J. Appl. Numer. Optim. 7 (2025), No. 3, pp. 421-458
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.7.2025.3.08

A LINE DECOMPOSITION ALGORITHM FOR MULTIOBJECTIVE OPTIMIZATION

EMMA SORIANO*, MISHKO MITKOVSKI, MARGARET M. WIECEK

School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC, USA

Abstract. Decomposition techniques are proven highly effective in addressing complexity of optimization problems. For multiobjective optimization problems (MOPs), a variety of objective-space decomposition approaches are developed and applied in practice, while decision-space decomposition remains rather underexplored. We develop a line-decomposition algorithm for computing an approximation of the efficient set of strictly convex MOPs. The feasible region is decomposed into lines whose efficient sets are used to reconstruct the overall efficient set. Because the algorithm relies on solving a collection of single objective line search problems, it is immediately applicable to single-objective optimization with no modifications. We prove the algorithm convergence and provide a preliminary error analysis. The algorithm is implemented in Python and tested on biobjective and single objective problems with bounded variables. Numerical results are also included.

Keywords. Decision-space decomposition; Efficient set; Multiobjective line search.
2020 Mathematics Subject Classification. 90C29, 90C25.

1. INTRODUCTION

Real-world decision problems are often large-scale due to the number of decision variables or the complexity of objective and constraint functions. This can make the resulting optimization models numerically challenging or even infeasible to solve as a single problem. Decomposition techniques have proven highly effective in addressing this complexity across various optimization problems [1]. This technique involves dividing the original problem into a collection of subproblems, solving each subproblem individually, and then combining their solutions to construct the solution to the original problem. For example, a subproblem might focus on optimizing only a subset of the objectives or apply a relaxation or restriction to the decision space. In any case, solving these subproblems shall be computationally much easier than tackling the original problem directly.

Multiobjective optimization problems (MOPs) can be decomposed based on various modeling aspects, including the scientific or engineering disciplines used to create the mathematical model, the objective functions, the scenarios in which the system is expected to operate, the

*Corresponding author.
E-mail address: esorian@clemson.edu (E. Soriano), wmalgor@clemson.edu (M. Wiecek), mmitkov@clemson.edu (M. Mitkovski).

Received 15 April 2025; Accepted 19 September 2025; Published online 24 November 2025.

involvement of decision-makers in the process, and more. As discussed in [2], current state-of-the-art decomposition approaches to MOPs include objective-space decomposition algorithms which have been developed and applied in practice. In contrast, decision-space decomposition remains relatively underexplored. Theoretical findings for decision-space decomposition are provided in [3, 4, 5], while [6] presents applications in location science and public transportation, which benefit from this type of decomposition, and provide a preliminary study on the block coordinate descent [7] for bi-objective problems.

The goal of this paper is to develop a decision-space decomposition algorithm to compute an approximation of the efficient set for strictly convex MOPs. The algorithm is based on an earlier theoretical study on decomposition for the efficient set and accompanying set-convergence in the Painlevé-Kuratowski sense [8]. That study had originally been inspired by the block-coordinate descent for single-objective problems and a decomposition theorem in [5]. The proposed algorithm is referred to as a line-decomposition algorithm since the feasible region is decomposed into lines whose efficient sets are used to reconstruct the overall efficient set. We prove the convergence of this algorithm and provide a preliminary error analysis for an implementation in $\mathbb{R}^n$. We implement this algorithm in Python and test for biobjective and single-objective problems with bounded variables. Due to the decomposition into lines, the algorithm conceptually relies on solving a multiobjective line search problem, which is executed as solving a collection of single-objective line-search problems. This execution reduces the complexity of computing the efficient set as well as makes the algorithm immediately applicable to single-objective optimization with no modifications.

The structure of this paper is as follows. In Section 2, the necessary background and the line decomposition theorem are presented. The approach to implementing the decomposition algorithm is outlined in Section 3, while the error accumulation and convergence of the algorithm are analyzed in Section 4. In Section 5, we provide computational results for the algorithm applied to both bi-objective and single-objective problems, and present the implementation details of the algorithm. The paper is concluded in Section 6.

2. Preliminaries

We first review foundational concepts for MOPs and basic results, then introduce a notion of set convergence, and lastly present the line decomposition theory which is implemented in later sections.

2.1. **Multiobjective optimization.** A multiobjective optimization problem can be formulated as below.

$$\begin{aligned} &\min f(x) = [f_1(x)\ ,\ f_2(x)\ ,\ \dots\ ,\ f_p(x)], \\ &\text{s.t. } x \in X \end{aligned} \tag{2.1}$$

where $X \subseteq \mathbb{R}^n$, $f : X \to \mathbb{R}^p$, and $p \geq 2$. We refer to the set X as the *feasible set*, and the set $Y := f(X) = \{y \in \mathbb{R}^p : y = f(x), x \in X\}$ as the *image set*.

A partial order must be prescribed to define optimal solutions to (2.1) that are referred to as efficient solutions. Let $u, v \in \mathbb{R}^p$. We define the binary relations which are partial orders: $u < v$ if $u_i < v_i$ for all $i \in [p]$; $u \leq v$ if $u_i \leq v_i$, for all $i \in [p]$ with at least one strict inequality; and $u \leqq v$ if $u_i \leq v_i$, for all $i \in [p]$. The most commonly used partial order is the relation $\leq$ which defines efficient solutions.

A point $x \in X$ is said to be an *efficient solution* to (2.1) if there is no $\hat{x} \in X$ such that $f(\hat{x}) \leq f(x)$. The set of all efficient solutions in X is denoted by $\mathscr{E}(X)$ and is called the *efficient set of* X. When $p = 1$, problem (2.1) reduces to a single-objective problem, and the definition of an efficient point reduces to the definition of a minimizer in single-objective optimization.

The image of an efficient solution is called a *Pareto point* for (2.1). The set of all Pareto points is denoted by $\mathscr{P}(Y) := \{y \in \mathbb{R}^p : y = f(x) \text{ for } x \in \mathscr{E}(X)\}$ and is called the *Pareto set of* Y.

The computation of the efficient set on a line is performed using Theorem 2.1 that allows to obtain this set in a closed form by solving multiple single-objective problems.

Theorem 2.1. [9] *Let $f : \mathbb{R}^n \to \mathbb{R}^p$ be strictly convex, and let the feasible set be a line segment, $L = \{\bar{x} + \alpha d \mid l \leq \alpha \leq u\}$, where $\bar{x}, d \in \mathbb{R}^n, l, u \in \mathbb{R}$. For each $r \in [p]$, define $\alpha_r^* := \arg\min_{\alpha \in [l,u]} f_r(\alpha d + \bar{x})$ to be the unique minimizer of f_r. Then, the efficient set of L is exactly*

$$\mathscr{E}(L) = \left\{\bar{x} + \alpha d \mid \min_{r \in [p]}\{\alpha_r^*\} \leq \alpha \leq \max_{r \in [p]}\{\alpha_r^*\}\right\}.$$

Since the line decomposition algorithm proposed in the next section produces approximations of the efficient set, a measure of approximation quality is needed. To compare approximate solutions to the true efficient set, we introduce the notion of Hausdorff distance of sets and Painlevé-Kuratowski convergence of sets below.

