \right) \right\| \|x-y\|. \tag{Opt3}$$

(iii) Option 4:

$$A_4(x,y,\lambda) = \left(\max_{i=1,\ldots,m} \|\nabla f_i(x) - \nabla f_i(y)\| \right) \|x-y\|. \tag{Opt4}$$

We then easily obtain the following result related to $A_\ell(x,y,\lambda)$.

Lemma 3.4. *Suppose that Problem* (1.1) *satisfies Assumption 1 then for every* $\ell \in \{1,2,3,4\}$, *the following inequalities hold.*

$$|A_1(x,y,\lambda)| \leq |A_\ell(x,y,\lambda)| \leq L_{\max}\|x-y\|^2. \tag{3.5}$$

Proof. For the left-hand side inequality, one can use some simple evaluations as follows.

$$\begin{aligned}
|A_1(x,y,\lambda)| &= \left| \sum_{i=1}^{m} \lambda_i \langle \nabla f_i(x) - \nabla f_i(y), x-y \rangle \right| \leq \sum_{i=1}^{m} \lambda_i \left| \langle \nabla f_i(x) - \nabla f_i(y), x-y \rangle \right| \\
&\leq \sum_{i=1}^{m} \lambda_i \max_{i=1,\ldots,m} |\langle \nabla f_i(x) - \nabla f_i(y), x-y \rangle| \\
&= \max_{i=1,\ldots,m} |\langle \nabla f_i(x) - \nabla f_i(y), x-y \rangle| \ (\text{ since } \lambda \in \triangle_m) \\
&= A_2(x,y,\lambda).
\end{aligned} \tag{3.6}$$

In addition,

$$\begin{aligned}
|A_1(x,y,\lambda)| &= \left| \sum_{i=1}^{m} \lambda_i \langle \nabla f_i(x) - \nabla f_i(y), x-y \rangle \right| = \left| \left\langle \sum_{i=1}^{m} \lambda_i \left(\nabla f_i(x) - \nabla f_i(y) \right), x-y \right\rangle \right| \\
&\leq \left\| \sum_{i=1}^{m} \lambda_i \left(\nabla f_i(x) - \nabla f_i(y) \right) \right\| \|x-y\| = A_3(x,y,\lambda) \\
&\leq \sum_{i=1}^{m} \lambda_i \|(\nabla f_i(x) - \nabla f_i(y))\| \|x-y\| \\
&\leq \sum_{i=1}^{m} \lambda_i \left(\max_{i=1,\ldots,m} \|\nabla f_i(x) - \nabla f_i(y)\| \right) \|x-y\| \\
&= \left(\max_{i=1,\ldots,m} \|\nabla f_i(x) - \nabla f_i(y)\| \right) \|x-y\| \ (\text{ since } \lambda \in \triangle_m) \\
&= A_4(x,y,\lambda).
\end{aligned} \tag{3.7}$$

(3.8)

For the right-hand side inequality, the proof is based on the globally Lipschitz continuity of ∇f_i with constant L_i from Assumption 1 and $L_{\max} = \max\{L_1, ..., L_m\}$. □

In the following, we propose novel adaptive stepsizes used for steepest descent method (1.2) to solve Problem (1.1). In which, the function $A_\ell(x,y,\lambda)$ plays an important role to determine our

stepsizes.

Algorithm 3.1 NSDMOℓ, $\ell \in \{1,2,3,4\}$

Preparation: From Definition 3.3, picking an option for the function $A_\ell(x,y,\lambda),\ \ell \in \{1,2,3,4\}$. We use the name **NSDMO**ℓ for the corresponding steepest descent algorithm with this option.

Step 0. Choose $x^0 \in \mathbb{R}^n$, set $k=0$, select $t_0 > 0$, $0 < \eta_1 < \eta_0 < 1$, $\varepsilon > 0$, and a convergent positive series $\sum\limits_{k=0}^{+\infty} \varepsilon_k$.

Step 1. Compute

$$\lambda^k = \underset{\lambda \in \Delta_m}{\arg\min} \frac{1}{2} \left\| \sum_{i=1}^m \lambda_i \nabla f_i(x^k) \right\|^2 \tag{3.9}$$

$$d^k = -\sum_{i=1}^m \lambda_i^k \nabla f_i(x^k) \tag{3.10}$$

Step 2.

If $\|d^k\| < \varepsilon$ **then** STOP

else $x^{k+1} = x^k + t_k d^k$

if $A_\ell(x^{k+1}, x^k, \lambda^k) > \dfrac{\eta_0}{t_k} \|x^{k+1} - x^k\|^2$ (3.11)

then $t_{k+1} = \eta_1 \dfrac{\|x^{k+1} - x^k\|^2}{A_\ell(x^{k+1}, x^k, \lambda^k)}$ (3.12)

else $t_{k+1} = (1+\varepsilon_k) t_k$ (3.13)

$k := k+1$ and return to **Step1**.

Remark 3.5. (i) When $m = 1$ we see that: Algorithms NSDMO3 and NSDMO4 are the same and quite similar to NGD in Hoai et al. (2024); Algorithms NSDMO1 and NSDMO2 are the same but different from NGD.

(ii) It is worth noting that from Lemma 3.4 we have $|A_1(x,y,\lambda)| \le |A_\ell(x,y,\lambda)|$. Hence in the cases where the stepsize t_{k+1} is computed by formula (3.12) it feels intuitively like NSDMO1 might provide longer stepsizes than the remaining ones.

(iii) It is observed that for single objective case, the related algorithm NGD in Hoai et al. (2024) only requires locally Lipschitz continuity of the gradient of the objective, which is weaker than the global Lipschitz condition in Assumption 1. However, in order to establish the convergence results, NGD **needs the convexity of the objective** while **our new proposed algorithms can be applied for a broad class of nonconvex functions** satisfying Assumption 2.

To investigate the convergence of our new methods, some main properties of the sequence of our new stepsizes are necessary to analyze.

Lemma 3.6. *For Algorithm 3.1-NSDMOℓ, $\ell = 1, ..., 4$, we have $\inf_{k\geq 0} t_k > 0$ and $\{t_k\}$ converges to a positive limit t^*.*

Proof. We divide the proof into two parts:

(i) **Part 1:** For $k = 0$, following Algorithm 3.1, if (3.11) is true, i.e.,

$$A_\ell(x^1, x^0, \lambda^0) > \frac{\eta_0}{t_0}\|x^1 - x^0\|^2$$

then

$$t_1 = \eta_1 \frac{\|x^1 - x^0\|^2}{A_\ell(x^1, x^0, \lambda^0)} \underset{\text{by (3.5)}}{\geq} \eta_1 \frac{\|x^1 - x^0\|^2}{L_{\max}\|x^1 - x^0\|^2} = \frac{\eta_1}{L_{\max}}.$$

Conversely, we have $t_1 = (1 + \varepsilon_0)t_0 > t_0$. Therefore $t_1 \geq \min\left\{\frac{\eta_1}{L_{\max}}, t_0\right\} > 0$. By using analogous argument we derive that $t_{k+1} \geq \min\left\{\frac{\eta_1}{L_{\max}}, t_k\right\} \geq \min\left\{\frac{\eta_1}{L_{\max}}, t_0\right\}$ for all $k \geq 0$. This follows $\inf_{k\geq 0} t_k \geq \min\left\{\frac{\eta_1}{L_{\max}}, t_0\right\} = t_{\min} > 0$.

(ii) **Part 2:** Let $u_k = \ln t_{k+1} - \ln t_k$ for $k \geq 0$, we have $u_k = u_k^+ - u_k^-$, where

$$u_k^+ = \max\{0, u_k\}, u_k^- = -\min\{0, u_k\}.$$

Then $u_k^+ \geq 0$ and $u_k^- \geq 0$ for all $k \geq 0$. From the definition of t_k in Algorithm 3.1, we derive that

$$u_k = \ln\frac{t_{k+1}}{t_k} \leq \ln(1 + \varepsilon_k) \leq \varepsilon_k \quad \forall k \geq 0,$$

which implies $u_k^+ \leq \varepsilon_k$. Since $\sum_{k=0}^{+\infty} \varepsilon_k$ is convergent, we obtain $\sum_{k=0}^{+\infty} u_k^+ < +\infty$. Observing that $\sum_{k=0}^{+\infty} u_k^-$ is a nonnegative series and using the following relation

$$\ln t_{k+1} - \ln t_0 = \sum_{i=0}^{k} u_i = \sum_{i=0}^{k}(u_i^+ - u_i^-) = \sum_{i=0}^{k} u_i^+ - \sum_{i=0}^{k} u_i^-, \tag{3.14}$$

we assert that if $\lim_{k\to+\infty} \sum_{i=0}^{k} u_i^- = +\infty$ then

$$\lim_{k\to+\infty}(\ln t_{k+1}) = -\infty \iff \lim_{k\to+\infty} t_k = 0.$$

From **Part 1**, we have $\inf_{k\geq 0} t_k > 0$. This contradiction proves the convergence of $\sum_{k=0}^{+\infty} u_k^-$. Finally, from (3.14) we get the desired conclusion that $\lim_{k\to+\infty} t_k = t^* < +\infty$.

□

Lemma 3.7. *For each $\ell \in \{1,2,3,4\}$, let $\{x^k\}$ and $\{\lambda^k\}$ be sequences generated by Algorithm 3.1 - NSDMOℓ. Then there exists $\hat{k}_\ell$ such that*

$$A_\ell(x^{k+1}, x^k, \lambda^k) \leq \frac{\eta_0}{t_k}\|x^{k+1} - x^k\|^2 \ \ \forall k \geq \hat{k}_\ell. \tag{3.15}$$

Proof. Suppose by contradiction that there exists $\{k_j\}, k_j \to +\infty$ such that

$$A_\ell(x^{k_j+1}, x^{k_j}, \lambda^{k_j}) > \frac{\eta_0}{t_{k_j}}\|x^{k_j+1} - x^{k_j}\|^2.$$

For this case,

$$t_{k_j+1} = \eta_1 \frac{\|x^{k_j+1} - x^{k_j}\|^2}{A_\ell(x^{k_j+1}, x^{k_j}, \lambda^{k_j})}.$$

Consequently,

$$\frac{\eta_1\|x^{k_j+1} - x^{k_j}\|^2}{t_{k_j+1}} = A_\ell(x^{k_j+1}, x^{k_j}, \lambda^{k_j}) > \frac{\eta_0}{t_{k_j}}\|x^{k_j+1} - x^{k_j}\|^2,$$

i.e., $\dfrac{t_{k_j+1}}{t_{k_j}} < \dfrac{\eta_1}{\eta_0}$ for all k_j. On the other hand, from Lemma 3.6 we have $\lim\limits_{k_j\to+\infty} t_{k_j+1} = \lim\limits_{k_j\to+\infty} t_{k_j} = \lim\limits_{k\to+\infty} t_k = t^*$, hence we deduce that $\dfrac{t^*}{t^*} \leq \dfrac{\eta_1}{\eta_0} < 1$. It is a contradiction and we finish the proof. □

Remark 3.8. From Lemmas 3.6 and 3.7 we obtain the following relation

(i) $t_k \geq t_{\min}$ for all $k \geq 0$;

(ii) $t_{\min} \leq t_{\hat{k}_\ell} \leq t_k \leq t_{k+1} \leq t^*$ for all $k \geq \hat{k}_\ell$.

Lemma 3.9. *If Problem (1.1) satisfies Assumption 1 and Assumption 2 then the sequence $\{x^k\}$ generated by Algorithm 3.1 - NSDMOℓ, ($\ell \in \{1,2,3,4\}$) has the following property:*

$$\sum_{i=1}^m \lambda_i^k \left(f_i(x^k) - f_i(x^{k+1})\right) \geq \frac{1-\eta_0}{t_k}\|x^{k+1} - x^k\|^2, \ \forall k \geq \hat{k}_\ell. \tag{3.16}$$

Especially, if $\ell \in \{2,4\}$ then we have a stronger assertion that

$$f_i(x^k) - f_i(x^{k+1}) \geq \frac{1-\eta_0}{t_k}\|x^{k+1} - x^k\|^2, \ \forall k \geq \hat{k}_\ell \tag{3.17}$$

for any $i \in \{1, ..., m\}$.

Proof. We divide the proof into two parts:

Part 1: To prove (3.16).

We easily obtain the following relation from the Fundamental Theorem of Calculus

$$\begin{aligned}\sum_{i=1}^{m}\lambda_i^k\left(f_i(x^{k+1})-f_i(x^k)\right) &= \sum_{i=1}^{m}\lambda_i^k\int_0^1\langle\nabla f_i(x^k+t(x^{k+1}-x^k)),x^{k+1}-x^k\rangle dt\\ &= \sum_{i=1}^{m}\lambda_i^k\langle\nabla f_i(x^k),x^{k+1}-x^k\rangle+\int_0^1 h_k(t)dt,\end{aligned}\tag{3.18}$$

where

$$\begin{aligned}h_k(t) &= \sum_{i=1}^{m}\lambda_i^k\left\langle\nabla f_i(x^k+t(x^{k+1}-x^k))-\nabla f_i(x^k),x^{k+1}-x^k\right\rangle\\ &= g_{x^kx^{k+1}\lambda^k}(t)-\sum_{i=1}^{m}\lambda_i^k\left\langle\nabla f_i(x^k),x^{k+1}-x^k\right\rangle.\end{aligned}\tag{3.19}$$

By Assumption 2, $g_{x^kx^{k+1}\lambda^k}(t)$ is quasiconvex on $[0,1]$ and hence so $h_k(t)$ is. Therefore, for all $t\in[0,1]$,

$$\begin{aligned}h_k(t) &\le \max\{h_k(0),h_k(1)\}=\max\{0,h_k(1)\}\\ &\le |h_k(1)|=\left|\sum_{i=1}^{m}\lambda_i^k\left\langle\nabla f_i(x^{k+1})-\nabla f_i(x^k),x^{k+1}-x^k\right\rangle\right|\\ &= |A_1(x^{k+1},x^k,\lambda^k)|\le|A_\ell(x^{k+1},x^k,\lambda^k)|\ \text{(by Lemma 3.4)}.\end{aligned}\tag{3.20}$$

For Algorithm 3.1 - NSDMOℓ, with $\ell\in\{1,2,3,4\}$, utilizing the result of Lemma 3.7 we take $k\ge\hat{k}_\ell$ and then using (3.20) to get

$$\begin{aligned}\int_0^1 h_k(t)dt &\le \left|A_\ell(x^{k+1},x^k,\lambda^k)\right|\\ &\le \frac{\eta_0}{t_k}\left\|x^{k+1}-x^k\right\|^2,\ \ \forall k\ge\hat{k}_\ell.\end{aligned}\tag{3.21}$$

Moreover, from Lemma 2.3 (i), we have

$$\langle\nabla f_i(x^k),x^{k+1}-x^k\rangle=\langle\nabla f_i(x^k),t_kd^k\rangle\le-t_k\|d^k\|^2=-\frac{1}{t_k}\|x^{k+1}-x^k\|^2\tag{3.22}$$

which follows that

$$\sum_{i=1}^{m}\lambda_i^k\langle\nabla f_i(x^k),x^{k+1}-x^k\rangle\le-\frac{\sum_{i=1}^{m}\lambda_i^k}{t_k}\left\|x^{k+1}-x^k\right\|^2=-\frac{1}{t_k}\|x^{k+1}-x^k\|^2.\tag{3.23}$$

Now from (3.18), (3.21) and (3.23) we obtain that

$$\sum_{i=1}^{m} \lambda_i^k \left(f_i(x^k) - f_i(x^{k+1}) \right) \geq \frac{1-\eta_0}{t_k} \|x^{k+1} - x^k\|^2, \ \forall k \geq \hat{k}_\ell.$$

Hence, the proof of (3.16) is finished.

Part 2: In order to prove (3.17), we use analogous arguments of Part 1 for each function $f_i, i = 1, ..., m$ as follows

$$\begin{aligned} f_i(x^{k+1}) - f_i(x^k) &= \int_0^1 \langle \nabla f_i(x^k + t(x^{k+1} - x^k)), x^{k+1} - x^k \rangle dt \\ &= \langle \nabla f_i(x^k), x^{k+1} - x^k \rangle + \int_0^1 h_k^i(t) dt, \end{aligned} \tag{3.24}$$

with

$$\begin{aligned} h_k^i(t) &= \left\langle \nabla f_i(x^k + t(x^{k+1} - x^k)) - \nabla f_i(x^k), x^{k+1} - x^k \right\rangle \\ &= g_{x^k x^{k+1} e^i}(t) - \left\langle \nabla f_i(x^k), x^{k+1} - x^k \right\rangle, \end{aligned}$$

where e^i is the $i-$th unit vector of $\mathbb{R}^m$. By Assumption 2, $g_{x^k x^{k+1} e^i}(t)$ is quasiconvex on $[0,1]$ and hence so $h_k^i(t)$ is. Therefore, for all $t \in [0,1]$,

$$\begin{aligned} h_k^i(t) &\leq \max\{h_k^i(0), h_k^i(1)\} = \max\{0, h_k^i(1)\} \\ &\leq |h_k^i(1)| = \left| \left\langle \nabla f_i(x^{k+1}) - \nabla f_i(x^k), x^{k+1} - x^k \right\rangle \right| \\ &\leq \max_{i=1,...,m} \left| \left\langle \nabla f_i(x^{k+1}) - \nabla f_i(x^k), x^{k+1} - x^k \right\rangle \right| = |A_2(x^{k+1}, x^k, \lambda^k)| \qquad (3.25) \\ &\leq \max_{i=1,...,m} \left\| \nabla f_i(x^{k+1}) - \nabla f_i(x^k) \right\| \left\| x^{k+1} - x^k \right\| = |A_4(x^{k+1}, x^k, \lambda^k)|. \qquad (3.26) \end{aligned}$$

In short, we can use Lemma 3.7 for Algorithm 3.1 - NSDMOℓ with $\ell \in \{2, 4\}$ and plugging (3.25) and (3.26) into $\displaystyle\int_0^1 h_k^i(t) dt$ as below

$$\int_0^1 h_k^i(t) dt \leq \left| A_\ell(x^{k+1}, x^k, \lambda^k) \right| \leq \frac{\eta_0}{t_k} \left\| x^{k+1} - x^k \right\|^2, \ \ \forall k \geq \hat{k}_\ell. \tag{3.27}$$

Finally, combining (3.22), (3.24) with (3.27) we obtain that

$$f_i(x^k) - f_i(x^{k+1}) \geq \frac{1-\eta_0}{t_k} \|x^{k+1} - x^k\|^2, \ \forall k \geq \hat{k}_\ell.$$

The desired inequality (3.17) is then obtained. □

From Lemma 3.9, it is easy to derive some nice properties of Algorithm 3.1 - NSDMOℓ, $\ell \in$

$\{1, 2, 3, 4\}$ in the following corollary.

Corollary 3.10. *Suppose that Problem* (1.1) *satisfies Assumption 1 and Assumption 2 then Algorithm 3.1 - NSDMOℓ (for $\ell \in \{1, 2, 3, 4\}$) generates the sequence $\{x^k\}$ satisfying that*

(i) the objective sequence $\{F(x^k)\}_{k\geq \hat{k}_\ell}$ is non-increasing, i.e. $F(x^k) \not\leq F(x^{k+1})$ for all $k \geq \hat{k}_\ell$;

(ii)

$$\max_{i\in\{1,\ldots,m\}} \left(f_i(x^k) - f_i(x^{k+1})\right) \geq \frac{1-\eta_0}{t_k}\|x^{k+1} - x^k\|^2 = (1-\eta_0)t_k\|d^k\|^2,\ \forall k \geq \hat{k}_\ell.$$

The results of Corollary 3.10 show that from iteration $\hat{k}_\ell$, Algorithm 3.1-NSDMOℓ, $\ell \in \{1, 2, 3, 4\}$ navigate the sequence $\{x^k\}_{k\geq \hat{k}_\ell}$ out of the dominated domain step by step, i.e., $F(x^{k+1}) \notin F(x^k) + \mathbb{R}^m_+$. Unless $\|d^k\| = 0$, we can always find x^{k+1} that is no worse than x^k in the sense that $F(x^{k+1})$ is not dominated by $F(x^k)$, and at least one component of $F(x^{k+1})$ is smaller than the corresponding one of $F(x^k)$ by a significant positive amount $(1-\eta_0)t_k\|d^k\|^2$. Nevertheless, these results are not enough for ensuring the convergence of Algorithm 3.1 in solving Problem (1.1), i.e., showing $\|d^k\| \to 0$ or at least $\inf_{k\geq 0} \|d^k\| = 0$. Fortunately, with $\ell = 2$ or 4, we can prove that if at least one objective is lower bounded on $\mathbb{R}^n$. This result is given in the following theorem.

Theorem 3.11. *Suppose that Problem* (1.1) *satisfies Assumption 1 and Assumption 2 and there exists $i_0 \in \{1, ..., m\}$ such that f_{i_0} is lower bounded on $\mathbb{R}^n$. Then the sequence $\{x^k\}$ generated by Algorithm 3.1 - NSDMOℓ with $\ell = 2$ or 4 satisfies*

(i) The objective sequence $\{F(x^k)\}_{k\geq \hat{k}_\ell}$ is decreasing, i.e., $F(x^k) \geq F(x^{k+1})$ for all $k \geq \hat{k}_\ell$.

(ii) $\|d^k\| \to 0$ as $k \to +\infty$ and

$$\min_{\hat{k}_\ell \leq k \leq K-1} \|d^k\| \leq O\left(\frac{1}{\sqrt{K}}\right),\ \ K > \hat{k}_\ell.$$

(iii) Any accumulation point of $\{x^k\}$ is Pareto critical.

Proof. (i) Using Lemma 3.9 - inequality (3.17) for any $i \in \{1, ..., m\}$ we have that

$$\begin{aligned} f_i(x^k) - f_i(x^{k+1}) &\geq \frac{1-\eta_0}{t_k}\|x^{k+1} - x^k\|^2 = (1-\eta_0)t_k\|d^k\|^2 \\ &\geq t_{\min}(1-\eta_0)\|d^k\|^2,\ \forall k \geq \hat{k}_\ell. \end{aligned} \tag{3.28}$$

This follows that $\{f_i(x^k)\}_{k\geq \hat{k}_\ell}$ is monotone decreasing for all $i \in \{1, ..., m\}$. We obtain the first conclusion.

(ii) Now utilizing the lower bounded condition imposed on the function f_{i_0} we derive that the sequence $\{f_{i_0}(x^k)\}_{k\geq \hat{k}_\ell}$ is decreasing and bounded from below hence it converges to a finite

limit $\hat{f}_{i_0}$ as $k \to +\infty$. It means that $\lim_{k\to+\infty}(f_{i_0}(x^k) - f_{i_0}(x^{k+1})) = 0$ and from (3.28), we get that $\lim_{k\to+\infty}\|d^k\| = 0$. Next, summing up inequality (3.28) from $k = \hat{k}_\ell$ to $K-1 \geq \hat{k}_\ell$, yielding that

$$t_{\min}(1-\eta_0)\sum_{k=\hat{k}_\ell}^{K-1}\|d^k\|^2 \leq f_{i_0}(x^{\hat{k}_\ell}) - f_{i_0}(x^K) \leq f_{i_0}(x^{\hat{k}_\ell}) - \hat{f}_{i_0}.$$

Thus, we obtain

$$\min_{\hat{k}_\ell \leq k \leq K-1}\|d^k\| \leq \sqrt{\frac{f_{i_0}(x^{\hat{k}_\ell}) - \hat{f}_{i_0}}{t_{\min}(1-\eta_0)(K-\hat{k}_\ell)}} = O\left(\frac{1}{\sqrt{K}}\right). \tag{3.29}$$

(iii) Suppose that x^* is a cluster point of $\{x^k\}$ then there exists a subsequence $\{x^{k_j}\}$ such that $x^{k_j} \to x^*$. Remember that $d^k = d(x^k)$ and $x \mapsto d(x)$ is continuous (by Lemma 2.2(c)) therefore

$$d(x^*) = \lim_{x^{k_j}\to x^*} d(x^{k_j}) = \lim_{k_j\to+\infty} d(x^{k_j}) = \lim_{k\to+\infty} d(x^k) = \lim_{k\to+\infty} d^k = 0.$$

Again, applying Lemma 2.2 (a) we infer that x^* is a Pareto critical point.

□

4. The convergence rate results of Algorithm 3.1-NSDMOℓ ($\ell = 2$ or 4) for solving the convex multiobjective optimization problems

As seen in the previous section, Algorithm 3.1 - NSDMOℓ with $\ell = 2$ or 4 ensures the decreasing of all objectives simultaneously from a finite iteration $\hat{k}_\ell$. This property is essential in establishing the convergence rate of these algorithms. In this section, we will analyze the convergence properties of Algorithm 3.1 - NSDMOℓ with $\ell = 2$ or 4 more thoroughly in the case when all objective functions matching the Assumption 3 below.

Assumption 3. *All the objectives f_i, $i = 1, ..., m$ are convex.*

Obviously, Assumption 3 is stronger than Assumption 2. Therefore, under Assumption 1 and Assumption 3, Algorithm 3.1 - NSDMOℓ with $\ell = 2$ or 4 have all properties in Theorem 3.11. Besides, it admits an additional result below.

Lemma 4.1. *Under Assumption 1 and Assumption 3, and for any $x \in \mathbb{R}^n$, Algorithm 3.1 - NSDMOℓ with $\ell = 2$ or 4 generates a sequence $\{x^k\}$ satisfying that*

$$\sum_{i=1}^m \lambda_i^k\left(f_i(x) - f_i(x^{k+1})\right) \geq \frac{1-\eta_0}{t_k}\|x^{k+1} - x^k\|^2 + \frac{1}{t_k}\langle x^k - x^{k+1}, x - x^k\rangle,\ \forall k \geq \hat{k}_\ell. \tag{4.1}$$

Proof. Firstly, we use the convexity of f_i, for all $i = 1, ..., m$ to derive that

$$\begin{aligned}\sum_{i=1}^{m} \lambda_i^k \left(f_i(x) - f_i(x^{k+1})\right) &= \sum_{i=1}^{m} \lambda_i^k \left(f_i(x) - f_i(x^k) + f_i(x^k) - f(x^{k+1})\right) \\ &\geq \sum_{i=1}^{m} \lambda_i^k \left(\langle \nabla f_i(x^k), x - x^k\rangle + \langle \nabla f_i(x^{k+1}), x^k - x^{k+1}\rangle\right) \\ &= \left\langle \sum_{i=1}^{m} \lambda_i^k \nabla f_i(x^k), x - x^k \right\rangle + \sum_{i=1}^{m} \lambda_i^k \left\langle \nabla f_i(x^{k+1}), x^k - x^{k+1}\right\rangle. \qquad (4.2)\end{aligned}$$

Because of the definition of $\{x^k\}$ it is easy to see that

$$\left\langle \sum_{i=1}^{m} \lambda_i^k \nabla f_i(x^k), x - x^k \right\rangle = \frac{1}{t_k}\langle x^k - x^{k+1}, x - x^k\rangle. \qquad (4.3)$$

From the definition of $A_\ell(x, y, \lambda)$ and Lemma 3.4,

$$\begin{aligned}\sum_{i=1}^{m} \lambda_i^k \left\langle \nabla f_i(x^{k+1}) - \nabla f_i(x^k), x^k - x^{k+1}\right\rangle &= -\sum_{i=1}^{m} \lambda_i^k \left\langle \nabla f_i(x^k) - \nabla f_i(x^{k+1}), x^k - x^{k+1}\right\rangle \\ &= -A_1(x^{k+1}, x^k, \lambda^k) \\ &\geq -A_\ell(x^{k+1}, x^k, \lambda^k) \text{ (for } \ell = 2; 4 - \text{ from (3.6); (3.8))} \\ &\underset{\text{by(3.15)}}{\geq} -\frac{\eta_0}{t_k}\|x^{k+1} - x^k\|^2 \quad \forall k \geq \hat{k}_\ell. \qquad (4.4)\end{aligned}$$

Therefore

$$\begin{aligned}\sum_{i=1}^{m} \lambda_i^k \left\langle \nabla f_i(x^{k+1}), x^k - x^{k+1}\right\rangle &\geq -\frac{\eta_0}{t_k}\|x^{k+1} - x^k\|^2 + \sum_{i=1}^{m} \lambda_i^k \left\langle \nabla f_i(x^k), x^k - x^{k+1}\right\rangle \quad \forall k \geq \hat{k}_\ell \\ &= \frac{1 - \eta_0}{t_k}\|x^{k+1} - x^k\|^2 \quad \forall k \geq \hat{k}_\ell. \qquad (4.5)\end{aligned}$$

Lastly, equation (4.3) and inequalities (4.2), (4.5) indicate the desired conclusion (4.1). □

We continue to analyze the convergence results of Algorithm 3.1-NSDMOℓ with $\ell = 2$ or 4 with the following assumption.

Assumption 4. *The set $T = \{x \in \mathbb{R}^n \mid f_i(x) \leq f_i(x^k),\ \forall k \geq \hat{k}_\ell\}$ is nonempty.*

It is worth noting that in the literature, Assumption 4 was used in some papers such as Drummond & Iusem (2004) and Cruz et al. (2012) for projected gradient method in solving constrained multiobjective optimization problems. Basically, it is related to the completeness of $\mathrm{Im}(f)$ that is equivalent to the existence of efficient set of Problem (1.1) (see e.g., Luc, 1989).

Theorem 4.2. *If Problem* (1.1) *satisfies Assumption 1, Assumption 3 and Assumption 4. Then for Algorithm 3.1 - NSDMOℓ with $\ell = 2$ or 4, we have that*

(i) The sequence $\{x^k\}$ converges to a weakly efficient point x^* of Problem (1.1) and $x^* \in T$.

(ii) For $K > \hat{k}_\ell$,

$$\sum_{i=1}^{m} \bar{\lambda}_i^{K-1} f_i(x^K) - \sum_{i=1}^{m} \bar{\lambda}_i^{K-1} f_i(x^*) \leq \frac{D}{2t_{\min}(K - \hat{k}_\ell)} = O\left(\frac{1}{K}\right),$$

where $\bar{\lambda}_i^{K-1} = \dfrac{1}{K - \hat{k}_\ell} \sum\limits_{k=\hat{k}_\ell}^{K-1} \lambda_i^k$, $\bar{\lambda}^{K-1} = (\bar{\lambda}_1^{K-1}, ..., \bar{\lambda}_m^{K-1}) \in \triangle_m$, and

$$D = \max\left\{\|x^{\hat{k}_\ell} - x^*\|^2, \|x^{\hat{k}_\ell} - x^*\|^2 + \frac{t^*(2\eta_0 - 1)}{1 - \eta_0}(f_{i_0}(x^{\hat{k}_\ell}) - f_{i_0}(x^*)\right\}$$

with i_0 is an arbitrary member of $\{1, ..., m\}$.

(iii) If $0 < \eta_0 \leq \frac{1}{2}$ and f_i is strongly convex with modulus $\mu_i > 0$, $i = 1, ..., m$ then $\{x^k\}_{k \geq \hat{k}_\ell}$ is linearly convergent.

Proof. (i) For any $x^* \in T$ we have

$$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 + \|x^{k+1} - x^k\|^2 + 2\langle x^{k+1} - x^k, x^k - x^*\rangle. \tag{4.6}$$

Next, we rewrite

$$\begin{aligned} 2\langle x^{k+1} - x^k, x^k - x^*\rangle &= 2t_k \sum_{i=1}^{m} \lambda_i^k \langle \nabla f_i(x^k), x^* - x^k\rangle \\ &\leq 2t_k \sum_{i=1}^{m} \lambda_i^k (f_i(x^*) - f_i(x^k)) \leq 0, \ \forall k \geq \hat{k}_\ell, \end{aligned} \tag{4.7}$$

where the two last inequalities are followed from the convexity of $f_i, i = 1, ..., m$ and $x^* \in T$. Moreover, from (3.17), taking some $i_0 \in \{1, ..., m\}$ we remember that

$$\|x^{k+1} - x^k\|^2 \leq \frac{t_k}{1 - \eta_0}(f_{i_0}(x^k) - f_{i_0}(x^{k+1})) \underset{\text{Remark 3.8}}{\leq} \frac{t^*}{1 - \eta_0}(f_{i_0}(x^k) - f_{i_0}(x^{k+1})), \ \forall k \geq \hat{k}_\ell. \tag{4.8}$$

Now, plugging (4.8) and (4.7) into (4.6) to get that

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 + \frac{t^*}{1 - \eta_0}\left(f_{i_0}(x^k) - f_{i_0}(x^{k+1})\right). \tag{4.9}$$

Observing that for all $K > \hat{k}_\ell$ we have

$$\sum_{k=\hat{k}_\ell}^{K-1} \frac{t^*}{1 - \eta_0}\left(f_{i_0}(x^k) - f_{i_0}(x^{k+1})\right) = \frac{t^*}{1 - \eta_0}\left(f_{i_0}(x^{\hat{k}_\ell}) - f_{i_0}(x^K)\right) \leq \frac{t^*}{1 - \eta_0}\left(f_{i_0}(x^{\hat{k}_\ell}) - f_{i_0}(x^*)\right).$$

Therefore, from (4.9) and according to Definition 2.4 we obtain that $\{x^k\}_{k\geq\hat{k}_\ell}$ is quasi-Fejér convergent to T. Hence, by Lemma 2.5 we derive the boundedness of $\{x^k\}$. Suppose that $\bar{x}$ is a cluster point of $\{x^k\}$. We will prove that $\bar{x}\in T$. Indeed, taking a subsequence $\{x^{k_j}\}$ that converges to $\bar{x}$. Note that, for each $i\in\{1,...,m\}$, $\{f_i(x^k)\}_{k\geq\hat{k}_\ell}$ is monotone decreasing and lower bounded by $f_i(z), z\in T$ hence it converges to f_i^*. Thus $\lim\limits_{k\to+\infty} f_i(x^k)=\lim\limits_{k_j\to+\infty} f_i(x^{k_j})=f_i^*$. However, the continuity of f_i follows that $\lim\limits_{k_j\to+\infty} f_i(x^{k_j})=f_i(\bar{x})$. Consequently, $\lim\limits_{k\to+\infty} f_i(x^k)=f_i^*=f_i(\bar{x})$ which indicates $f_i(\bar{x})\leq f_i(x^k)$ for all $k\geq\hat{k}_\ell$. It means that $\bar{x}\in T$. Now, we can apply Lemma 2.5 for $\{x^k\}_{k\geq\hat{k}_\ell}$ which satisfies two conditions: first, it is quasi-Fejér convergent to T and second, all cluster points of $\{x^k\}_{k\geq\hat{k}_\ell}$ belong to T. Therefore $\{x^k\}_{k\geq\hat{k}_\ell}$ converges to some $x^*\in T$.