2.2. **Set convergence.** Throughout this paper we use $||\cdot||$ to denote the ℓ^2-norm and $||\cdot||_{\text{op}}$ to denote the operator norm. Let $\mathscr{X} \subseteq \mathbb{R}^n$. Then, for $A, B \subseteq \mathscr{X}$ the *Hausdorff distance* from A to B is $\Delta_H(A,B) := \max\left(\sup_{b \in B} \text{dist}(b,A)\ ,\ \sup_{a \in A} \text{dist}(a,B)\right)$, where $\text{dist}(x,A) := \inf_{a \in A} ||x-a||$. Let $\{A_N\}_{N=1}^{\infty} \subseteq \mathscr{X}$ be a sequence of subsets of $\mathscr{X}$. The set A is called the *Painlevé-Kuratowski limit of* $\{A_N\}$ if $\underset{N\to\infty}{\text{Liminf}}\ A_N = \underset{N\to\infty}{\text{Limsup}}\ A_N = A$ where

$$\underset{N\to\infty}{\text{Liminf}}\ A_N := \{x \in \mathscr{X} : \forall \varepsilon > 0,\ B(x,\varepsilon) \cap A_N \neq \emptyset \text{ for all but finitely many } N\},$$
$$\underset{N\to\infty}{\text{Limsup}}\ A_N := \{x \in \mathscr{X} : \forall \varepsilon > 0,\ B(x,\varepsilon) \cap A_N \neq \emptyset \text{ for infinitely many } N\}.$$

We denote this type of convergence by $A_N \xrightarrow{K} A$. If $\mathscr{X}$ is a compact space, the Painlevé-Kuratowski convergence is equivalent to the Hausdorff convergence, i.e., $\Delta_H(A_N, A) \to 0 \iff A_N \xrightarrow{K} A$.

2.3. **Line decomposition.** We now introduce the notation used to decompose the decision space using lines.

Definition 2.1. Let $\mathbb{S}^{n-1}$ denote the unit sphere in $\mathbb{R}^n$. Then, for each $d \in \mathbb{S}^{n-1}$ define the set $P(d) := \text{proj}_d(X)$, where $\text{proj}_d(X)$ denotes the orthogonal projection of X onto the hyperplane $\{x \in \mathbb{R}^n : d^T x = 0\}$.

Below is a useful property of the projection operator.

Proposition 2.1. *Let $v \in \mathbb{R}^n$. Then, $||\text{proj}_d(v)|| \leq ||v||$.*

The proof follows immediately from the fact that the operator norm of the projection operator is $||\text{proj}_d||_{\text{op}} := \sup_{||v||=1} ||\text{proj}_d(v)|| = 1$.

Definition 2.2. For a fixed $d \in \mathbb{S}^{n-1}$ and $x \in P(d)$, define the feasible line segment $X(d,x)$ by

$$X(d,x) := \{x + \alpha d : \alpha \in \mathbb{R}\} \cap X.$$

Note that if X is convex, then every set $X(d,x)$ may be written as a single line segment of the form $\{x+\alpha d : \alpha \in \mathbb{R} \text{ s.t. } \alpha_l \leq \alpha \leq \alpha_u\}$ for some $\alpha_l, \alpha_u \in \mathbb{R} \cup \{\pm\infty\}$. To simplify the notation used for line search subproblems, we also define the variables that describe the feasible step sizes range for a line segment $X(d,x)$.

Definition 2.3. For a fixed direction $d \in \mathbb{S}^{n-1}$ and $x \in P(d)$ we define the feasible step size bounds for a line segment $X(d,x)$ by

$$\alpha_l^x := \min\{\alpha \in \mathbb{R} : l \leq x_i + \alpha d_i \leq u, \forall d_i \neq 0\},$$
$$\alpha_u^x := \max\{\alpha \in \mathbb{R} : l \leq x_i + \alpha d_i \leq u, \forall d_i \neq 0\}.$$

The line decomposition theorem which gives a foundation for the proposed algorithm is stated below in Theorem 2.2. This decomposition method involves decomposing X into a collection of line segments, denoted by $X(d,x)$, and then finding the efficient set on each of these lines. The efficient line segments are denoted by $\mathscr{E}(X(d,x))$. Theorem 2.2 specifies the exact collection of efficient line segments needed to recover the overall efficient set to the MOP, $\mathscr{E}(X)$.

Theorem 2.2 ([8])**.** *Let $X \subseteq \mathbb{R}^n$ be compact and $f : X \to \mathbb{R}^p$ be a continuous function, $p \geq 1$. Then $\mathscr{E}(X) = \bigcap_{d \in \mathbb{S}^{n-1}} \bigcup_{x \in P(d)} \mathscr{E}(X(d,x))$.*

Remark 2.1. Only direction vectors in the upper half of $\mathbb{S}^{n-1}$ are needed since vectors in the lower half of the sphere will define the same lines.

Remark 2.2. For $p = 1$, Theorem 2.1 can still be used to solve each line-search subproblem. Thus, the proposed decomposition can be applied to multi- or single-objective optimization interchangeably in contrast to other optimization methods that are suitable only to either scalar or vector optimization.

Intuitively, the set $\bigcup_{x \in P(d)} \mathscr{E}(X(d,x))$ gives the points which are efficient in the direction d. Then, taking an intersection over all $d \in \mathbb{S}^{n-1}$ will give points which are efficient in all directions (i.e., efficient points to the original MOP).

A notable trait of this decomposition theorem is that no additional optimization is needed after decomposing the MOP. The efficient set can be recovered by only taking a union and intersection of subproblem solutions. Note that the sets $\mathbb{S}^{n-1}$ and $P(d)$ which define the intersection and union in Theorem 2.2 are continuous, and therefore infinitely many subproblems must be solved to recover the efficient set. However, in practical implementation only a finite number of subproblems can be solved. Theorem 2.3 below establishes convergence of the line decomposition method as discrete indexing sets converge to the continuous sets $\mathbb{S}^{n-1}$ and $P(d)$.

Theorem 2.3 ([8])**.** *Let $X \subseteq \mathbb{R}^n$ be a polytope with non-empty interior, and $f : X \to \mathbb{R}^p$ be a continuous and strictly convex function. Let $\{S_M\}_{M=1}^{\infty} \subseteq \mathbb{S}^{n-1}$ be a monotonic non-decreasing sequence of finite subsets $S_M \subseteq \mathbb{S}^{n-1}$ such that $S_M \xrightarrow{K} \mathbb{S}^{n-1}$ as $M \to \infty$. For each $d \in \mathbb{S}^{n-1}$ let $\{P_N^{(d)}\}_{N=1}^{\infty} \subseteq P(d)$ be a monotonic non-decreasing sequence of finite subsets $P_N^{(d)} \subseteq P(d)$ such that $P_N^{(d)} \xrightarrow{K} P(d)$ as $N \to \infty$. Then $\lim_{M\to\infty} \bigcap_{d \in S_M} \lim_{N\to\infty} \bigcup_{x \in P_N^{(d)}} \mathscr{E}(X(d,x)) = \mathscr{E}(X)$.*

2.4. **Error analysis.** In our error analysis presented in Section 4, we use a theorem by Subotić, Hauswirth, and Dörfler [10] to help describe the distance between parallel efficient line segments. In Theorem 2.4 we state their result in a form which is relevant to our work.