It remains to show that x^* is a weakly efficient point of Problem (1.1). Indeed, since $\{\lambda^k\}$ lies in a unit simplex of $\mathbb{R}^m$ then we can take a subsequence $\{\lambda^{k_\nu}\}$ such that $\lambda^{k_\nu}\to\lambda^*$ and $\sum\limits_{i=1}^m\lambda_i^*=1$. Therefore, by Theorem 3.11 (ii) we derive that

$$\left\|\sum_{i=1}^m\lambda_i^*\nabla f_i(x^*)\right\|=\lim_{k_\nu\to+\infty}\left\|\sum_{i=1}^m\lambda_i^{k_\nu}\nabla f_i(x^{k_\nu})\right\|=\lim_{k\to+\infty}\left\|\sum_{i=1}^m\lambda_i^k\nabla f_i(x^k)\right\|=0$$

which infers

$$\sum_{i=1}^m\lambda_i^*\nabla f_i(x^*)=0. \tag{4.10}$$

Now, utilizing the convexity of f_i, $i=1,..,m$, we obtain the weak efficiency of x^* for Problem (1.1). One can see e.g., Chapter 3 in Ehrgott (2005) for details.

(ii) Taking some $x^*\in T$, from Lemma 4.1, we have for all $k\geq\hat{k}_\ell$,

$$\begin{aligned}\sum_{i=1}^m\lambda_i^k\left(f_i(x^{k+1})-f_i(x^*)\right)&\leq\frac{-1+\eta_0}{t_k}\|x^{k+1}-x^k\|^2-\frac{1}{2t_k}\left(\|x^{k+1}-x^*\|^2-\|x^k-x^*\|^2-\|x^{k+1}-x^k\|^2\right),\\&\leq\frac{1}{2t_k}\left(\|x^k-x^*\|^2-\|x^{k+1}-x^*\|^2\right)+\frac{2\eta_0-1}{2t_k}\|x^{k+1}-x^k\|^2.\end{aligned} \tag{4.11}$$

Hence,

$$2t_k\sum_{i=1}^m\lambda_i^k\left(f_i(x^{k+1})-f_i(x^*)\right)\leq\left(\|x^k-x^*\|^2-\|x^{k+1}-x^*\|^2\right)+(2\eta_0-1)\|x^{k+1}-x^k\|^2. \tag{4.12}$$

The nonnegative of $\left(f_i(x^{k+1})-f_i(x^*)\right)$ and $t_k\geq t_{\min}$ for all $k\geq 0$ follow that

$$2t_{\min}\sum_{i=1}^m\lambda_i^k\left(f_i(x^{k+1})-f_i(x^*)\right)\leq\left(\|x^k-x^*\|^2-\|x^{k+1}-x^*\|^2\right)+(2\eta_0-1)\|x^{k+1}-x^k\|^2,\ \forall k\geq\hat{k}_\ell. \tag{4.13}$$

Summing up (4.13) from $k=\hat{k}_\ell$ to $K-1\geq\hat{k}_\ell$ we obtain that

$$2t_{\min}\sum_{k=\hat{k}_\ell}^{K-1}\sum_{i=1}^{m}\lambda_i^k\left(f_i(x^{k+1})-f_i(x^*)\right)\leq\left(\|x^{\hat{k}_\ell}-x^*\|^2-\|x^K-x^*\|^2\right)+(2\eta_0-1)\sum_{k=\hat{k}_\ell}^{K-1}\|x^{k+1}-x^k\|^2$$
$$\leq\|x^{\hat{k}_\ell}-x^*\|^2+(2\eta_0-1)\sum_{k=\hat{k}_\ell}^{K-1}\|x^{k+1}-x^k\|^2. \tag{4.14}$$

Nevertheless, from (4.8)

$$\sum_{k=\hat{k}_\ell}^{K-1}\|x^{k+1}-x^k\|^2\leq\sum_{k=\hat{k}_\ell}^{K-1}\frac{t_k}{1-\eta_0}(f_i(x^k)-f_i(x^{k+1}))\leq\frac{t^*}{1-\eta_0}(f_{i_0}(x^{\hat{k}_\ell})-f_{i_0}(x^K))$$
$$\leq\frac{t^*}{1-\eta_0}(f_{i_0}(x^{\hat{k}_\ell})-f_{i_0}(x^*)). \tag{4.15}$$

Remember that for all $i=1,...,m$, $0\leq f_i(x^K)-f_i(x^*)\leq f_i(x^{k+1})-f_i(x^*)$ with $\hat{k}_\ell\leq k\leq K-1$. Combining (4.14) and (4.15) we get that

$$2t_{\min}\sum_{i=1}^{m}\sum_{k=\hat{k}_\ell}^{K-1}\lambda_i^k\left(f_i(x^K)-f_i(x^*)\right)\leq 2t_{\min}\sum_{i=1}^{m}\sum_{k=\hat{k}_\ell}^{K-1}\lambda_i^k\left(f_i(x^{k+1})-f_i(x^*)\right)\leq D, \tag{4.16}$$

where $D=\max\left\{\|x^{\hat{k}_\ell}-x^*\|^2,\|x^{\hat{k}_\ell}-x^*\|^2+\dfrac{t^*(2\eta_0-1)}{1-\eta_0}(f_{i_0}(x^{\hat{k}_\ell})-f_{i_0}(x^*)\right\}$. The last inequality derives that

$$\sum_{i=1}^{m}\bar{\lambda}_i^{K-1}f_i(x^K)-\sum_{i=1}^{m}\bar{\lambda}_i^{K-1}f_i(x^*)\leq\frac{D}{2t_{\min}(K-\hat{k}_\ell)}=O\left(\frac{1}{K}\right),$$

where $\bar{\lambda}_i^{K-1}=\dfrac{1}{K-\hat{k}_\ell}\sum\limits_{k=\hat{k}_\ell}^{K-1}\lambda_i^k$. Obviously, $\bar{\lambda}^{K-1}=(\bar{\lambda}_1^{K-1},...,\bar{\lambda}_m^{K-1})\in\triangle_m$.

(iii) Let $\mu=\min\{\mu_i,i=1,...,m\}>0$ and update inequality (4.2) in Lemma 4.1 for $\mu-$ strongly convex f_i, $i=1,...,m$, we have for all $x\in\mathbb{R}^n$,

$$\sum_{i=1}^{m}\lambda_i^k\left(f_i(x)-f_i(x^{k+1})\right)\geq\frac{1-\eta_0}{t_k}\|x^{k+1}-x^k\|^2+\frac{1}{t_k}\langle x^k-x^{k+1},x-x^k\rangle+\frac{\mu}{2}\|x^k-x^*\|^2,\ \forall k\geq\hat{k}_\ell. \tag{4.17}$$

Repeating the arguments in (4.11) yields that

$$\begin{aligned} 0 \leq \sum_{i=1}^{m} \lambda_i^k \left(f_i(x^{k+1}) - f_i(x^*) \right) &\leq \frac{1}{2t_k} \left(\|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2 \right) + \\ &\quad + \frac{2\eta_0 - 1}{2t_k} \|x^{k+1} - x^k\|^2 - \frac{\mu}{2} \|x^k - x^*\|^2 \\ &\underset{\eta_0 \leq \frac{1}{2}}{\leq} \left(\frac{1}{2t_k} - \frac{\mu}{2} \right) \|x^k - x^*\|^2 - \frac{1}{2t_k} \|x^{k+1} - x^*\|^2, \ \forall k \geq \hat{k}_\ell. \end{aligned} \tag{4.18}$$

The last inequality indicates that

$$\|x^{k+1} - x^*\|^2 \leq (1 - \mu t_k) \|x^k - x^*\|^2 \leq (1 - \mu t_{\min}) \|x^k - x^*\|^2, \ \forall k \geq \hat{k}_\ell. \tag{4.19}$$

This proves the linear convergence of $\{x^k\}_{k \geq \hat{k}_\ell}$.

□

5. Numerical experiments

In this section, we compare the performance of our proposed NSDMOℓ ($\ell = 1, 2, 3, 4$) algorithms with the basic version of steepest descent method SDMO (Fliege et al., 2019; Fliege & Svaiter, 2000) and BBDMO (Chen et al., 2023), where the backtracking stepsizes of these ones are determined based on the Armijo rule as below.

Algorithm 5.1 Armijo_line_search

Input : $x^k \in \mathbb{R}^n, d^k \in \mathbb{R}^n, JF(x^k) \in \mathbb{R}^{m \times n}, \sigma, \gamma \in (0, 1), \beta^k = 1$
Output: β^k

while $F(x^k + \beta^k d^k) - F(x^k) \not\leq \sigma \beta^k JF(x^k) d^k$ **do**
  $\beta^k \leftarrow \gamma \beta^k$
end

Algorithm 5.2 SDMO (Fliege & Svaiter, 2000; Fliege et al., 2019)

Step 0. Choose $x^0 \in \mathbb{R}^n, \varepsilon > 0$ and set $k = 0$.

Step 1. Compute

$$\lambda^k = \arg\min_{\lambda \in \Delta_m} \frac{1}{2} \left\| \sum_{i=1}^{m} \lambda_i \nabla f_i(x^k) \right\|^2$$

$$d^k = -\sum_{i=1}^{m} \lambda_i^k \nabla f_i(x^k)$$

Step 2.

if $\|d^k\| < \varepsilon$ **then** STOP

else

$\beta^k = \text{Armijo\_line\_search}(x^k, d^k, JF(x^k))$

$x^{k+1} = x^k + \beta^k d^k$

$k := k + 1$ and return to **Step1**.

The detailed description of BBDMO is also provided in the following.

Algorithm 5.3 BBDMO (Chen et al., 2023)

Step 0. Choose $x^0 \in \mathbb{R}^n, \varepsilon > 0$ and set $k = 0$.

Step 1. Compute α^k

$$\textbf{if} \quad k = 0 \quad \textbf{then} \quad \alpha_i^k = 1, i = 1, \ldots, m$$

else

$$s^{k-1} = x^k - x^{k-1}, y_i^{k-1} = \nabla f_i\left(x^k\right) - \nabla f_i\left(x^{k-1}\right), \quad i = 1, \ldots, m,$$

$$\alpha_i^k = \begin{cases} \max\left\{\alpha_{\min}, \min\left\{\frac{\left\langle s^{k-1}, y_i^{k-1}\right\rangle}{\left\langle s^{k-1}, s^{k-1}\right\rangle}, \alpha_{\max}\right\}\right\}, & \left\langle s^{k-1}, y_i^{k-1}\right\rangle > 0, \\ \max\left\{\alpha_{\min}, \min\left\{\frac{\left\|y_i^{k-1}\right\|}{\left\|s^{k-1}\right\|}, \alpha_{\max}\right\}\right\}, & \left\langle s^{k-1}, y_i^{k-1}\right\rangle < 0, \\ \alpha_{\min}, & \left\langle s^{k-1}, y_i^{k-1}\right\rangle = 0. \end{cases}$$

Step 2. Compute

$$\nabla \hat{f}_i(x^k) = \frac{1}{\alpha_i^k} \nabla f_i(x^k), i = 1, \ldots, m$$

$$\lambda^k = \arg\min_{\lambda \in \Delta_m} \frac{1}{2} \left\| \sum_{i=1}^m \lambda_i \nabla \hat{f}_i(x^k) \right\|^2$$

$$d_{BB}^k = -\sum_{i=1}^m \lambda_i^k \nabla \hat{f}_i(x^k)$$

Step 3.

$$\textbf{if} \quad \|d_{BB}^k\| < \varepsilon \quad \textbf{then} \quad \text{STOP}$$

else

$$\beta^k = \text{Armijo\_line\_search}\left(x^k, d_{BB}^k, JF\left(x^k\right)\right)$$

$$x^{k+1} = x^k + \beta^k d_{BB}^k$$

$k := k + 1$ and return to **Step1**.

The descent directions of all algorithms are determined by using the Frank-Wolfe method. We used the public code[1] provided by Sener & Koltun (2018). The parameters used in the SDMO, BB-DMO, and Armijo_line_search algorithms are taken similarly to those in Chen et al. (2023). Specifically, we set $\sigma = 10^{-4}, \gamma = 0.5, \alpha_{\min} = 10^{-3}, \alpha_{\max} = 10^3$. For NSDMO$\ell$ ($\ell = 1, 2, 3, 4$), we choose $\eta_0 = 0.99, \eta_1 = 0.98$. The convergent positive $\sum_{k=1}^{+\infty} \varepsilon_k$ is created by $\varepsilon_k = 0.9^k$ for NSDMO1, NSDMO3 and $\varepsilon_k = \frac{1.2(\ln k)^4}{k^{1.1}}$ for NSDMO2, NSDMO4. The initial stepsizes t_0 for all considered algorithms are

[1]https://github.com/isl-org/MultiObjectiveOptimization/blob/master/multi_task/min_norm_solvers_numpy.py

determined by using Armijo_line_search for the first iteration. In the sequel, for a set S, we denote $|S|$ for the cardinality of the set S.

For all tested algorithms we put $\varepsilon = 2 \times 10^{-8}$, the maximum number of iterations is limited by 500. All experiments were implemented in Python 3.11 and executed on a personal computer equipped with Intel Core i7-11370H, 3.30 GHz processor, and 16 GB of RAM.

We use two sets of problems for doing experiments as follows.

- Set 1: including 20 benchmark problems chosen from papers in literature from two to five objectives. The details are given in Table 1.

- Set 2: Problem (3.3) with 10 real datasets taken from Kaggle[2] in Table 2.

Problem	m	n	x_L	x_U	**Ref.**
SSFYY2	2	1	-100	100	(Shim et al., 2002)
PNR	2	2	$(-2,-2)^T$	$(2,2)^T$	(Preuss et al., 2006)
Hil	2	2	$(0,0)^T$	$(5,5)^T$	(Hillermeier, 2001)
FF1	2	2	$(-1,-1)^T$	$(1,1)^T$	(Fonseca & Fleming, 1995)
VU1	2	2	$(-3,-3)^T$	$(3,3)^T$	(Valenzuela-Rendón & Uresti-Charre, 1997)
Imbalance1	2	2	$(-2,-2)^T$	$(2,2)^T$	(Chen et al., 2023)
Imbalance2	2	2	$(-2,-2)^T$	$(2,2)^T$	(Chen et al., 2023)
SP1	2	2	$(-100,-100)^T$	$(100,100)^T$	(Sefrioui & Perlaux, 2000)
SD	2	4	$(1,\sqrt{2},\sqrt{2},1)^T$	$(3,3,3,3)^T$	(Stadler & Dauer, 1992)
DD1	2	5	$(-20,...,-20)^T$	$(20,...,20)^T$	(Das & Dennis, 1998)
JOS1	2	50	$(-50,...,-50)^T$	$(50,...,50)^T$	(Jin et al., 2001)
MHHM1	3	1	0	1	(Mao et al., 2000)
IKK1	3	2	$(-50,-50)^T$	$(50,50)^T$	(Ikeda et al., 2001)
AP1	3	2	$(-10,-10)^T$	$(10,10)^T$	(Ansary & Panda, 2015)
AP4	3	3	$(-10,-10,-10)^T$	$(10,10,10)^T$	(Ansary & Panda, 2015)
MGH26a	3	3	$(-1,-1,-1)^T$	$(1,1,1)^T$	(Moré et al., 1981)
FDS	3	10	$(-2,...,-2)^T$	$(2,...,2)^T$	(Fliege et al., 2009a)
TRIDIA2	4	4	$(-1,-1,-1,-1)^T$	$(1,1,1,1)^T$	Mita et al., 2019; Toint, 1983
MGH26b	4	4	$(-1,-1,-1,-1)^T$	$(1,1,1,1)^T$	(Moré et al., 1981)
MGH26c	5	5	$(-1,...,-1)^T$	$(1,...,1)^T$	(Moré et al., 1981)

Table 1: Set 1 of tested problems which include 20 benchmark examples given in the literature.

[2]https://www.kaggle.com/datasets

Dataset	Desciption	Samples (N)	Features (p)
Vehicle dataset (car) https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho/data?select=CAR+DETAILS+FROM+CAR+DEKHO.csv	Predicting car prices from information about used cars.	301	7
Vehicle dataset (car_v3) https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho/data?select=Car+details+v3.csv	Predicting car prices from information about used cars listed on different websites.	8128	10
Medical Cost Personal Datasets (insur) https://www.kaggle.com/datasets/mirichoi0218/insurance/data?select=insurance.csv	Predicting medical insurance charges from personal attributes.	1337	6
Housing Prices Dataset (house) https://www.kaggle.com/datasets/yasserh/housing-prices-dataset/data?select=Housing.csv	Predicting housing prices from a dataset of features.	545	14
Marketing Campaign (campaign) https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign/data?select=marketing_campaign.csv	Predicting the total amount spent using customer attributes and marketing campaign interactions.	2240	19
Real estate price prediction (price) https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction/data?select=Real+estate.csv	Predicting real estate prices from a dataset of property features.	414	5
White Wine Quality (quality) https://www.kaggle.com/datasets/piyushagni5/white-wine-quality/data?select=winequality-white.csv	Predicting white wine quality from chemical composition features.	4898	11
Ames Housing Dataset (ames_house) https://www.kaggle.com/datasets/shashanknecrothapa/ames-housing-dataset/data?select=AmesHousing.csv	Predicting house sale prices from residential property features in Ames.	2930	44
Salary_Data (salary) https://www.kaggle.com/datasets/mohithsairamreddy/salary-data/data?select=Salary_Data.csv	Predicting employee salary from personal and job attributes.	1787	3
E-Commerce Data (sales) https://www.kaggle.com/datasets/carrie1/ecommerce-data/data?select=data.csv	Predicting total invoice price from e-commerce transaction data.	540455	47

Table 2: Set 2 of tested problems which correspond to Problem (3.3) with 10 real datasets taken from Kaggle, $[x_L, x_U] = [0, 1]$

From the original datasets on Kaggle (as referenced in the link), we performed initial pre-processing steps, including handling missing values, converting date-time and categorical features into numerical form, and removing redundant columns. The number of samples (N) and features (p) corre-

sponding to each dataset are shown in Table 1. We then use StandardScale in sklearn.preprocessing to scale the dataset so that each feature has mean zero and standard deviation 1. For training purpose, we take 80% of data to construct the objectives of Problem (3.3).

The performance of considered algorithms are evaluated via the convergence speed and Pareto front quality. In particular, for each algorithm a, and problem p, we measure the **convergence speed** metrics throughout CPU time (Time), number of iterations (Iter.), CPU time per iteration (Time/Iter.), stepsize. The metrics for **Pareto front quality** include the number of nondominated points (# P), the purity, the hypervolume (HV) and modified inverted generational distance (IGD^+).

Except for the number of nondominated points, each metric is defined by taking the average result over 200 runs. Each run corresponds to an initial point x^0 randomly generated in $[x_L, x_U]$ by uniform distribution, where $x_L, x_U \in \mathbb{R}^n$, $x_L \leq x_U$, are provided in advance for each tested problem (in Tables 1 and 2).

For the way to compute the evaluation metrics, it is easy to export the convergence speed ones as stopping criterion is satisfied. For the quality of Pareto front metrics, let us describe the computation in more detail. Let a be a an algorithm belonging to the set of considered algorithms A, p be a problem belonging to Set 1 or Set 2.

- The set of Pareto points $F_{a,p}$ of algorithm a in solving problem p is the nondominated points founded from 200 output points in the value space. Then *the number of nondominated points* obtained by algorithm a for problem p is $|F_{a,p}|$. The higher number of obtained nondominated points indicates the better convergence of the method.

- Let F_p be the set of nondominated points of $\cup_{a\in A} F_{a,p}$. Then *the purity* (Morovati et al. (2016, 2018)) of algorithm a for problem p is defined by $\dfrac{|F_{a,p} \cap F_p|}{|F_p|}$. The higher purity reflects the higher percentage for a nondominated point obtained by algorithm a being a Pareto point of problem p.

- Let $\mathbf{r}^p \in \mathbb{R}^m$ be a reference point of $\cup_{a\in A} F_{a,p}$ which is determined as a point in outcome space dominated by all members of $\cup_{a\in A} F_{a,p}$. In the implementation we choose this point as follows

$$\mathbf{r}^p_j = \max\{y_j \mid y \in \cup_{a\in A} F_{a,p}\} + 1, \ \forall j = 1, ..., m.$$

  The hypervolume (Zitzler & Thiele (1998)) of algorithm a for problem p is then defined by the volume of the union of all hyper-rectangles where the diagonal of each hyper-rectangle is the line segment connecting r^p to some nondominated point in $F_{a,p}$, i.e., the volume of the copolyblock $\cup_{y\in F_{a,p}}[y, \mathbf{r}^p]$. The concepts of polyblock or copolyblock are used for the representation of search region in multiobjective programming in Hoai et al., 2021. The hypervolume metric is known as a Pareto-compliant indictor where the bigger value of hypervolume provides the better dominance.

- *The modified inverted generational distance* (Ishibuchi et al. (2015)) obtained by algorithm a for

problem p is the average modified distance from a point z in the reference Pareto front F_p to the nondominated set $F_{a,p}$. It is denoted by $\text{IGD}^+(F_{a,p})$ and computed by the following formulas.

$$\text{IGD}^+(F_{a,p}) = \frac{1}{|F_p|}\sum_{z\in F_p}\min_{y\in F_{a,p}}\sqrt{\sum_{j=1}^{m}(\max\{y_j - z_j, 0\})^2}.$$

This indicator is proved to be weakly Pareto compliant (for details, see e.g., Ishibuchi et al. (2015)) where the lower value indicates the better approximation.

We exported the tables of all considered metrics for the two sets of tested problems in Tables 3, 4 and 5 in Appendix of this paper. To have a general visualization quickly, we additionally summarize all results by using the concept of Performance profiles (Dolan & Moré (2002)). More precisely, suppose that $t_{a,p}$ is the metric derived by running algorithm a for problem p. And the smaller $t_{a,p}$ is, the better performance is. We first compute the performance ratio

$$r_{a,p} = \frac{t_{a,p}}{\min\{t_{a,p} \mid a \in A\}}$$

then the performance profiles for algorithm a in solving problem $p \in$ Set i, $i = 1, 2$, is the following

$$\rho_a(\tau) = \frac{1}{|\text{ Set } i|}\big|p \in \text{ Set } i | r_{a,p} \leq \tau\big|,$$

where $\tau \geq 1$. One can see that the main difference between Set 1 and Set 2 is in the used data, one is benchmark examples with synthetic data and one is an applicative example with real data. We would like to investigate the performance of the considered algorithms for each set of problems separately to see how the performance depends on the synthetic/real input data.

Now we are ready to analyze the Performance profiles for Set 1 and Set 2 in Figure 1-Figure 8 and Figure 9-Figure 16, respectively.

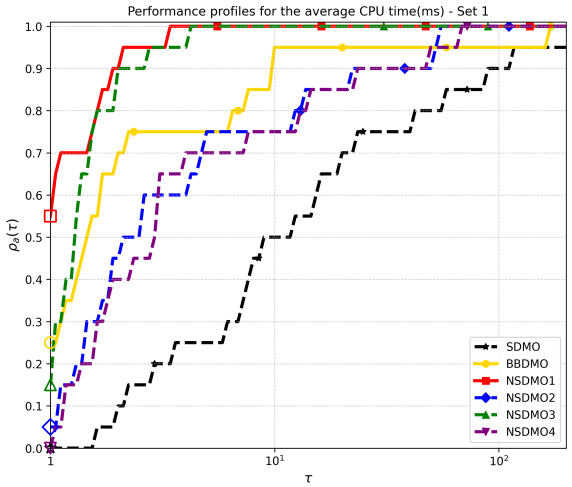

Figure 1: Performance profiles for the average CPU time(ms) of Set 1

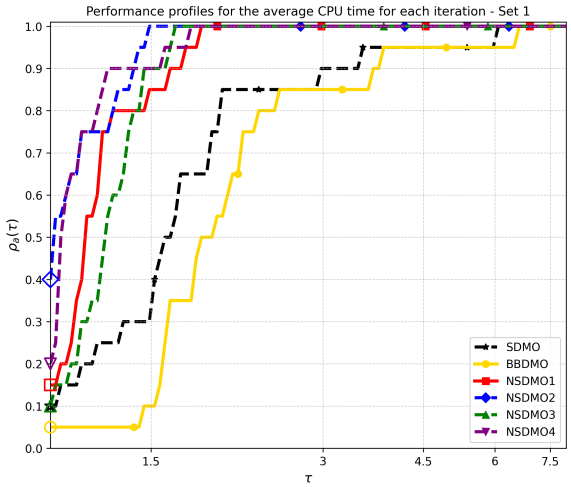

Figure 2: Performance profiles for the average CPU time for each iteration of Set 1

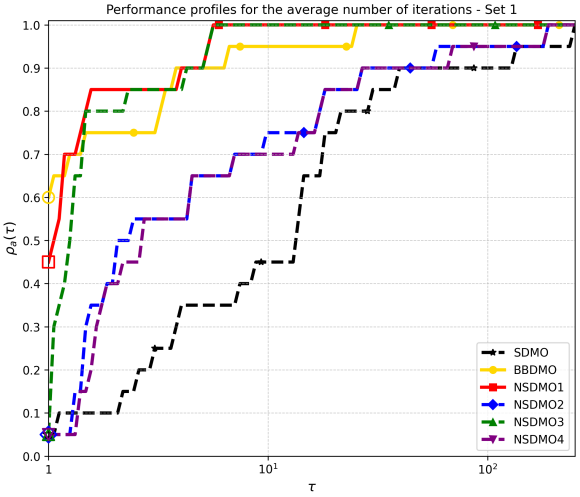

Figure 3: Performance profiles for the average number of iterations of Set 1

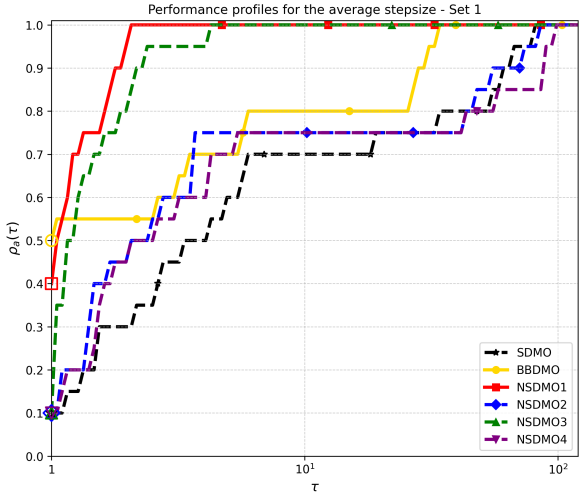

Figure 4: Performance profiles for the average stepsize of Set 1

We first discuss the convergence speed of all methods when solving for 20 problems in Set 1. It can be observed from Figure 1 and Figure 2 that our proposed algorithms, particularly NSDMO1, achieve the best performance in terms of CPU time. Moreover, all four proposed algorithms outperform SDMO and BBDMO with respect to the per-iteration CPU time. In addition, Table 3 and Table 4 indicate that some instances, such as SSFYY2, SP1, FDS, and TRIDIA2, our methods exhibit particularly favorable CPU times.

From Figure 3 and Figure 4, we see that the the method with longer stepsize needs fewer the number of iterations. Particularly, for the number of iterations and the stepsize, BBDMO and NSDMO1 share the first and second ranks for $\rho_a(1)$ while NSDMO1 and NSDMO 3 provide the best results $\rho_a(\tau)$ when τ increases. Although the result of iteration counts obtained by BBDMO for Set 1 is significantly good but the cost for each iteration is expensive due to the line search computation so that it is not fast as NSDMO1 for most of cases.

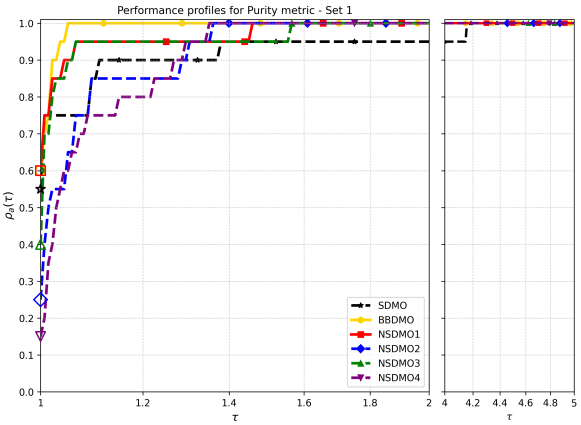

Figure 5: Performance profiles for Purity metric of Set 1

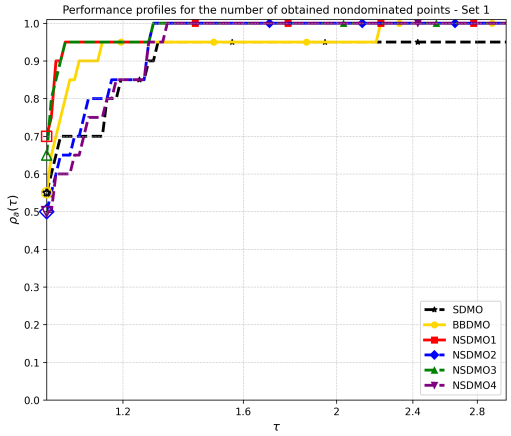

Figure 6: Performance profiles for the number of obtained nondominated points of Set 1

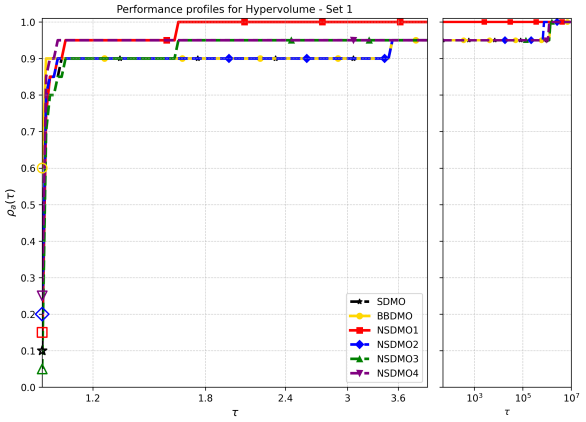

Figure 7: Performance profiles for Hypervolume of Set 1

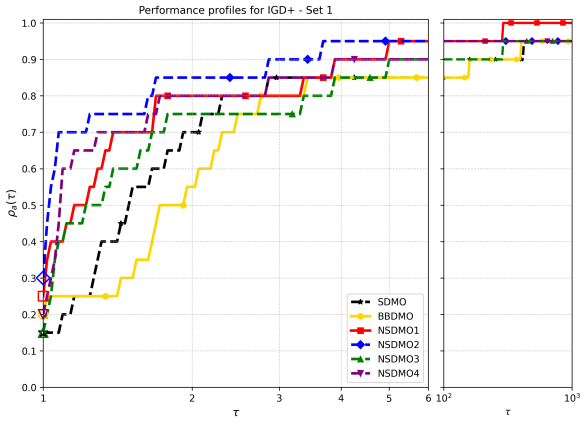

Figure 8: Performance profiles for IGD^+ of Set 1

Figure 5 to Figure 8 present the performance indicators computed from the sets of solutions obtained by the six methods. Although none of our proposed methods consistently stand out across all four indicators, the overall results remain competitive. As illustrated in Figure 5, Figure 6 and supported by the tabulated results, NSDMO1 demonstrates its advantage by producing a larger number of nondominated points and achieving better purity values on test problems, such as TRIDIA2. The HV results of NSDMO1 and NSDMO4, shown in Figure 7, further indicate their ability to identify distant nondominated points, thereby providing better coverage. In addition, the IGD^+ values in Figure 8 reveal that our algorithms, particularly NSDMO2, exhibit a diverse distribution of solutions that approximates the ideal Pareto front.

In Set 1, the overall performance of our proposed algorithms is promising. In particular, NSDMO1 exhibits outstanding results across most evaluation indicators, highlighting its computational efficiency in terms of CPU time, the number of iterations, and the quality of the obtained solution sets.