Theorem 2.4 (Subotic et al., [10] Theorem 3)**.** *Let $\Xi \subseteq \mathbb{R}^t$ and $\phi : \mathbb{R}^n \to \mathbb{R}$ be strictly convex and twice continuously differentiable. Let $g : \mathbb{R}^n \times \Xi \to \mathbb{R}^m$ and $h : \mathbb{R}^n \times \Xi \to \mathbb{R}^\ell$ be linear functions. Consider the parametric nonlinear optimization problem below,*

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \ \phi(x,\xi), \\ &s.t. \ \ h(x,\xi) = 0, \\ &\qquad g(x,\xi) \leq 0 \end{aligned} \tag{2.2}$$

where $\xi \in \Xi$. Let x^* be a minimizer to (2.2) for $\bar{\xi}$ which satisfies the linear independence constraint qualification condition. Then on a neighborhood $\mathcal{N} \subseteq \Xi$ of $\bar{\xi}$ there exists a Lipschitz continuous map $\hat{x} : \mathcal{N} \to \mathbb{R}^n$ such that

(1) $\hat{x}(\bar{\xi}) = x^*$,
(2) for all $\xi \in \mathcal{N}$, $\hat{x}(\xi)$ is a local minimizer for (2.2).

Having established the necessary theoretical and methodological concepts, we apply them to develop the line decomposition algorithm in the next section.

3. Line Decomposition Algorithm

In this section we present the line decomposition algorithm which is an application of Theorem 2.2. For our implementation of the algorithm, we assume the set $X \subseteq \mathbb{R}^n$ is a hypercube defined by $X = [l,u]^n$, and $f : X \to \mathbb{R}^p$ is continuous and strictly convex. In order to apply Theorem 2.2 in practice, we must convert $\mathbb{S}^{n-1}$ and $P(d)$ into finite discrete sets that will serve as indexing sets in implementation. Intuitively, the set of vectors used to represent $\mathbb{S}^{n-1}$ are the directions in which we check for efficiency in the decision space.

Before presenting a pseudocode, we provide an overview of the process for the line decomposition algorithm by describing its key steps and introducing notation for the discrete variables used. The first two steps are to represent $\mathbb{S}^{n-1}$ and X as discrete sets. Let $S \subseteq \mathbb{S}^{n-1}$ such that $|S| = k < \infty$ denote a discrete set of unit vectors. To represent the feasible hypercube $X = [l,u]^n$, we first represent the interval $[l,u]$ using m equally spaced points. Denote this discrete set by $D \subseteq [l,u]$. Then a discrete representation of X is defined by $X^m := D \times \cdots \times D \subseteq [l,u]^n$. Once the discrete sets S and X^m are defined, we may begin the first iteration of the algorithm. The outermost for-loop in the pseudocode which iterates through each unit vector $d^i \in S$, outlines the steps for each iteration of the algorithm. The first step inside this loop is to compute a discrete representation of $P(d^i)$. Then the set $P(d^i)$ is approximated by projecting each point $x \in X^m$ onto the hyperplane $\{x \in \mathbb{R}^n : (d^i)^T x = 0\}$ using the formula, $\text{proj}_{d^i}(x) = x - (d^i)^T x d^i$. This representation of $P(d^i)$ is saved into a set P, and then we enter the second for-loop in the pseudocode. This loop iterates over each point in $x \in P$ and each time the efficient set $\mathscr{E}(X(d^i,x))$ is computed as follows. First, the range of feasible step sizes for the line search optimization are determined, α_l^x, α_u^x. Then we enter the third for-loop in the pseudocode and solve a line search problem for each objective. For fixed $d^i \in S$ and $\bar{x} \in X$, a single-objective line search problem

is solved for each $r \in [p]$,

$$\min_{\alpha \in [l,u]} f_r(\bar{x} + \alpha d^i) \tag{3.1}$$

whose optimal solution, α_r^*, is then used to determine the efficient set, $\mathscr{E}(X(d^i,x))$, on the line segment $X(d^i,x)$ by applying Theorem 2.1, $\mathscr{E}(X(d^i,x)) = \{\bar{x} + \alpha d^i \mid \min_{r\in[p]}\{\alpha_r^*\} \leq \alpha \leq \max_{r\in[p]}\{\alpha_r^*\}\}$.

Algorithm: Line Decomposition Algorithm

Input:
Objective function: $f : \mathbb{R}^n \to \mathbb{R}^p$;
Variable bounds: $[l,u] \subseteq \mathbb{R}$;
Number of points to represent the interval $[l,u]$: $m \in \mathbb{N}$;
Number of direction vectors to be used: $k \in \mathbb{N}$;
Output:
Approximate efficient set: E;

Discretize unit sphere: $S = \{d^1, \ldots, d^k\}$;
Discretize feasible set: $X^m = \{x^1, x^2, \ldots, x^{m^n}\}$;

Initialize:
$U_i = \emptyset$ for each $i \in [k]$;
$E = \emptyset$;

for $1 \leq i \leq k$ **do**
  Set $P = \text{proj}_{d^i}(X^m)$;
  for $x \in P$ **do**
    Calculate α_l^x and α_u^x ;
    for $1 \leq r \leq p$ **do**
      Solve $\alpha_r^* := \arg\min_{\alpha\in\mathbb{R}}\{f_r(x + \alpha d^i) : \alpha_l^x \leq \alpha \leq \alpha_u^x\}$;
    end
    Compute $\alpha_{\min} := \min\{\alpha_r^* : r \in [p]\}$ and $\alpha_{\max} := \max\{\alpha_r^* : r \in [p]\}$;
    Set $\mathscr{E}(X(d^i,x)) = \{x + \alpha d^i : \alpha_{\min} \leq \alpha \leq \alpha_{\max}\}$;
    Set $U_i = U_i \cup \mathscr{E}(X(d^i,x))$;
  end
  if $i = 1$ **then**
    $E = U_i$;
  else
    $E = E \cap U_i$;
  end
end

Each solution set $\mathscr{E}(X(d^i,x))$ is saved in a list, U_i. Once all subproblem solution sets are saved into U_i for a fixed direction $d^i \in S$, the approximation of the overall efficient set is updated. Let

E denote the approximation of $\mathscr{E}(X)$ generated by the algorithm. If $i = 1$, then there is only one collection of subproblem solutions, U_1, and therefore the initial approximation of $\mathscr{E}(X)$ is set to be $E = U_1$. If $i \geq 2$, then E is updated by taking the intersection of itself with the new collection of subproblem solutions U_i. Once this update has been performed for each direction $d^i \in S$, the algorithm is complete.