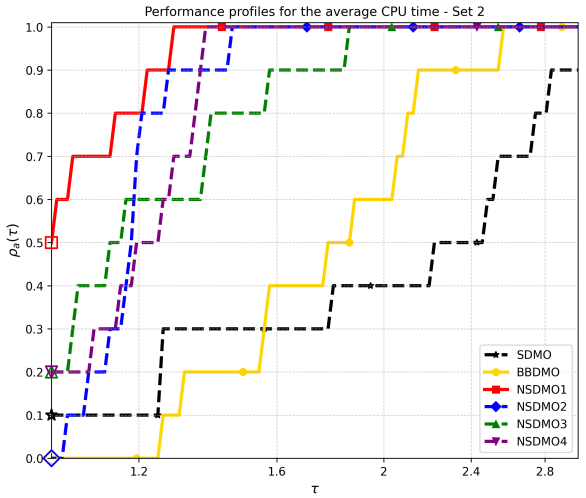

Figure 9: Performance profiles for the average CPU time of Set 2

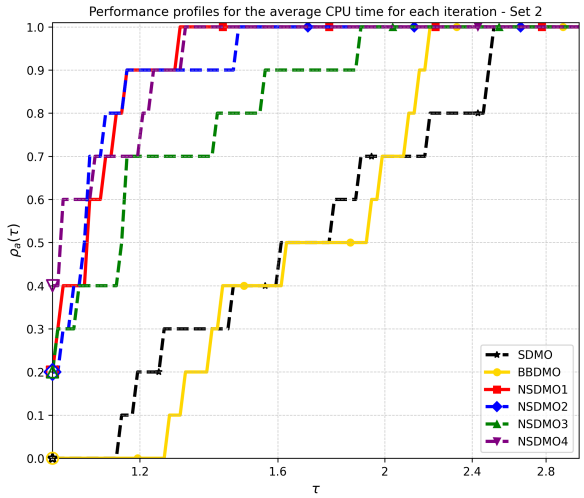

Figure 10: Performance profiles for the average CPU time for each iteration of Set 2

The results on CPU time and CPU time per iteration for the datasets in Set 2 are presented in Figure 9 and Figure 10. Our method continue to show its advantage in terms of computational efficiency, with NSDMO1 performing particularly well on both indicators. The numerical results in Table 5 further confirm this observation.

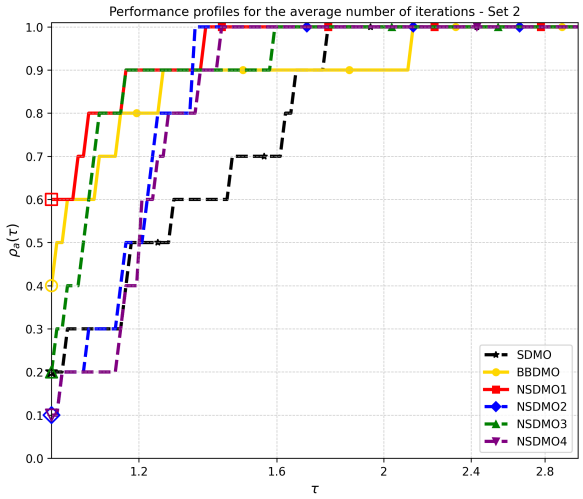

Figure 11: Performance profiles for the average number of iterations of Set 2

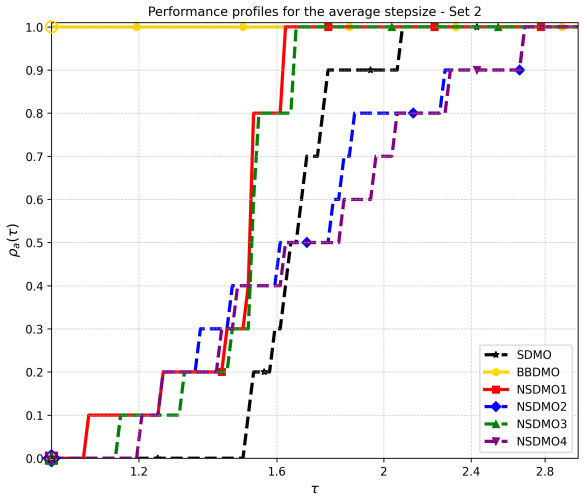

Figure 12: Performance profiles for the average stepsize of Set 2

NSDMO1 demonstrates a lower average number of iterations across a larger set of problem instances, as illustrated in Figure 11, with notable cases including *price* and *quality*. With respect to the stepsize, BBDMO retains providing the biggest stepsize for ten data sets that partly helps the iteration counts of this method stand at the second rank as in Figure 11. Nevertheless, the CPU time per iteration of BBDMO is long (Figure 10) which causes the largest running time of this method as evidenced by Table 5 and Figure 9.

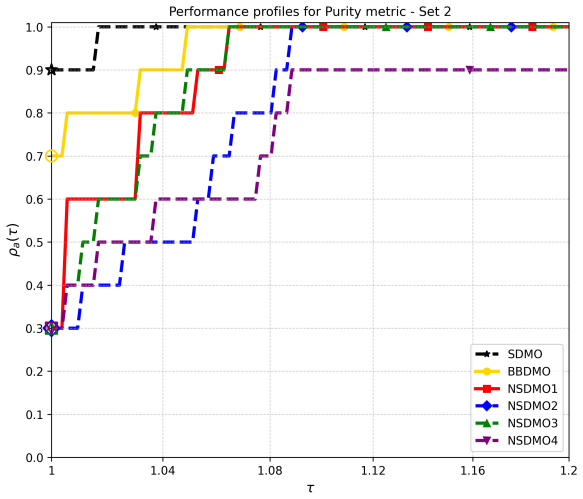

Figure 13: Performance profiles for Purity metric of Set 2

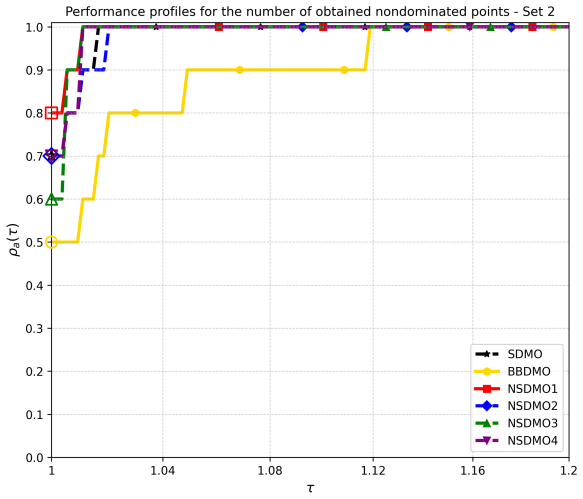

Figure 14: Performance profiles for the number of obtained nondominated points of Set 2

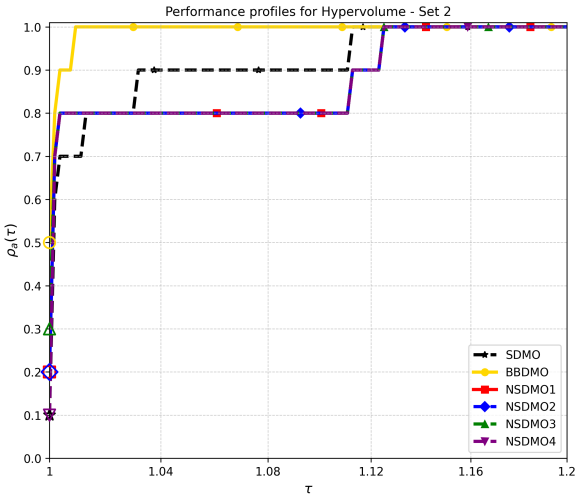

Figure 15: Performance profiles for Hypervolume of Set 2

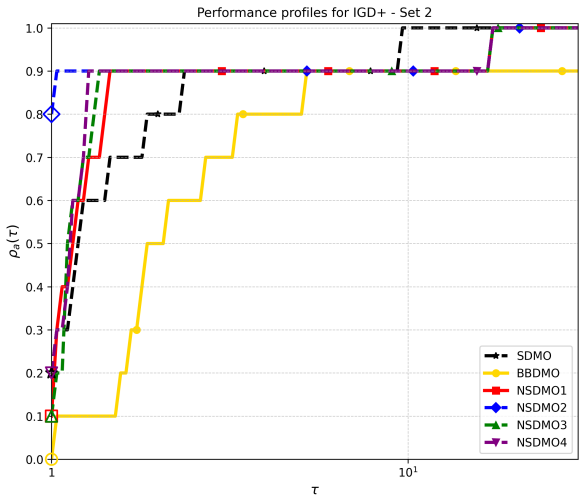

Figure 16: Performance profiles for IGD$^+$ of Set 2

The quality indicators of the obtained solution sets in Set 2 do not show that any of our methods are particularly outstanding across all metrics. As shown in Figure 13 and Figure 15, the purity metric and hypervolume values of SDMO and BBDMO are notably superior to those of our algorithms. However, the steep lines given in Figure 15 with very small τ reflect the tiny gap among the hypervolume obtained by different methods. Figure 14 indicates that NSDMO1 is able to find a larger number of nondominated points in most datasets. The IGD$^+$ results in Figure 16 show that NSDMO4 performs well, demonstrating that its obtained solution sets are of high quality and closely approximate the ideal Pareto front. Overall, the results of Set 2 indicate that our methods still perform reasonably well in terms of CPU time, iterations, the number of nondominated solutions and competitively for other metrics.

Based on the numerical results obtained from Set 1 and Set 2, it can be observed that our algorithms perform effectively. In particular, NSDMO1 achieves considerably better CPU times, the average number of iterations, the number of nondominated points for both of synthetic data and real data. Other indicators, such as CPU time per iteration, purity, hypervolum (HV), IGD$^+$, also show that our methods are competitive.

6. Conclusions

In this paper, we propose four adaptive strategies for selecting the stepsize used in the steepest descent method to solve unconstrained multiobjective optimization problems. The resulting algorithms are evaluated with respect to both theoretical and computational aspects. Particularly, under the globally Lipschitz continuity condition imposed on the gradients of all objectives, we derive a computational complexity of $O(1/\sqrt{K})$ for the min-norm of the steepest descent direction from a fixed iteration. This rate is proved for a wide class of nonconvex problems (*quacavex* functions). In

the case of convex and strongly convex objective functions, we show sublinear and linear convergence rates for the iterates generated by our proposed algorithms. The reported numerical results further demonstrate the efficient performance of our new stepsize selection strategies. Typically, our new algorithm NSDMO1 costs the cheapest in CPU time and provides the largest number of nondominated points across almost all tested instances. Future research can focus on accelerated or stochastic versions of the proposed methods.

Acknowledgements

The authors would like to express their gratitude to the editors and reviewers for the useful comments which greatly help to improve the quality of the paper. They also thank Professor Jian Chen for his helpful comments on our Python codes of the algorithms and results presented in his joint work Chen et al. (2023).

References

Ansary, M., & Panda, G. (2015). A modified quasi-newton method for vector optimization problem. *Optimization*, *64*, 2289–2306. doi:10.1080/02331934.2014.947500.

Barzilai, J., & Borwein, J. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, *8*, 141–148.

Burachik, R., Drummond, L. M. G., Iusem, A. N., & Svaiter, B. F. (2010). Full convergence of the steepest descent method with inexact line searches. *Optimization*, *59*, 137–146. URL: https://doi.org/10.1080/02331939508844042. doi:10.1080/02331939508844042.

Chen, J., Tang, L., & Yang, X. (2023). A barzilai-borwein descent method for multiobjective optimization problems. *European Journal of Operational Research*, *311*, 196–209. doi:10.1016/j.ejor.2023.04.026.

Cruz, J. Y. B., Pérez, L. R. L., & Melo, J. G. (2012). Convergence of the projected gradient method for quasiconvex multiobjective optimization. *Nonlinear Analysis: Theory, Methods & Applications*, *75*, 1232–1243. URL: https://doi.org/10.1016/j.na.2011.04.067. doi:10.1016/j.na.2011.04.067.

Das, I., & Dennis, J. E. (1998). Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, *8*, 631–657. doi:10.1137/S1052623496307510.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*, 201–213.

Drummond, L. M. G., & Iusem, A. N. (2004). A projected gradient method for vector optimization problems. *Computational Optimization and Applications*, *28*, 5–29. URL: https://link.springer.com/article/10.1023/B:COAP.0000018877.86161.8b. doi:10.1023/B:COAP.0000018877.86161.8b.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer-Verlag Berlin Heidelberg. URL: https://link.springer.com/book/10.1007/3-540-27659-9. doi:10.1007/3-540-27659-9.

Evans, G. (1984). Overview of techniques for solving multiobjective mathematical programs. *Management Science*, *30*, 1268–1282. doi:10.1287/mnsc.30.11.1268.

Fliege, J., Drummond, L. M. G. n., & Svaiter, B. F. (2009a). Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, *20*, 602–626. doi:10.1137/08071692X.

Fliege, J., Grana Drummond, L. M., & Svaiter, B. F. (2009b). Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, *20*, 602–626. doi:10.1137/080730123.

Fliege, J., & Svaiter, B. F. (2000). Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, *51*, 479–494. doi:10.1007/s001860000077.

Fliege, J., Vaz, A. I. F., & Vicente, L. N. (2019). Complexity of gradient descent for multiobjective optimization. *Optimization Methods and Software*, *34*, 949–959. doi:10.1080/10556788.2018.1510928.

Fliege, J., & Werner, R. (2014). Robust multiobjective optimization & applications in portfolio optimization. *European Journal of Operational Research*, *234*, 422–433. URL: https://www.sciencedirect.com/science/article/pii/S0377221713008515. doi:https://doi.org/10.1016/j.ejor.2013.10.028. 60 years following Harry Markowitz's contribution to portfolio theory and operations research.

Fonseca, C. M., & Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, *3*, 1–16. doi:10.1162/evco.1995.3.1.1.

Fukuda, E. H., & Drummond, L. M. G. (2014). A survey on multiobjective descent methods. *Pesquisa Operacional*, *34*, 585–620. doi:10.1590/0101-7438.2014.034.03.0585.

Gravel, M., Martel, J. M., Nadeau, R., Price, W., & Tremblay, R. (1992). A multicriterion view of optimal resource allocation in job-shop production. *European Journal of Operational Research*, *61*, 230–244. doi:10.1016/0377-2217(92)90284-G.

Hillermeier, C. (2001). Generalized homotopy approach to multiobjective optimization. *Journal of Optimization Theory and Applications*, *110*, 557–583. doi:https://doi.org/10.1023/A:1017536311488.

Hoai, P., Le Thi, H. A., & Nam, N. C. (2021). Half-open polyblock for the representation of the search region in multiobjective optimization problems: its application and computational aspects. *4OR-Q. J. Oper. Res.*, *19*, 41–70.

Hoai, P., Vinh, N., & Chung, N. (2024). A novel stepsize for gradient descent method. *Operations Research Letters*, (p. 107072). doi:https://doi.org/10.1016/j.orl.2024.107072.

Ikeda, K., Kita, H., & Kobayashi, S. (2001). Failure of pareto-based moeas: Does non-dominated really mean near to optimal? *IEEE Congress on Evolutionary Computation (CEC)*, *2*, 957–962. doi:10.1109/CEC.2001.934293.

Ishibuchi, H., Masuda, H., Tanigaki, Y., & Nojima, Y. (2015). Modified distance calculation in generational distance and inverted generational distance. In *Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization, Part I* (pp. 110–125). Guimarães, Portugal.

Jin, Y., Olhofer, M., & Sendhoff, B. (2001). Dynamic weighted aggregation for evolutionary multiobjective optimization: Why does it work and how? *Genetic and Evolutionary Computation Conference (GECCO)*, (pp. 1042–1049).

Leschine, T. M., Wallenius, H., & Verdini, W. A. (1992). Interactive multiobjective analysis and assimilative capacity-based ocean disposal decisions. *European Journal of Operational Research*, *56*, 278–289. doi:10.1016/0377-2217(92)90228-2.

Luc, D. T. (1989). *Theory of Vector Optimization* volume 319 of *Lecture Notes in Economics and Mathematical Systems*. Berlin: Springer.

Mao, J., Hirasawa, K., Hu, J., & Murata, J. (2000). Genetic symbiosis algorithm for multiobjective optimization problem. *Robot and Human Interactive Communication (RO-MAN 2000)*, (pp. 137–142). doi:10.1109/ROMAN.2000.892484.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*, 369–395. doi:10.1007/s00158-003-0368-6.

Mita, K., Fukuda, E. H., & Yamashita, N. (2019). Nonmonotone line searches for unconstrained multiobjective optimization problems. *Journal of Global Optimization*, *75*, 63–90. doi:10.1007/s10898-019-00802-0.

Moré, J. J., Garbow, B. S., & Hillstrom, K. E. (1981). Testing unconstrained optimization software. *ACM Trans. Math. Softw.*, *7*, 17–41. doi:10.1145/355934.355936.

Morovati, V., Basirzadeh, H., & Pourkarimi, L. (2018). Quasi-Newton methods for multiobjective optimization problems. *4OR - A Quarterly Journal of Operations Research*, *16*, 261–294. doi:10.1007/s10288-017-0363-1.

Morovati, V., Pourkarimi, L., & Basirzadeh, H. (2016). Barzilai and borwein's method for multiobjective optimization problems. *Numerical Algorithms*, *72*, 539–604. URL: https://doi.org/10.1007/s11075-015-0058-7. doi:10.1007/s11075-015-0058-7.

Preuss, M., Naujoks, B., & Rudolph, G. (2006). Pareto set and emoa behavior for simple multimodal multiobjective functions. *Parallel Problem Solving from Nature (PPSN IX)*, (pp. 513–522). doi:10.1007/11844297_52.

Sefrioui, M., & Perlaux, J. (2000). Nash genetic algorithms: examples and applications. *Congress on Evolutionary Computation (CEC)*, *1*, 509–516. doi:10.1109/CEC.2000.870339.

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

Shim, M.-B., Suh, M.-W., Furukawa, T., Yagawa, G., & Yoshimura, S. (2002). Pareto-based continuous evolutionary algorithms for multiobjective optimization. *Engineering Computations*, *19*, 22–48. doi:10.1108/02644400210413649.

Stadler, W., & Dauer, J. (1992). Multicriteria optimization in engineering: A tutorial and survey. In *Structural Optimization: Status And Promise* chapter 10. (pp. 209–249). URL: https://arc.aiaa.org/doi/abs/10.2514/5.9781600866234.0209.0249. doi:10.2514/5.9781600866234.0209.0249.

Tapia, M. G. C., & Coello, C. A. C. (2007). Applications of multi-objective evolutionary algorithms in economics and finance: A survey. In *2007 IEEE Congress on Evolutionary Computation* (pp. 532–539). doi:10.1109/CEC.2007.4424516.

Toint, P. L. (1983). Test problems for partially separable optimization and results for the routine pspmin. *The University of Namur, Department of Mathematics, Belgium, Tech. Rep*, .

Valenzuela-Rendón, M., & Uresti-Charre, E. (1997). A non-generational genetic algorithm for multiobjective optimization. *International Conference on Genetic Algorithms*, . URL: https://api.semanticscholar.org/CorpusID:1767681.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In A. E. Eiben et al. (Eds.), *Parallel Problem Solving from Nature (PPSN V)* (pp. 292–301). Berlin, Germany: Springer.

Appendix

The list of 20 test problems included in Set 1 is as follows:

1. **SSFYY2** (Shim et al., 2002)

$$\min_x \begin{cases} f_1(x) = 10 + x^2 - 10\cos(x\frac{\pi}{2}), \\ f_2(x) = (x-4)^2. \end{cases}$$

2. **PNR** (Preuss et al., 2006)

$$\min_x \begin{cases} f_1(x) = x_1^4 + x_2^4 - x_1^2 + x_2^2 - 10x_1x_2 + 0.25x_1 + 20, \\ f_2(x) = (x_1-1)^2 + x_2^2. \end{cases}$$

3. **Hil** (Hillermeier, 2001)

$$\min_x \begin{cases} f_1(x) = \cos(a(x)).b(x), \\ f_2(x) = \sin(a(x)).b(x), \end{cases}$$

with

$$\begin{cases} a(x) = \frac{2\pi}{360}\left(45 + 40\sin(2\pi x_1) + 25\sin(2\pi x_2)\right), \\ b(x) = 1 + 0.5\cos(2\pi x_1). \end{cases}$$

4. **FF1** (Fonseca & Fleming, 1995)

$$\min_x \begin{cases} f_1(x) = 1 - e^{-(x_1-1)^2-(x_2+1)^2}, \\ f_2(x) = 1 - e^{-(x_1+1)^2-(x_2-1)^2}. \end{cases}$$

5. **VU1** (Valenzuela-Rendón & Uresti-Charre, 1997)

$$\min_x \begin{cases} f_1(x) = \frac{1}{x_1^2+x_2^2+1} \\ f_2(x) = x_1^2 + 3x_2^2 + 1. \end{cases}$$

6. **Imbalance1** (Chen et al., 2023)

$$\min_x \begin{cases} f_1(x) = 0.1x_1^2 + 10x_2^2, \\ f_2(x) = (x_1-50)^2 + 100(x_2+50)^2. \end{cases}$$

7. **Imbalance2** (Chen et al., 2023)

$$\min_x \begin{cases} f_1(x) = x_1^2 + x_2^2, \\ f_2(x) = 100(x_1-50)^2 + 100(x_2+50)^2. \end{cases}$$

8. **SP1** (Sefrioui & Perlaux, 2000)

$$\min_x \begin{cases} f_1(x) = (x_1 - 1)^2 + (x_1 - x_2)^2, \\ f_2(x) = (x_2 - 3)^2 + (x_1 - x_2)^2. \end{cases}$$

9. **SD** (Stadler & Dauer, 1992)

$$\min_x \begin{cases} f_1(x) = 2x_1 + \sqrt{2}x_2 + \sqrt{2}x_3 + x_4, \\ f_2(x) = \frac{2}{x_1} + \frac{2\sqrt{2}}{x_2} + \frac{2\sqrt{2}}{x_3} + \frac{2}{x_4}. \end{cases}$$

10. **DD1** (Das & Dennis, 1998)

$$\min_x \begin{cases} f_1(x) = \sum_{i=1}^{5} x_i^2, \\ f_2(x) = 3x_1 + 2x_2 - \frac{x_3}{3} + 0.01(x_4 - x_5)^3. \end{cases}$$

11. **JOS1** (Jin et al., 2001)

$$\min_x \begin{cases} f_1(x) = \frac{1}{n}\sum_{i=1}^{n} x_i^2, \\ f_2(x) = \frac{1}{n}\sum_{i=1}^{n} (x_i - 2)^2. \end{cases}$$

12. **MHHM1** (Mao et al., 2000)

$$\min_x \begin{cases} f_1(x) = (x - 0.8)^2, \\ f_2(x) = (x - 0.85)^2, \\ f_3(x) = (x - 0.9)^2. \end{cases}$$

13. **IKK1** (Ikeda et al., 2001)

$$\min_x \begin{cases} f_1(x) = x_1^2, \\ f_2(x) = (x_1 - 20)^2, \\ f_3(x) = x_2^2. \end{cases}$$

14. **AP1** (Ansary & Panda, 2015)

$$\min_x \begin{cases} f_1(x) = \frac{1}{4}((x_1 - 1)^4 + 2(x_2 - 2)^4), \\ f_2(x) = e^{(x_1 + x_2)/2} + x_1^2 + x_2^2, \\ f_3(x) = \frac{e^{-x_1} + 2e^{-x_2}}{6}. \end{cases}$$

15. **AP4** (Ansary & Panda, 2015)

$$\min_x \begin{cases} f_1(x) = \frac{1}{9}((x_1-1)^4 + 2(x_2-2)^4 + 3(x_3-3)^4), \\ f_2(x) = e^{(x_1+x_2+x_3)/3} + x_1^2 + x_2^2 + x_3^2, \\ f_3(x) = \frac{3e^{-x_1}+4e^{-x_2}+3e^{-x_3}}{12}. \end{cases}$$

16. **MGH26a** (Moré et al., 1981)

$$\min_x f_j(x) = 3 - \sum_{i=1}^{3} \cos(x_i) + j(1-\cos(x_j)) - \sin(x_j) \quad \text{with } j = 1,2,3.$$

17. **FDS** (Fliege et al., 2009a)

$$\min_x \begin{cases} f_1(x) = \dfrac{1}{n^2} \sum\limits_{i=1}^{n} i(x_i - i)^4, \\ f_2(x) = \exp(\sum\limits_{i=1}^{n} \dfrac{x_i}{n}) + \|x\|_2^2, \\ f_3(x) = \dfrac{1}{n(n+1)} \sum\limits_{i=1}^{n} i(n-i+1)e^{-x_i}. \end{cases}$$

18. **TRIDIA2** (Mita et al., 2019; Toint, 1983)

$$\min_x \begin{cases} f_1(x) = (2x_1 - 1)^2 + x_2^2, \\ f_2(x) = 2(2x_1 - x_2)^2 - x_1^2 + 2x_2^2, \\ f_3(x) = 3(2x_2 - x_3)^2 - 2x_2^2 + 3x_3^2, \\ f_4(x) = 4(2x_3 - x_4)^2 - 3x_3^2. \end{cases}$$

19. **MGH26b** (Moré et al., 1981)

$$\min_x f_j(x) = 4 - \sum_{i=1}^{4} \cos(x_i) + j(1-\cos(x_j)) - \sin(x_j), \quad j = 1,2,3,4.$$

20. **MGH26c** (Moré et al., 1981)

$$\min_x f_j(x) = 5 - \sum_{i=1}^{5} \cos(x_i) + j(1-\cos(x_j)) - \sin(x_j) \quad j = 1,2,3,4,5.$$

In the following, there are some representative figures illustrating the solution sets for test problems in Set 1 and the results obtained on several datasets in Set 2.

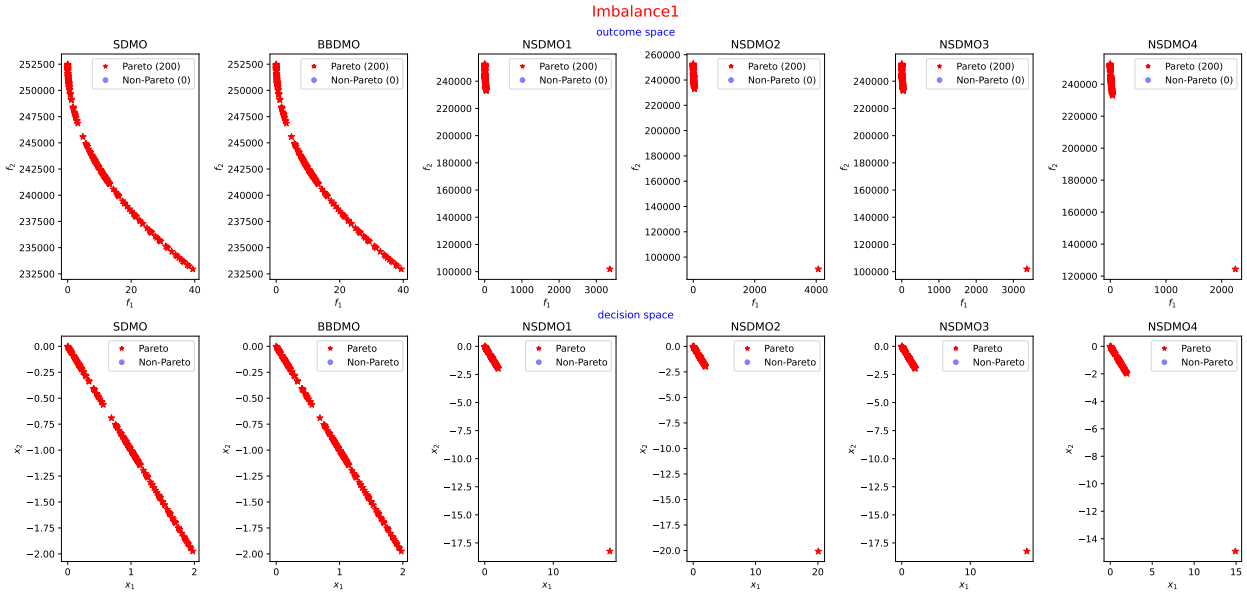

Figure 17: Imbalance1 Problem. Our methods can find distant Pareto points.

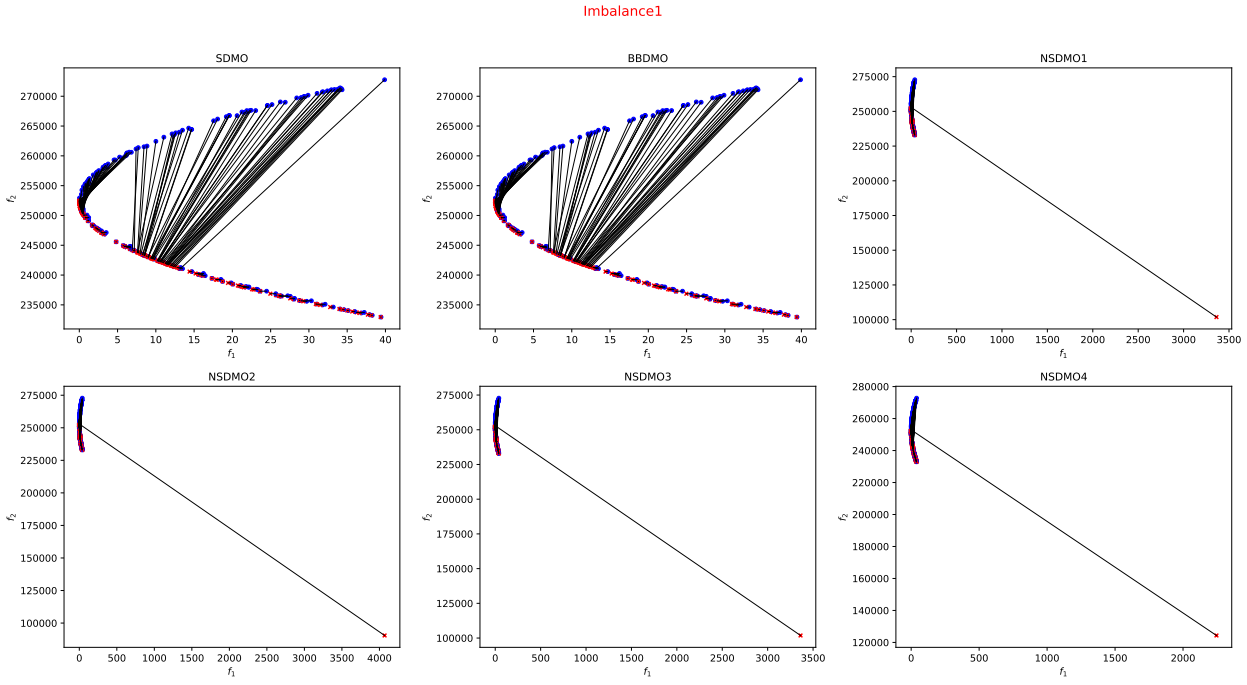

Figure 18: Imbalance1 Problem - describing the blue initial point and the red resulting point in the objective space.

From the Pareto front of Problem Imbalance1, shown in Figure 17 respectively, it is evident that our algorithms are able to find solutions located farther away, which leads to better HV and IGD$^+$ values, as reported in Table 3.

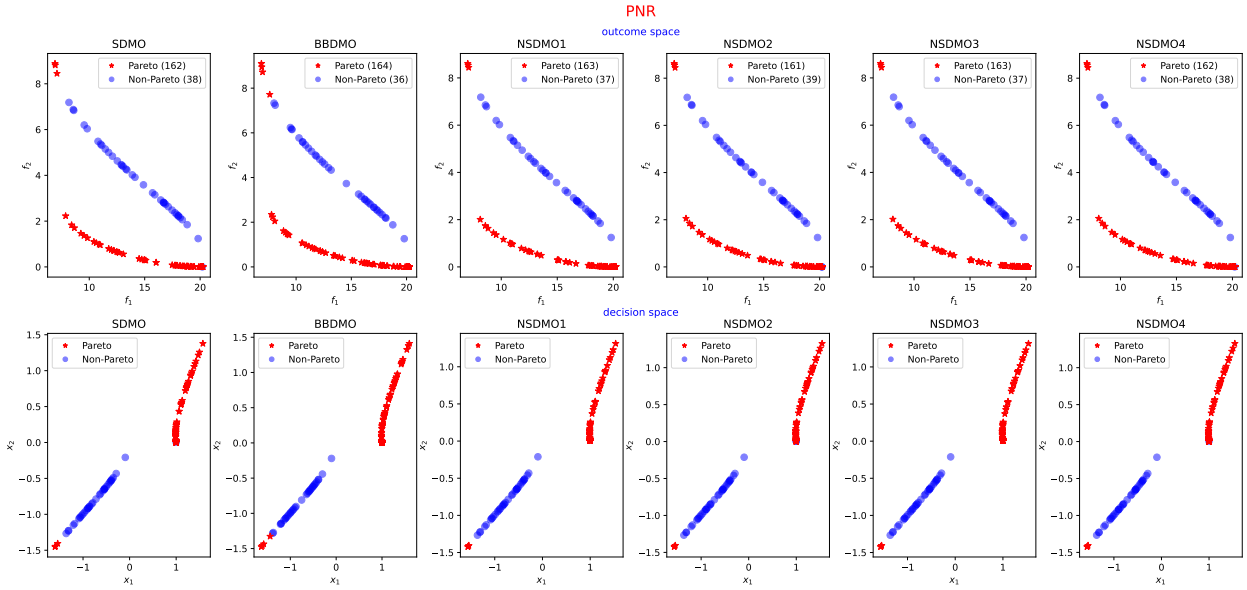

Figure 19: PNR Problem

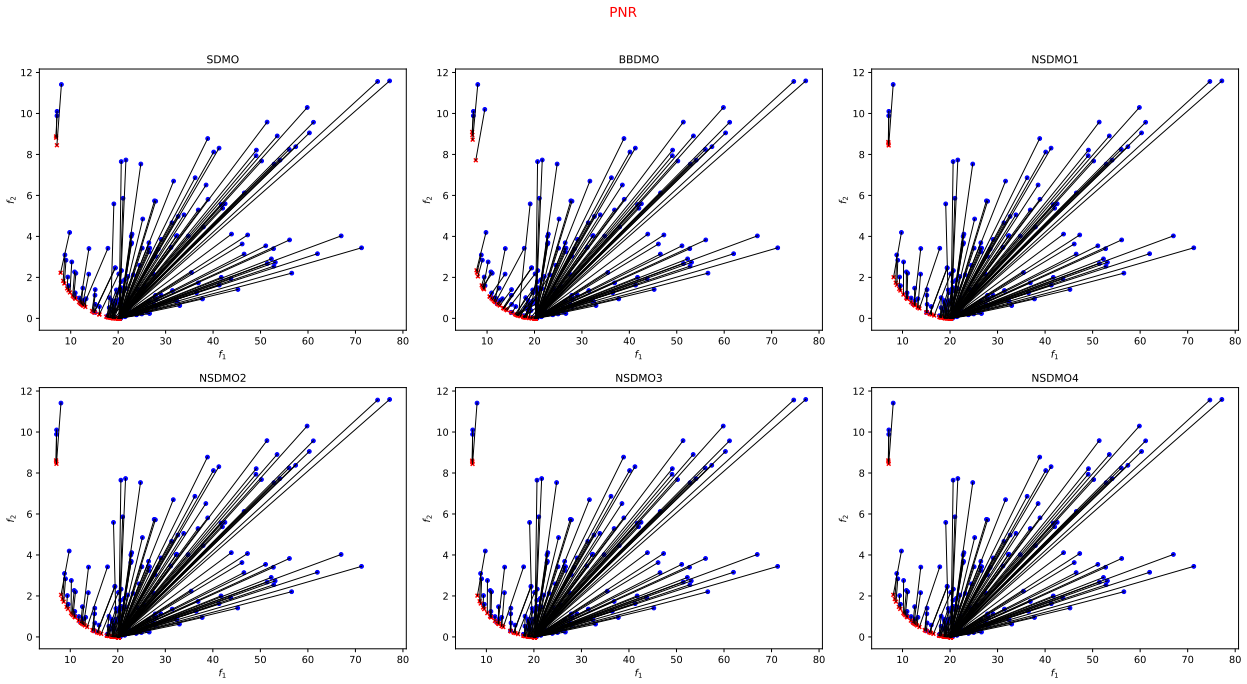

Figure 20: PNR Problem - describing the blue initial point and the red resulting point in the objective space.