In the next section, we analyze the error accumulated in the steps of the algorithm. We approach error from two perspectives, theoretic and experimental. In Section 4, we prove theoretic error bounds and in Section 5.4 we describe measures taken to reduce experimental error. Specifically, experimental error is reduced by implementing a modified intersection during the update of E.

Observe that updating E at the end of an iteration requires taking an intersection with the set U_i. The set U_i is a discrete representation of the continuous set $\bigcup_{x\in P(d^i)} \mathscr{E}(X(d,x))$. In theory, E should be updated by taking the intersection $E \cap \bigcup_{x\in P(d^i)} \mathscr{E}(X(d,x))$. In our implementation, we apply a modified intersection method that reduces experimental error in the update of E. Without this modified intersection method, the updated set E could quickly become sparse or empty since both U_i and E are discrete collections of line segments. Details of this modified intersection method are given in Section 5.4.

4. Error Analysis

To evaluate the accuracy of the approximate efficient set generated by the line decomposition algorithm, we conduct an error analysis by examining the accumulation of error at each iteration. An iteration of the algorithm consists of computing the set $\bigcup_{x\in P} \mathscr{E}(X(d,x))$ for a fixed direction d and updating the approximate efficient set E. It follows from Theorem 2.2 that using finitely many directions $d \in \mathbb{S}^{n-1}$ for line decomposition will produce an approximation of the efficient set:

$$\bigcap_{i=1}^{k} \bigcup_{x\in P(d^i)} \mathscr{E}(X(d^i,x)) \approx \bigcap_{d\in\mathbb{S}^{n-1}} \bigcup_{x\in P(d^i)} \mathscr{E}(X(d^i,x)).$$

In the implementation of the line decomposition algorithm, we take a finite union using indexing sets $P_m^{d^i} = \text{proj}_{d^i}(X^m)$ rather than the continuous set $P(d^i)$. Using Theorem 2.3, it has been proven that the sets $\bigcap_{i=1}^{k} \bigcup_{x\in P_m^{d^i}} \mathscr{E}(X(d^i,x))$ converge to this approximate efficient set as $m \to \infty$. This result is stated below in Corollary 4.1. Following this corollary, we provide upper bounds on the error accumulated while computing a set $\bigcup_{x\in P(d)} \mathscr{E}(X(d,x))$ in Propositions 4.1 through 4.5 and Theorem 4.1.

Corollary 4.1 ([8]). *Let $X = [l,u]^n$ be a hypercube with non-empty interior, and $f : X \to \mathbb{R}^p$ be continuous and strictly convex. Let $S \subseteq \mathbb{S}^{n-1}$ be a finite subset of the unit sphere in $\mathbb{R}^n$. For each $d \in S$, let $P_m^d = \text{proj}_d(X^m) \subseteq P(d)$. Then, $\lim_{m\to\infty} \bigcap_{d\in S} \bigcup_{x\in P_m^d} \mathscr{E}(X(d,x)) = \bigcap_{d\in S} \bigcup_{x\in P(d)} \mathscr{E}(X(d,x))$ in the Painlevé-Kuratowski sense.*

We begin our error analysis by proving Proposition 4.1 which gives an upper bound of the error for computing $P(d)$. In particular, this result quantifies how far apart neighboring points are in the set P.

Proposition 4.1. *Let $n,m \geq 2, d \in \mathbb{S}^{n-1}$, and $P = \text{proj}_d(X^m)$. Then, for any fixed $x^1 \in P$,*

$$\min_{\substack{x^2 \in P \\ x^2 \neq x^1}} ||x^1 - x^2|| \leq \frac{u-l}{m-1}.$$

Proof. Let $x^1 \in P$ be fixed. Then there exists $v^1 \in X^m$ such that $x^1 = \text{proj}_d(v^1)$. By construction of the discrete hypercube X^m, each point in X^m has neighboring points in at least two different coordinate directions. Therefore, v^1 has a neighboring point, $v^2 \in X^m$, such that $||v^1 - v^2|| = \frac{u-l}{m-1}$ and $\text{proj}_d(v^1) \neq \text{proj}_d(v^2)$. Let $x^2 = \text{proj}_d(v^2)$. Applying Proposition 2.1 we obtain,

$$||x^1 - x^2|| = ||\text{proj}_d(v^1 - v^2)|| \leq ||v^1 - v^2|| = \frac{u-l}{m-1}.$$

It follows that

$$\min_{\substack{x^2 \in P \\ x^2 \neq x^1}} ||x^1 - x^2|| \leq \frac{u-l}{m-1}.$$

□

In Proposition 4.2, we establish a bound on the distance between two line segments in the discrete partition of X defined by $\bigcup_{x \in P} X(d,x)$. This result is then used in Proposition 4.5 to describe the distance between efficient sets on parallel line segments.

To prove Proposition 4.2 we use the following basic lemma.

Lemma 4.1. *Let $\{x^1 + \alpha d : l_1 \leq \alpha \leq u_1\}, \{x^2 + \alpha d : l_2 \leq \alpha \leq u_2\} \subseteq \mathbb{R}^n$ be parallel line segments. Then the largest distance between these two line segments occurs at their endpoints. Specifically,*

$$\begin{aligned}&\Delta_H(\{x^1 + \alpha d : l_1 \leq \alpha \leq u_1\}, \{x^2 + \alpha d : l_2 \leq \alpha \leq u_2\}) \\ &= \max(||x^1 + l_1 d - (x^2 + l_2 d)||, ||x^1 + u_1 d - (x^2 + u_2 d)||).\end{aligned}$$

Proposition 4.2. *Let $X = [l,u]^n, d \in \mathbb{S}^{n-1}$ and $p,q \in P(d)$. Then the Hausdorff distance between the parallel line segments $X(d,p)$ and $X(d,q)$ is*

$$\Delta_H(X(d,p), X(d,q)) \leq ||p-q|| \sqrt{1 + \left(\max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}\right)^2},$$

and the difference in step sizes at the endpoints of $X(d,p)$ and $X(d,q)$ is bounded by

$$\max\{|\alpha_l^p - \alpha_l^q|, |\alpha_u^p - \alpha_u^q|\} \leq ||p-q|| \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}.$$

Proof. We start by proving the upper bound for the difference in step sizes $|\alpha_l^p - \alpha_l^q|$ and $|\alpha_u^p - \alpha_u^q|$. Without loss of generality, we assume $\alpha_u^p \leq \alpha_u^q$. By definition of α_u^p and α_u^q, we have

$$\begin{aligned}\alpha_u^p &:= \max\{\alpha \in \mathbb{R} : l \leq p_i + \alpha d_i \leq u\ , \forall i\} = \min\left\{\frac{l - p_i}{d_i} \text{ for } d_i < 0\ ,\ \frac{u - p_i}{d_i} \text{ for } d_i > 0\right\}, \\ \alpha_u^q &:= \max\{\alpha \in \mathbb{R} : l \leq q_i + \alpha d_i \leq u\ , \forall i\} = \min\left\{\frac{l - q_i}{d_i} \text{ for } d_i < 0\ ,\ \frac{u - q_i}{d_i} \text{ for } d_i > 0\right\}.\end{aligned} \tag{4.1}$$

Therefore $\alpha_u^p = \frac{l-p_j}{d_j}$ or $\alpha_u^p = \frac{u-p_j}{d_j}$ for some $j \in [n]$. Without loss of generality, we suppose $\alpha_u^p = \frac{u-p_j}{d_j}$. By $\alpha_u^p \leq \alpha_u^q$ and (4.1), one has

$$\alpha_u^p = \frac{u-p_j}{d_j} \leq \alpha_u^q \leq \frac{u-q_j}{d_j}.$$

Subtracting α_u^p from both sides of both inequalities, we obtain

$$0 \leq \alpha_u^q - \alpha_u^p \leq \frac{u-q_j}{d_j} - \frac{u-p_j}{d_j} = \frac{p_j-q_j}{d_j}.$$

Since these terms are non-negative, it follows that

$$|\alpha_u^q - \alpha_u^p| \leq \left|\frac{p_j-q_j}{d_j}\right|. \tag{4.2}$$

A similar argument shows that

$$|\alpha_l^q - \alpha_l^p| \leq \left|\frac{p_i-q_i}{d_i}\right| \tag{4.3}$$

for some $i \in [n]$. Using (4.2), (4.3), and the fact that $|p_i - q_i| \leq ||p-q||$ for all $i \in [n]$, we may bound the maximum difference in step sizes by

$$\max\{|\alpha_u^q - \alpha_u^p|, |\alpha_l^q - \alpha_l^p|\} \leq ||p-q|| \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}. \tag{4.4}$$

We now prove the upper bound for $\Delta_H(X(d,p),X(d,q))$. Observe that X being convex implies that

$$X(d,p) = \{p + \alpha d : \alpha_l^p \leq \alpha \leq \alpha_u^p\}$$

and

$$X(d,q) = \{q + \alpha d : \alpha_l^q \leq \alpha \leq \alpha_u^q\}.$$

By Lemma 4.1, the largest distance between these sets occurs at their endpoints. In particular,

$$\Delta_H(X(d,p),X(d,q)) = \max\left(||p + \alpha_l^p d - (q + \alpha_l^q d)||, ||p + \alpha_u^p d - (q + \alpha_u^q d)||\right).$$

We will bound this maximum distance between the endpoints by analyzing right triangles, where the hypotenuse represents the line segment connecting the two endpoints. Consider Figure 1 and the triangle defined by the points

$$A := p + \alpha_u^p d \ , \ B := q + \alpha_u^p d \ , \ C := q + \alpha_u^q d \ .$$

It can be shown that these points form a right triangle by using the fact that $(p-q) \perp d$ and verifying that side AB is in the direction of $p-q$ and side BC is in the direction of d. Note that the hypotenuse, AC, of this right triangle is exactly the line segment connecting the endpoints $p + \alpha_u^p d$ and $q + \alpha_u^q d$.

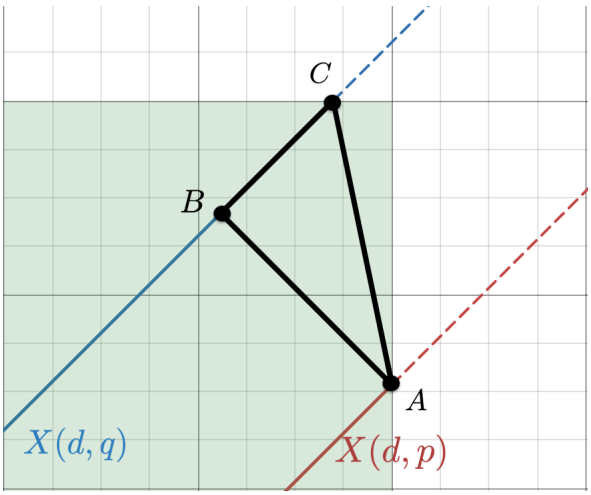


FIGURE 1. Triangle ABC

First, we calculate the two side lengths AB and BC by

$$||A-B|| = ||p+\alpha_u^p d-(q+\alpha_u^p d)|| = ||p-q||,$$
$$||B-C|| = ||q+\alpha_u^p d-(q+\alpha_u^q d)|| = |\alpha_u^p-\alpha_u^q|\cdot||d|| = |\alpha_u^p-\alpha_u^q|.$$

Then the hypotenuse, AC, is bounded above by

$$||A-C||^2 = ||A-B||^2+||B-C||^2 = ||p-q||^2+|\alpha_u^p-\alpha_u^q|^2$$
$$\leq ||p-q||^2+||p-q||^2\cdot\left(\max\left\{\frac{1}{|d_i|}: d_i\neq 0\right\}\right)^2,$$

where the last inequality results from (4.4). Taking a square root on both sides of this inequality yields

$$||p+\alpha_u^p d-(q+\alpha_u^q d)|| = ||A-C|| \leq ||p-q||\sqrt{1+\left(\max\left\{\frac{1}{|d_i|}: d_i\neq 0\right\}\right)^2}.$$

A similar argument holds for computing a bound on the distance between the lower endpoints, $p+\alpha_l^p d$ and $q+\alpha_l^q d$. Therefore the distance between the line segments is bounded by

$$\Delta_H(X(d,p),X(d,q)) \leq ||p-q||\sqrt{1+\left(\max\left\{\frac{1}{|d_i|}: d_i\neq 0\right\}\right)^2}.$$

□

Next, we bound the distance between parallel efficient line segments. To do this, we apply Theorem 2.4. Fix $\bar{d}\in\mathbb{S}^{n-1}$ and define the set of parameters $\Xi(\bar{d})\subset\mathbb{R}^{n+2}$ by

$$\Xi(\bar{d}) := \{(x,\alpha_l^x,\alpha_u^x): x\in P(\bar{d})\}\text{ , where }\alpha_l^x\text{ and }\alpha_u^x\text{ are defined in Definition 2.3.}$$

For any $\xi=(x,\alpha_l^x,\alpha_u^x)\in\Xi(\bar{d})$, line search problem (3.1) can be reformulated to assume the form of the parametric nonlinear problem (2.2). Introducing an auxiliary variable $v\in\mathbb{R}^n$, that

yields equality constraints, and converting the line search bounds into inequality constraints, we obtain problem NLP(ξ) below

$$\begin{aligned}(\text{NLP}(\xi)) \qquad & \min_{v\in\mathbb{R}^n,\alpha\in\mathbb{R}} f_r(v)\\ & \text{s.t. } h_j(v,\alpha,\xi) = v_j - (x_j + \alpha\bar{d}_j) = 0\ ,\forall j\in[n],\\ & \quad g_1(v,\alpha,\xi) = \alpha - \alpha_u^x \le 0,\\ & \quad g_2(v,\alpha,\xi) = \alpha_l^x - \alpha \le 0.\end{aligned}$$

In NLP(ξ), v and α are the decision variables. Theorem 2.4 describes the behavior of the solutions to NLP(ξ) in a neighborhood of $\bar{\xi}\in\Xi(\bar{d})$. Specifically, Theorem 2.4 ensures Lipschitz continuity of the solution maps to NLP(ξ), which are denoted by $\hat{v}:\Xi(\bar{d})\to\mathbb{R}^n$ and $\hat{\alpha}:\Xi(\bar{d})\to\mathbb{R}$. For more details on quantifying the Lipschitz constant, we refer the reader to [10] as this is highly problem specific.