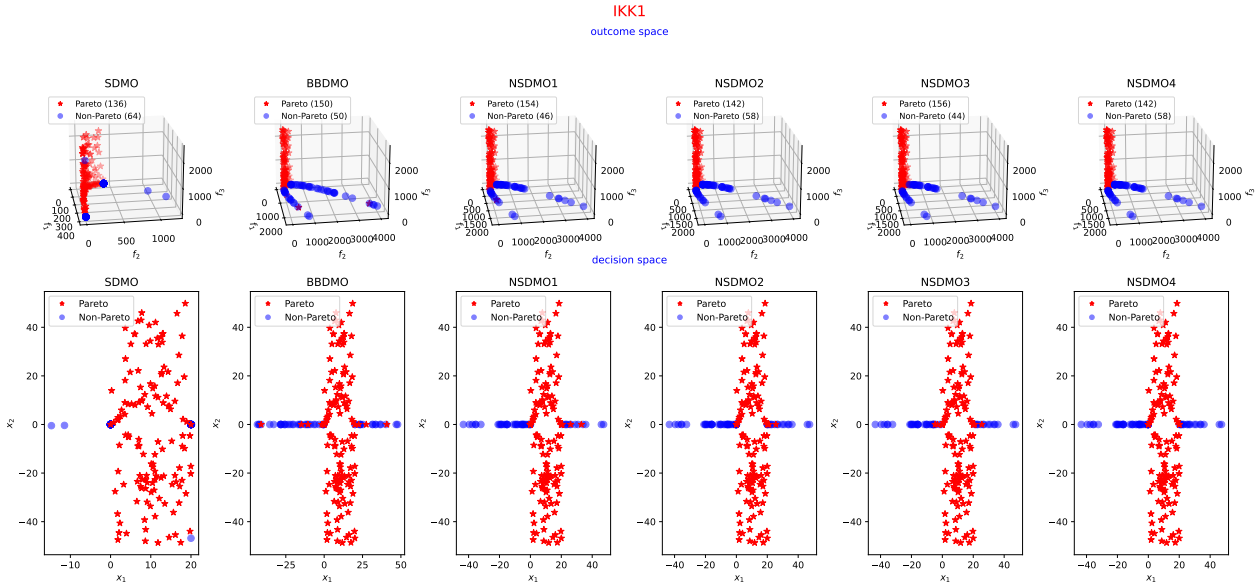

Figure 21: IKK1 Problem

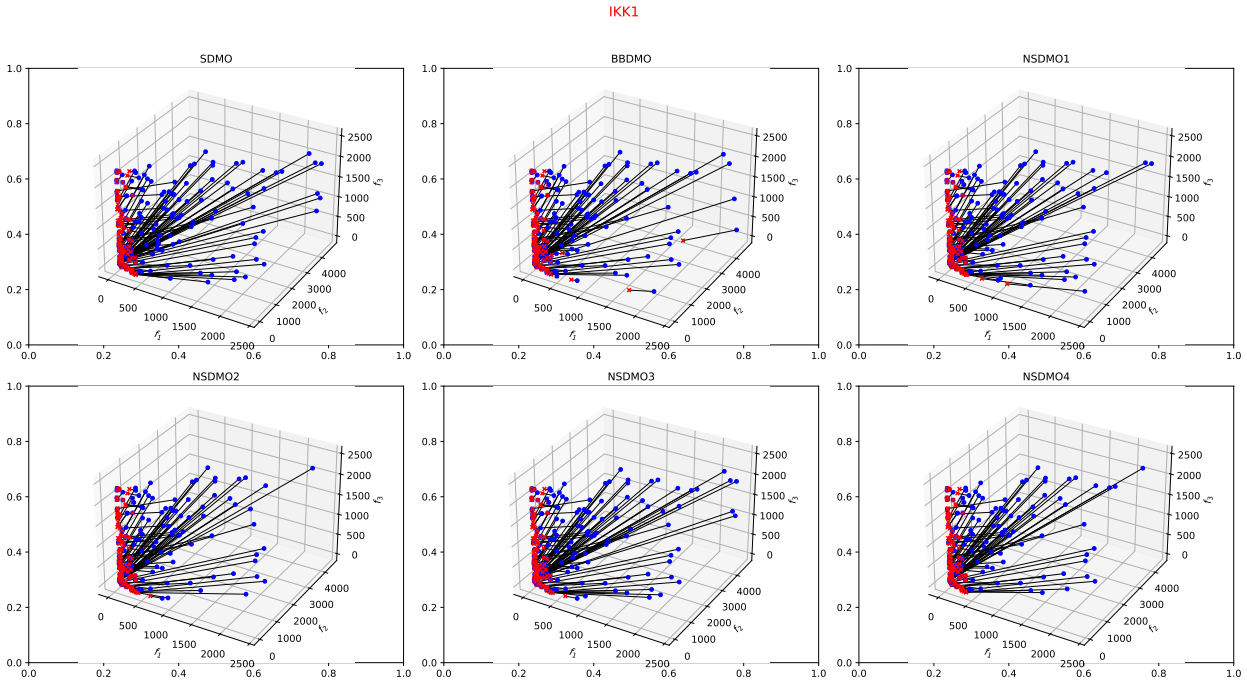

Figure 22: IKK1 Problem - describing the blue initial point and the red resulting point in the objective space.

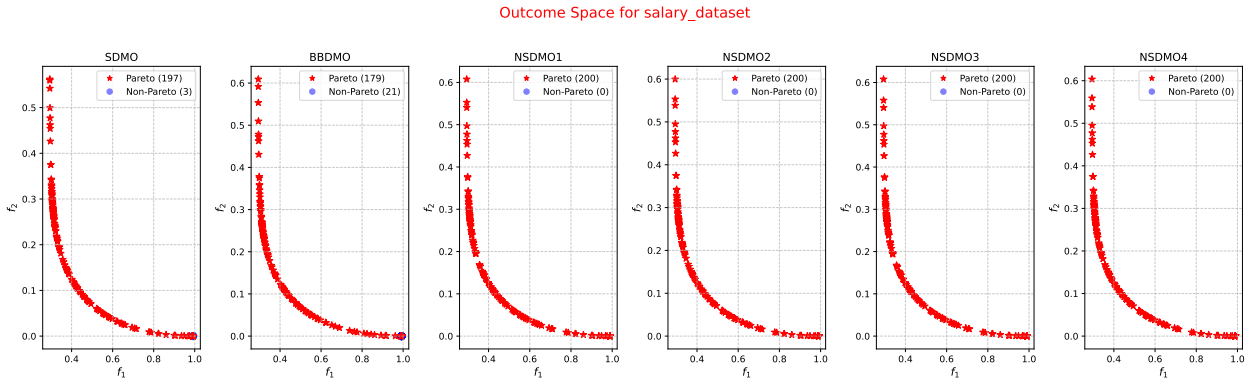

Figure 23: The results of Problem (3.3) on the *salary* dataset

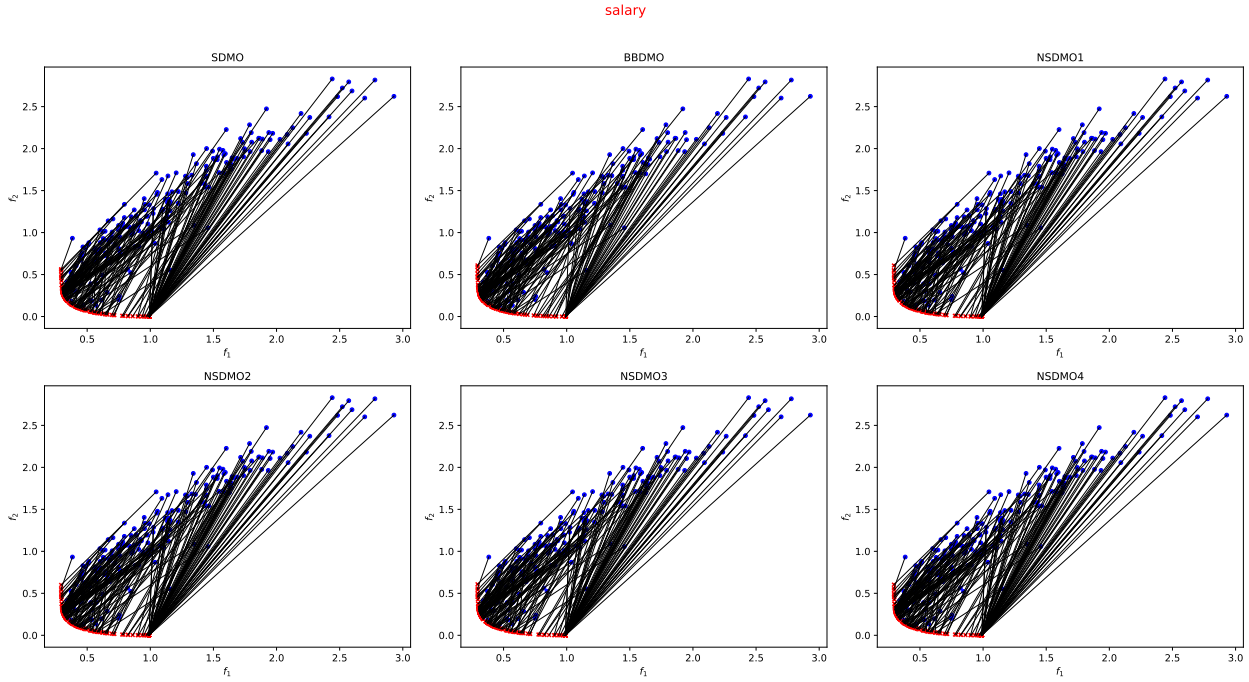

Figure 24: The results of Problem (3.3) on the *salary* dataset

The numerical results for 20 synthetic problems and for Problem (3.3) with 10 real datasets are presented in Table 3 - Table 4 and Table 5, respectively. We highlight the best results for each metric by bold characters.

Problem	Method	Time(ms)	Iter.	Time/Iter.	Stepsize	# P	Purity	HV	IGD^+
SSFYY2	SDMO	14.3230	38.3150	**0.3738**	0.5342	**185**	**1**	121.8783	0.01385
	BBDMO	26.6960	66.3450	0.4024	**0.6172**	163	0.9939	124.3806	0.00926
	NSDMO1	**0.1634**	**2.6600**	0.0614	0.5973	178	0.9944	123.0469	0.00834
	NSDMO2	0.3064	4.7550	0.0644	0.4253	174	0.9943	**125.1113**	**0.00606**
	NSDMO3	0.2042	2.7150	0.0752	0.5942	178	0.9944	123.0469	0.00834
	NSDMO4	0.2961	4.7550	0.0623	0.4253	174	1	**125.1113**	**0.00606**
PNR	SDMO	1.5894	11.1050	0.1431	0.2337	162	0.9012	128.5076	0.00732
	BBDMO	0.3386	**3.0550**	0.1108	**0.7568**	**164**	0.9939	**129.1306**	0.01090
	NSDMO1	**0.1985**	4.0800	**0.0486**	0.4265	163	**1**	126.8782	0.00833
	NSDMO2	0.5096	7.1600	0.0712	0.2146	161	0.9130	127.3366	0.00516
	NSDMO3	0.2639	4.3450	0.0607	0.4211	163	**1**	126.9407	0.00825
	NSDMO4	0.6097	7.9600	0.0766	0.1816	162	0.8210	127.3477	**0.00499**
Imbalance1	SDMO	4.3395	60.8400	0.0713	0.5422	**200**	0.995	79285157.2121	430.99949
	BBDMO	**0.2875**	**2.8450**	0.1011	0.7496	**200**	**1**	79286506.1225	430.97430
	NSDMO1	0.5785	11.1350	0.0520	**2.5744**	**200**	0.985	172104696.0484	10.56311
	NSDMO2	1.3265	27.9750	**0.0474**	1.0477	**200**	0.99	79427600.1050	**2.75517**
	NSDMO3	0.7693	11.6150	0.0662	2.2761	**200**	0.995	172104694.5525	10.56309
	NSDMO4	2.1529	37.3750	0.0576	0.8170	**200**	0.99	**277375316.7022**	66.25837
Imbalance2	SDMO	81.7729	486.9400	0.1679	0.0079	**200**	**1**	175778.7503	0.01136
	BBDMO	**0.3540**	**2.0000**	0.1770	**0.5234**	**200**	**1**	**175804.9177**	**0.00545**
	NSDMO1	0.5618	10.4400	0.0538	0.3339	**200**	**1**	175777.4375	0.02688
	NSDMO2	17.7106	360.3500	0.0491	0.0121	**200**	0.78	175775.5362	0.01532
	NSDMO3	0.6762	10.4400	0.0648	0.3339	**200**	0.995	175777.4375	0.02688
	NSDMO4	17.3648	360.3500	**0.0482**	0.0121	**200**	0.79	175775.5362	0.01532
Hil	SDMO	2.9522	16.1950	0.1823	0.1879	183	**1**	4.7296	0.00144
	BBDMO	1.5737	9.2550	0.1700	**0.7892**	183	0.9891	**4.7310**	0.00188
	NSDMO1	**0.8187**	**6.5000**	0.1259	0.4864	180	**1**	4.7282	0.00140
	NSDMO2	1.6580	13.3750	0.1240	0.2201	184	0.9946	4.7285	0.00116
	NSDMO3	1.6144	14.5500	**0.1110**	0.1896	183	0.9945	4.7285	0.00121
	NSDMO4	1.9110	13.7350	0.1391	0.1878	**184**	0.9783	4.7289	**0.00113**
FF1	SDMO	3.3955	41.2950	0.0822	1.0000	**200**	0.995	3.0079	**0.00046**
	BBDMO	0.4946	**4.9200**	0.1005	0.9544	**200**	**1**	**3.0168**	0.00065
	NSDMO1	0.5003	6.9600	0.0719	**5.6579**	**200**	0.995	3.0050	0.00058
	NSDMO2	1.8151	21.0000	0.0864	2.8404	**200**	0.99	3.0077	0.00048
	NSDMO3	**0.4498**	7.1950	**0.0625**	5.5573	**200**	0.98	3.0050	0.00056
	NSDMO4	1.3576	21.1400	0.0642	2.8625	**200**	0.935	3.0077	0.00048
SP1	SDMO	1.3535	12.9750	0.1043	0.3912	**200**	**1**	23.0455	0.00524
	BBDMO	1.1212	**12.0050**	0.0934	**0.8350**	191	0.9843	23.0365	0.00790
	NSDMO1	**0.8716**	14.0050	0.0622	0.4380	**200**	**1**	23.0521	**0.00459**
	NSDMO2	1.2136	17.9700	0.0675	0.4926	**200**	**1**	**23.0660**	0.00489
	NSDMO3	0.9804	15.4150	0.0636	0.4034	199	**1**	23.0487	0.00503
	NSDMO4	0.9947	19.0200	**0.0523**	0.4714	**200**	0.995	23.0532	0.00499
VU1	SDMO	31.9425	377.4400	0.0846	0.3277	154	0.7273	61.5249	0.00285
	BBDMO	1.2740	15.8300	0.0805	0.9636	**200**	**1**	61.6084	0.00221
	NSDMO1	**0.7864**	**12.9100**	0.0609	**26.0553**	**200**	**1**	**61.6179**	0.00365
	NSDMO2	17.1457	341.3500	**0.0502**	0.4831	172	0.7674	61.5796	**0.00218**
	NSDMO3	1.0330	16.0850	0.0642	21.2982	**200**	**1**	61.6151	0.00342
	NSDMO4	17.7575	341.9650	0.0519	0.4674	171	0.7719	61.5697	0.00225
SD	SDMO	1.8349	21.6650	0.0847	1.0000	**200**	**1**	25.9173	0.00340
	BBDMO	0.9217	**7.3500**	0.1254	0.9989	**200**	**1**	**25.9228**	0.00733
	NSDMO1	**0.6434**	7.4650	0.0862	**2.5579**	**200**	**1**	25.9169	0.00271
	NSDMO2	0.6878	9.6800	**0.0711**	2.3782	**200**	**1**	25.9157	0.00289
	NSDMO3	0.7319	7.5650	0.0968	2.5102	**200**	**1**	25.9158	**0.00270**
	NSDMO4	0.7500	9.8400	0.0762	2.3359	**200**	**1**	25.9156	0.00289
DD1	SDMO	4.9511	59.5350	0.0832	0.5077	138	**0.9855**	3.5979e+10	12896519.99
	BBDMO	0.9231	**8.1950**	0.1126	0.9587	**176**	0.9659	3.6236e+10	12896519.76
	NSDMO1	0.8456	9.6100	0.0880	**3.0168**	137	0.9854	**4.7381e+16**	**5.37330**
	NSDMO2	2.0211	36.4700	**0.0554**	1.1465	138	**0.9855**	6.6822e+10	12896467.71
	NSDMO3	**0.8081**	10.1800	0.0794	2.6885	137	0.9854	3.7883e+10	12896516.07
	NSDMO4	2.2893	36.5900	0.0626	1.1471	138	0.9783	4.3403e+10	12896505.43
JOS1	SDMO	30.5253	268.8400	0.1135	1.0000	54	0.2407	22.0598	0.01888
	BBDMO	**0.2651**	**2.0000**	0.1326	1.0000	90	**1**	22.0600	0.01800
	NSDMO1	0.8750	11.0100	0.0795	16.3976	197	0.6853	22.0621	0.01284
	NSDMO2	1.2922	14.0000	0.0923	**32.6662**	**200**	0.74	**22.2298**	**0.00004**
	NSDMO3	1.0725	11.0100	0.0974	16.3976	197	0.6396	22.0621	0.01284
	NSDMO4	1.0197	14.0000	**0.0728**	**32.6662**	**200**	0.745	**22.2298**	**0.00004**