Proposition 4.3 ensures that one of the necessary conditions of Theorem 2.4 is satisfied, while Proposition 4.4 applies this theorem to conclude that the solution maps $\hat{v},\hat{\alpha}$ are defined on all of $\Xi(\bar{d})$ and are Lipschitz continuous.

Proposition 4.3. *Let $\bar{d}\in\mathbb{S}^{n-1}$ be fixed. Then every feasible point (v,α) of NLP(ξ) satisfies the linear independence constraint qualification (LICQ) condition for all $\xi\in\Xi(\bar{d})$.*

Proof. Let $(v,\alpha)\in\mathbb{R}^{n+1}$ be feasible to NLP(ξ). By definition of $\Xi(\bar{d})$, there exists $x\in P(\bar{d})$ such that $\xi=(x,\alpha_l^x,\alpha_u^x)$. Furthermore, by Definition 2.3, we must have $\alpha_l^x\le\alpha_u^x$.

First, we consider the case when $\alpha_l^x<\alpha_u^x$ and suppose that g_1 is active at (v,α). This implies $\alpha=\alpha_u^x$. Then, using the fact that $\alpha_l^x<\alpha_u^x$, we may conclude that $g_2(v,\alpha,\xi)=\alpha_l^x-\alpha=\alpha_l^x-\alpha_u^x<0$. Thus g_2 is inactive. A similar argument can show that g_2 being active at (v,α) implies g_1 is inactive. Therefore, both g_1 and g_2 cannot be simultaneously active. It follows that the set of active constraints at (v,α) is either $\{\nabla g_1,\nabla h_1,\dots,\nabla h_n\}$, $\{\nabla g_2,\nabla h_1,\dots,\nabla h_n\}$, or $\{\nabla h_1,\dots,\nabla h_n\}$. Observe that

$$\nabla h_j(v,\alpha,\xi) = (e^j,-d_j)\in\mathbb{R}^{n+1}, \tag{4.5}$$

$$\nabla g_1(v,\alpha,\xi) = (0^n,1)\in\mathbb{R}^{n+1}, \tag{4.6}$$

$$\nabla g_2(v,\alpha,\xi) = (0^n,-1)\in\mathbb{R}^{n+1}. \tag{4.7}$$

Using equations (4.5)-(4.7), it can easily be verified that the gradient vectors in each set are linearly independent. Thus, (v,α) satisfies LICQ.

Next, we consider the case when $\alpha_l^x=\alpha_u^x$. Then $g_1=-g_2$ and hence these two inequality constraints are equivalent to the following equality constraint, $g_1(v,\alpha,\xi)=\alpha-\alpha_u^x=0$. Thus NLP($\xi$) can be reformulated into a problem with only equality constraints defined by the functions g_1 and h_j for all $j\in[n]$. Since the gradient vectors in $\{\nabla g_1,\nabla h_1,\dots,\nabla h_n\}$ are linearly independent, then (v,α) satisfies LICQ. □

Proposition 4.4. *Let $\bar{d}\in\mathbb{S}^{n-1}$ be fixed, and $f_r:\mathbb{R}^n\to\mathbb{R}$ be twice continuously differentiable and have a positive definite Hessian matrix. Then there exist Lipschitz continuous maps $\hat{v}:\Xi(\bar{d})\to\mathbb{R}^n$ and $\hat{\alpha}:\Xi(\bar{d})\to\mathbb{R}$ such that $(\hat{v}(\xi),\hat{\alpha}(\xi))$ is a minimizer to NLP(ξ) for all $\xi\in\Xi(\bar{d})$.*

Proof. To prove this result, we apply Theorem 2.4 and first show that there exists a minimizer, (v^*,α^*), to NLP(ξ) such that:

i) the LICQ condition holds at (v^*, α^*), and
ii) the Hessian matrix of f_r is positive definite at (v^*, α^*).

Let $\xi \in \Xi(\bar{d})$. Since the Hessian matrix of f_r is positive definite, then f_r is strictly convex. Note that the feasible set of NLP(ξ) is also convex. Therefore there exists a unique minimizer, $(v^*, \alpha^*) \in \mathbb{R}^{n+1}$, to NLP($\xi$). It follows from Proposition 4.3 that (v^*, α^*) satisfies the LICQ condition. Also, by assumption the Hessian matrix of f_r is positive definite. Therefore (v^*, α^*) satisfies condition ii). By Theorem 2.4, there exist Lipschitz continuous solution maps defined on a neighborhood of $\xi \in \Xi(\bar{d})$. Since $\xi \in \Xi(\bar{d})$ is arbitrary, then there exist Lipschitz continuous solution maps around every $\xi \in \Xi(\bar{d})$. Also, $\Xi(\bar{d})$ is compact for each $\bar{d}$ and therefore the solution maps exist globally on $\Xi(\bar{d})$ and are Lipschitz continuous. □

Below, in Proposition 4.5, we bound the distance between efficient sets on two parallel line segments $X(d, x^1)$ and $X(d, x^2)$.

Proposition 4.5. *Let $X = [l, u]^n \subseteq \mathbb{R}^n$ and $f : X \to \mathbb{R}^p$ be strictly convex and twice continuously differentiable. Fix $d \in \mathbb{S}^{n-1}$ and let $x^1, x^2 \in P(d)$. Then there exists $C > 0$ such that*

$$\Delta_H(\mathscr{E}(X(d, x^1)), \mathscr{E}(X(d, x^2))) \leq C||x^1 - x^2|| \cdot \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}.$$

Proof. First, we show that $\mathscr{E}(X(d, x^1))$ and $\mathscr{E}(X(d, x^2))$ can be written as line segments of the form

$$\{x + \alpha d : l_1 \leq \alpha \leq u_1\},$$

and apply Lemma 4.1 to find $\Delta_H(\mathscr{E}(X(d, x^1)), \mathscr{E}(X(d, x^2)))$. Then we use geometric arguments and apply Proposition 4.4 to obtain an upper bound on this distance. Define $\xi^1 := (x^1, \alpha_l^{x^1}, \alpha_u^{x^1})$ and $\xi^2 := (x^2, \alpha_l^{x^2}, \alpha_u^{x^2})$. By Theorem 2.1, the efficient line segments, $\mathscr{E}(X(d, x^1))$ and $\mathscr{E}(X(d, x^2))$, are defined by the minimizers of NLP(ξ^1) and NLP(ξ^2) for each objective function $f_r, r \in [p]$. We compute these efficient sets by solving the following line search problems.