Table 3: Numerical results for bi-objective test problems

Problem	Method	Time(ms)	Iter.	Time/Iter.	Stepsize	#P	Purity	HV	IGD$^+$
MHHM1	SDMO	0.5125	**0.9100**	0.5632	**0.4550**	**200**	**1**	**1.0226**	**0**
	BBDMO	0.5488	**0.9100**	0.6031	**0.4550**	**200**	**1**	**1.0226**	**0**
	NSDMO1	0.4683	**0.9100**	0.5146	**0.4550**	**200**	**1**	**1.0226**	**0**
	NSDMO2	**0.2570**	**0.9100**	**0.2824**	**0.4550**	**200**	**1**	**1.0226**	**0**
	NSDMO3	0.4091	**0.9100**	0.4495	**0.4550**	**200**	**1**	**1.0226**	**0**
	NSDMO4	0.2649	**0.9100**	0.2911	**0.4550**	**200**	**1**	**1.0226**	**0**
IKK1	SDMO	21.4880	24.9050	**0.8628**	0.7400	136	**1**	**15597189089**	0.0068
	BBDMO	**2.4921**	**1.8250**	1.3655	0.7383	150	0.9667	15597189012	0.0082
	NSDMO1	4.1919	2.8400	1.4760	0.6865	154	0.9610	15597187835	**0.0024**
	NSDMO2	3.1596	2.5650	1.2318	**0.6824**	142	0.9437	15597187008	0.0087
	NSDMO3	3.6925	2.6250	1.4067	0.6668	**156**	0.9551	15597187355	0.0042
	NSDMO4	3.8330	2.5500	1.5031	0.6655	142	0.9718	15597186982	0.0092
AP1	SDMO	155.1233	170.1950	**0.9114**	0.4884	169	0.9112	48992918.6687	0.0102
	BBDMO	18.8440	10.0750	1.8704	0.8787	196	**0.9949**	**49004401.0559**	0.0132
	NSDMO1	12.8793	**9.6550**	1.3340	**26.3011**	**200**	0.94	49002893.5806	**0.0053**
	NSDMO2	152.3974	166.0400	0.9178	0.3162	158	0.9114	48988441.6060	0.0090
	NSDMO3	**12.6794**	9.6650	1.3119	26.2981	**200**	0.94	49002893.5707	**0.0053**
	NSDMO4	163.6059	171.1000	0.9562	0.2717	157	0.9172	48988222.5807	0.0091
AP4	SDMO	175.3234	153.9050	1.1392	0.3682	175	0.9086	26509595.1550	**0.0078**
	BBDMO	108.6529	28.0700	3.8708	0.8299	199	**0.9899**	**26524695.0847**	0.0128
	NSDMO1	**10.9337**	**8.7800**	1.2453	**22.0900**	**200**	0.97	26524041.8087	0.0102
	NSDMO2	146.7892	144.6500	**1.0148**	0.2988	155	0.9355	26502748.7700	0.0126
	NSDMO3	11.1058	8.7850	1.2642	22.0570	**200**	0.97	26524041.9549	0.0102
	NSDMO4	154.5409	148.7050	1.0392	0.2490	151	0.9404	26501384.5293	0.0125
MGH26a	SDMO	13.9338	12.2950	1.1333	0.7306	**200**	**1**	**1.9786**	0.0028
	BBDMO	**6.2793**	**5.7550**	**1.0911**	**0.9382**	195	0.9795	1.9784	0.0037
	NSDMO1	9.1470	6.6450	1.3765	0.7247	**200**	**1**	1.9784	0.0019
	NSDMO2	10.2371	8.2750	1.2371	0.6505	199	0.9849	1.9785	0.0019
	NSDMO3	9.6028	7.5300	1.2753	0.6580	**200**	**1**	1.9783	0.0020
	NSDMO4	11.2974	9.7600	1.1575	0.6172	196	0.9745	**1.9786**	0.0022
FDS	SDMO	436.8657	279.0750	1.5654	0.2866	**200**	**1**	43135.3324	0.0765
	BBDMO	9.8393	**7.1900**	1.3685	0.9703	**200**	**1**	**44261.8620**	**0.0535**
	NSDMO1	**7.5050**	7.9450	**0.9446**	5.2867	**200**	**1**	40980.6205	0.1773
	NSDMO2	409.4826	406.8100	1.0066	0.1134	194	0.9588	43156.7171	0.0656
	NSDMO3	7.6427	7.9750	0.9583	**5.3217**	**200**	**1**	40981.0681	0.1773
	NSDMO4	501.7102	477.1950	1.0514	0.0603	184	0.8696	43158.2754	0.0681
TRIDIA2	SDMO	175.2704	57.1450	3.0671	0.2449	192	0.9740	795829.7027	0.0382
	BBDMO	194.2212	48.1600	4.0328	**0.6749**	183	0.9344	**842850.7601**	**0.0353**
	NSDMO1	**25.4157**	**14.4800**	1.7552	0.5601	**197**	**0.9797**	799868.7744	0.0364
	NSDMO2	46.6092	28.6200	**1.6286**	0.1904	181	0.9337	799069.1049	0.0357
	NSDMO3	32.7522	18.7350	1.7482	0.2941	191	0.9529	800520.7735	0.0371
	NSDMO4	74.9278	38.6500	1.9386	0.1259	172	0.9419	803704.8667	0.0381
MGH26b	SDMO	532.7254	104.7950	5.0835	0.5935	**200**	0.995	2.3567	0.0070
	BBDMO	146.5799	28.3900	5.1631	**0.8878**	**200**	**1**	**2.3595**	0.0080
	NSDMO1	**23.0453**	**7.4950**	3.0748	0.7575	**200**	**1**	2.3537	0.0044
	NSDMO2	25.2425	9.4000	**2.6854**	0.6411	**200**	**1**	2.3560	0.0040
	NSDMO3	23.8938	7.8700	3.0361	0.6713	**200**	0.995	2.3546	**0.0039**
	NSDMO4	31.0905	11.1300	2.7934	0.5824	**200**	0.985	2.3566	0.0043
MGH26c	SDMO	529.1399	99.2650	5.3306	0.5477	**200**	**1**	2.7556	0.0088
	BBDMO	267.4051	48.3900	5.5260	**0.8436**	**200**	**1**	**2.7589**	0.0120
	NSDMO1	**27.0096**	**7.5300**	3.5869	0.7626	**200**	**1**	2.7467	0.0061
	NSDMO2	38.0670	10.6900	3.5610	0.6144	**200**	0.995	2.7480	**0.0053**
	NSDMO3	34.8997	8.8650	3.9368	0.6686	**200**	**1**	2.7477	0.0055
	NSDMO4	41.4795	11.8700	**3.4945**	0.5319	**200**	0.99	2.7489	0.0057

Table 4: Numerical results for three-, four-, and five-objective test problems

Dataset	Method	Time	Iter.	Time/Iter.	Stepsize	#P	Purity	HV	IGD$^+$
car	SDMO	1.0614	10.0650	0.1055	0.5139	**200**	**1**	3.3411	0.00070961
	BBDMO	0.7456	**6.9350**	0.1075	**0.9087**	**200**	**1**	**3.3505**	0.00092864
	NSDMO1	0.4795	7.4350	0.0645	0.5632	**200**	0.995	3.3412	0.00067825
	NSDMO2	0.5779	9.3100	**0.0621**	0.5667	**200**	0.99	3.3415	**0.00059998**
	NSDMO3	**0.4793**	7.4300	0.0645	0.5507	**200**	0.995	3.3418	0.00064496
	NSDMO4	0.5215	9.3750	0.0556	0.5596	**200**	0.995	3.3420	0.00067148
car_v3	SDMO	15.8289	16.6950	0.9481	0.4014	**200**	0.985	2.2937	0.00066808
	BBDMO	11.4131	11.9100	0.9583	**0.8338**	198	**1**	2.2940	0.00061275
	NSDMO1	**5.5897**	**9.4350**	**0.5924**	0.5150	**200**	0.995	2.2938	0.00051974
	NSDMO2	6.6568	10.8050	0.6161	0.4551	**200**	0.975	2.2935	**0.00036705**
	NSDMO3	6.2472	10.0550	0.6213	0.5053	**200**	0.99	**2.2941**	0.00045018
	NSDMO4	6.6184	10.9700	0.6033	0.4279	**200**	0.985	2.2937	0.0004095
insur	SDMO	2.0694	6.6300	0.3121	0.5247	**200**	**1**	2.7641	0.00029938
	BBDMO	1.4911	**5.7200**	0.2607	**0.9206**	197	0.9695	2.7643	0.00035126
	NSDMO1	**0.8408**	6.6700	0.1260	0.6040	**200**	0.995	**2.7678**	0.00017852
	NSDMO2	0.8647	6.9150	**0.1251**	0.6326	196	0.9388	**2.7678**	**0.00013028**
	NSDMO3	1.1573	6.6300	0.1746	0.5984	**200**	0.985	**2.7678**	0.00017596
	NSDMO4	1.1366	6.8950	0.1648	0.6281	198	0.8333	**2.7678**	0.00014371
house	SDMO	0.6786	2.1000	0.3231	0.4944	198	**1**	1.3244	0.00013348
	BBDMO	0.5721	2.0450	0.2797	**0.7512**	**200**	**1**	**1.4728**	0.00066781
	NSDMO1	0.3441	**2.0350**	0.1691	0.4967	198	**1**	1.3245	0.00013286
	NSDMO2	0.3918	2.0600	0.1902	0.3319	198	**1**	1.3250	0.00013174
	NSDMO3	0.4996	**2.0350**	0.2455	0.4970	198	**1**	1.3246	0.00013257
	NSDMO4	**0.2687**	2.0650	**0.1301**	0.3297	198	**1**	1.3252	**0.0001314**
campaign	SDMO	3.7319	5.7000	0.6547	0.4644	**200**	**1**	**2.3036**	0.00027683
	BBDMO	3.2605	5.1150	0.6374	**0.7904**	196	**1**	**2.3036**	0.00047776
	NSDMO1	2.1564	**4.4600**	0.4835	0.5245	**200**	0.94	2.3035	0.00024256
	NSDMO2	2.3420	4.8100	0.4869	0.4393	**200**	0.925	2.3035	**0.0002298**
	NSDMO3	**2.0748**	4.6000	**0.4511**	0.5223	**200**	0.94	2.3035	0.0002618
	NSDMO4	2.7712	5.1050	0.5428	0.4297	**200**	0.925	**2.3036**	0.00027876
price	SDMO	3.2534	15.9650	0.2038	0.5055	**200**	**1**	1.9701	0.00047506
	BBDMO	2.0123	9.8850	0.2036	**0.8482**	**200**	**1**	1.9657	0.00110196
	NSDMO1	**0.9624**	**9.6100**	0.1001	0.6719	**200**	0.97	1.9705	0.00034147
	NSDMO2	1.0372	11.1550	**0.0930**	0.7347	**200**	0.945	1.9712	**0.00033496**
	NSDMO3	1.5054	10.4000	0.1447	0.6467	**200**	0.97	**1.9724**	0.00036167
	NSDMO4	1.3140	11.5150	0.1141	0.7059	**200**	0.93	1.9712	0.00041722
quality	SDMO	2.3017	3.0450	0.7559	0.4864	**200**	**1**	1.4883	**0.00008141**
	BBDMO	2.3090	2.8400	0.8130	**0.7668**	**200**	**1**	**1.5067**	0.00008278
	NSDMO1	**1.8416**	**2.5950**	0.7097	0.5052	**200**	**1**	1.3398	0.00140311
	NSDMO2	2.1651	3.1850	0.6798	0.4076	**200**	**1**	1.3402	0.001389
	NSDMO3	1.9207	2.6100	0.7359	0.5023	199	**1**	1.3398	0.00140109
	NSDMO4	2.1067	3.2950	**0.6394**	0.3744	**200**	**1**	1.3403	0.0013872
ames_house	SDMO	2.4031	**2.0000**	1.2015	0.5000	199	**1**	**1.0101**	**0.00000097**
	BBDMO	2.5271	**2.0000**	1.2635	**0.7500**	**200**	**1**	1.0025	0.0061623
	NSDMO1	2.1889	**2.0000**	1.0944	0.4933	199	**1**	**1.0101**	**0.00000097**
	NSDMO2	2.2363	**2.0000**	1.1181	0.2801	199	**1**	**1.0101**	**0.00000097**
	NSDMO3	2.2422	**2.0000**	1.1211	0.4933	199	**1**	**1.0101**	**0.00000097**
	NSDMO4	**1.9237**	**2.0000**	**0.9618**	0.2799	199	**1**	**1.0101**	**0.00000097**
salary	SDMO	2.3879	9.0050	0.2652	0.4749	197	**0.9949**	2.6913	0.00029203
	BBDMO	1.6463	**5.5700**	0.2956	**0.7709**	179	0.9497	**2.6916**	0.00045966
	NSDMO1	**0.8757**	5.8850	**0.1488**	0.7165	**200**	0.965	2.6914	0.00032521
	NSDMO2	1.1060	6.8950	0.1604	0.6167	**200**	0.915	2.6914	**0.00026195**
	NSDMO3	0.9189	6.1100	0.1504	0.6707	**200**	0.96	2.6912	0.00034443
	NSDMO4	1.1271	6.9500	0.1622	0.6129	**200**	0.915	2.6914	0.00026254
sales	SDMO	**3293.9095**	**10.7400**	306.6955	0.4999	**200**	**1**	3.9433	0.00767489
	BBDMO	8407.8642	22.6850	370.6354	**0.8233**	191	0.9948	4.0667	0.00149706
	NSDMO1	4016.1955	14.7800	271.7318	0.5751	**200**	0.95	4.0660	0.00093865
	NSDMO2	3915.2088	14.4250	271.4183	0.6089	**200**	0.95	4.0670	**0.00081776**
	NSDMO3	4547.4552	16.9950	**267.5761**	0.5708	199	0.9548	4.0667	0.00087973
	NSDMO4	4114.3336	15.2050	270.5908	0.5785	**200**	0.965	**4.0668**	0.00102671

Table 5: Numerical results for Problem (3.3) with 10 real datasets