$$\begin{aligned}(\hat{v}^r(\xi^1), \hat{\alpha}_r(\xi^1)) := \underset{(v,\alpha)\in\mathbb{R}^{n+1}}{\arg\min}\ & f_r(v), \\ \text{s.t. } & v - (x^1 + \alpha d) = 0, \\ & \alpha_l^{x^1} \leq \alpha \leq \alpha_u^{x^1}, \\ (\hat{v}^r(\xi^2), \hat{\alpha}_r(\xi^2)) := \underset{(v,\alpha)\in\mathbb{R}^{n+1}}{\arg\min}\ & f_r(v), \\ \text{s.t. } & v - (x^2 + \alpha d) = 0, \\ & \alpha_l^{x^2} \leq \alpha \leq \alpha_u^{x^2}.\end{aligned}$$

Let $j, t, s, w \in [p]$ be the indices of the objective functions that determine the efficient sets on the two parallel line segments

$$\begin{aligned}\hat{\alpha}_j(\xi^1) = \min_r \hat{\alpha}_r(\xi^1), \quad \hat{\alpha}_t(\xi^1) = \max_r \hat{\alpha}_r(\xi^1), \\ \hat{\alpha}_s(\xi^2) = \min_r \hat{\alpha}_r(\xi^2), \quad \hat{\alpha}_w(\xi^2) = \max_r \hat{\alpha}_r(\xi^2).\end{aligned} \tag{4.8}$$

Then, by Theorem 2.1, the efficient sets are

$$\mathscr{E}(X(d,x^1)) = \{x^1 + \alpha d : \hat{\alpha}_j(\xi^1) \leq \alpha \leq \hat{\alpha}_t(\xi^1)\}$$

and

$$\mathscr{E}(X(d,x^2)) = \{x^2 + \alpha d : \hat{\alpha}_s(\xi^2) \leq \alpha \leq \hat{\alpha}_w(\xi^2)\}.$$

Applying Lemma 4.1, the Hausdorff distance between parallel efficient line segments is,

$$\begin{aligned}&\Delta_H(\mathscr{E}(X(d,x^1)),\mathscr{E}(X(d,x^2)))\\ &= \max\{||x^1 + \hat{\alpha}_j(\xi^1)d - (x^2 + \hat{\alpha}_s(\xi^2)d)||, ||x^1 + \hat{\alpha}_t(\xi^1)d - (x^2 + \hat{\alpha}_w(\xi^2)d)||\}.\end{aligned}$$

Since the optimal solutions, $(\hat{v}^r(\xi^1), \hat{\alpha}_r(\xi^1))$ and $(\hat{v}^r(\xi^2), \hat{\alpha}_r(\xi^2))$, must be feasible to the two line search problems above, it follows that, for all $r \in [p]$,

$$\hat{v}^r(\xi^1) = x^1 + \hat{\alpha}_r(\xi^1)d, \tag{4.9}$$

$$\hat{v}^r(\xi^2) = x^2 + \hat{\alpha}_r(\xi^2)d. \tag{4.10}$$

Using equations (4.9) and (4.10), the Hausdorff distance between parallel efficient line segments can be re-written as

$$\Delta_H(\mathscr{E}(X(d,x^1)),\mathscr{E}(X(d,x^2))) = \max\{||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||, ||\hat{v}^t(\xi^1) - \hat{v}^w(\xi^2)||\}. \tag{4.11}$$

We bound this Hausdorff distance by first analyzing the distance $||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||$ using geometric arguments. Without loss of generality, assume $\hat{\alpha}_j(\xi^1) \leq \hat{\alpha}_s(\xi^2)$. Consider the right triangle depicted below in Figure 2 and defined by the points

$$\hat{v}^j(\xi^1)\ ,\ \hat{v}^s(\xi^2)\ ,\ x^2 + \hat{\alpha}_j(\xi^1)d.$$

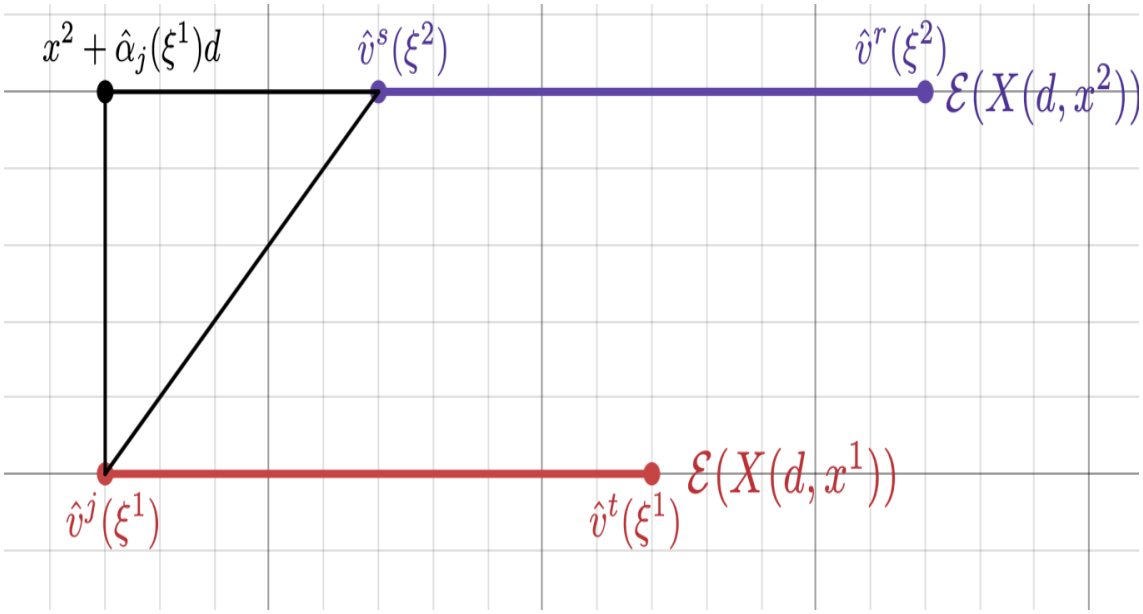


FIGURE 2. Right triangle

The length of the hypotenuse, $\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)$, of this triangle can be calculated applying the Pythagorean theorem

$$||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||^2 = ||x^2 + \hat{\alpha}_j(\xi^1)d - \hat{v}^s(\xi^2)||^2 + ||\hat{v}^j(\xi^1) - (x^2 + \hat{\alpha}_j(\xi^1)d)||^2.$$

Expanding $\hat{v}^s(\xi^2)$ and $\hat{v}^j(\xi^1)$, using (4.9) and (4.10), and using the fact that $||d|| = 1$, we may simply terms to obtain

$$||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||^2 = |\hat{\alpha}_j(\xi^1) - \hat{\alpha}_s(\xi^2)|^2 + ||x^1 - x^2||^2. \tag{4.12}$$

To bound the right-hand side, note that (4.8) implies $\hat{\alpha}_s(\xi^2) \leq \hat{\alpha}_j(\xi^2)$. Since we assumed $\hat{\alpha}_j(\xi^1) \leq \hat{\alpha}_s(\xi^2)$, then we must have

$$\begin{aligned} 0 \leq |\hat{\alpha}_j(\xi^1) - \hat{\alpha}_s(\xi^2)| &= \hat{\alpha}_s(\xi^2) - \hat{\alpha}_j(\xi^1) \\ &\leq \hat{\alpha}_j(\xi^2) - \hat{\alpha}_j(\xi^1) \\ &= |\hat{\alpha}_j(\xi^2) - \hat{\alpha}_j(\xi^1)|. \end{aligned}$$

Using this, we may bound equation (4.12) by

$$\begin{aligned} ||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||^2 &= |\hat{\alpha}_j(\xi^1) - \hat{\alpha}_s(\xi^2)|^2 + ||x^1 - x^2||^2 \\ &\leq |\hat{\alpha}_j(\xi^1) - \hat{\alpha}_j(\xi^2)|^2 + ||x^1 - x^2||^2. \end{aligned} \tag{4.13}$$

Next, it follows from Proposition 4.4 that $\hat{\alpha}_j : \Xi(d) \to \mathbb{R}$ is Lipschitz continuous. Therefore there exists a Lipschitz constant $C_j \in \mathbb{R}$ such that

$$|\hat{\alpha}_j(\xi^1) - \hat{\alpha}_j(\xi^2)| \leq C_j \cdot ||\xi^1 - \xi^2||.$$

To further simplify the right-hand side of the inequality above, we observe that

$$\begin{aligned} ||\xi^1 - \xi^2|| &= \sqrt{\sum_{i=1}^{n} (x_i^1 - x_i^2)^2 + (\alpha_l^{x^1} - \alpha_l^{x^2})^2 + (\alpha_u^{x^1} - \alpha_u^{x^2})^2} \\ &\leq ||x^1 - x^2|| + |\alpha_l^{x^1} - \alpha_l^{x^2}| + |\alpha_u^{x^1} - \alpha_u^{x^2}| \\ &\leq ||x^1 - x^2|| + 2||x^1 - x^2|| \max\left\{ \frac{1}{|\bar{d}_i|} : \bar{d}_i \neq 0 \right\} \\ &= ||x^1 - x^2|| \left(1 + 2 \max\left\{ \frac{1}{|d_i|} : d_i \neq 0 \right\} \right), \end{aligned}$$

where the second inequality follows from Proposition 4.2. Thus

$$|\hat{\alpha}_j(\xi^1) - \hat{\alpha}_j(\xi^2)| \leq C_j \cdot ||x^1 - x^2|| \left(1 + 2 \max\left\{ \frac{1}{|d_i|} : d_i \neq 0 \right\} \right).$$

To obtain a more general upper bound which does not depend on j, we take a maximum over all objective functions to obtain

$$\begin{aligned} |\hat{\alpha}_j(\xi^1) - \hat{\alpha}_j(\xi^2)| &\leq \max_r |\hat{\alpha}_r(\xi^1) - \hat{\alpha}_r(\xi^2)| \\ &\leq C_{max} \cdot ||x^1 - x^2|| \left(1 + 2 \max\left\{ \frac{1}{|d_i|} : d_i \neq 0 \right\} \right), \end{aligned}$$

where $C_{max} = \max_{r\in[p]}(C_r)$. Finally, using this in equation (4.13), we have

$$||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||^2 \leq C_{max}^2 \cdot ||x^1 - x^2||^2 \left(1 + 2\max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}\right)^2 + ||x^1 - x^2||^2.$$

A similar argument shows that the distance between the right-most endpoints of the parallel efficient line segments, v^t and v^w, is also bounded by the same constants

$$||\hat{v}^t(\xi^1) - \hat{v}^w(\xi^2)||^2 \leq C_{max}^2 \cdot ||x^1 - x^2||^2 \left(1 + 2\max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}\right)^2 + ||x^1 - x^2||^2.$$

Taking the square root on both sides of the inequality and factoring out the term $||x^1 - x^2||$, we have

$$\begin{aligned}&\max\{||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||\ ,\ ||\hat{v}^t(\xi^1) - \hat{v}^w(\xi^2)||\}\\ &\leq ||x^1 - x^2||\sqrt{C_{max}^2 \cdot \left(1 + 2\max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}\right)^2 + 1}.\end{aligned} \tag{4.14}$$

It can be shown that there exists $C > 0$ such that

$$\sqrt{C_{max}^2 \cdot \left(1 + 2\max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}\right)^2 + 1} \leq C \cdot \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}.$$

Applying this to Equation (4.14), we obtain

$$\max\{||\hat{v}^j(\xi^1) - \hat{v}^s(\xi^2)||\ ,\ ||\hat{v}^t(\xi^1) - \hat{v}^w(\xi^2)||\} \leq C||x^1 - x^2|| \cdot \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}.$$

Finally, using (4.11) it follows that the Hausdorff distance of efficient line segments is bounded by,

$$\Delta_H(\mathscr{E}(X(d,x^1)), \mathscr{E}(X(d,x^2))) \leq C||x^1 - x^2|| \cdot \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}.$$

□

With the distance between parallel efficient line segments established, we now prove Theorem 4.1, which provides an upper bound on the error in the computation of a set $\bigcup_{x\in P(d)} \mathscr{E}(X(d,x))$. The error is calculated as the Hausdorff distance between two collections of efficient line segments in the direction d. Both collections are calculated as the union of efficient line segments with the difference that one union is taken with respect to the true projection of the feasible set, while the other is taken with respect to a discrete representation of this set.

Theorem 4.1. *Let $X = [l,u]^n \subseteq \mathbb{R}^n$ and $f : X \to \mathbb{R}^p$ be strictly convex and twice continuously differentiable. Let $m \geq 2, d \in \mathbb{S}^{n-1}$ and $P = \text{proj}_d(X^m)$. Then there exists $C > 0$ such that*

$$\Delta_H\left(\bigcup_{x\in P} \mathscr{E}(X(d,x)), \bigcup_{x\in P(d)} \mathscr{E}(X(d,x))\right) \leq C\sqrt{2}\frac{u-l}{(m-1)} \cdot \max\left\{\frac{1}{|d_i|} : d_i \neq 0\right\}.$$

Proof. Since

$$\bigcup_{x\in P} \mathscr{E}(X(d,x)) \subseteq \bigcup_{x\in P(d)} \mathscr{E}(X(d,x)),$$

then the Hausdorff distance in the theorem statement is calculated by

$$\Delta_H\left(\bigcup_{x\in P}\mathscr{E}(X(d,x)),\bigcup_{x\in P(d)}\mathscr{E}(X(d,x))\right)=\sup_{u\in\bigcup\limits_{x\in P(d)}\mathscr{E}(X(d,x))}\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}||u-v||. \tag{4.15}$$

To prove the inequality in the theorem statement, we calculate an upper bound on

$$\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}||u-v||$$

for an arbitrary $u\in\bigcup_{x\in P(d)}\mathscr{E}(X(d,x))$ and apply it to (4.15).

Fix $u\in\bigcup_{x\in P(d)}\mathscr{E}(X(d,x))$ and let $v\in\bigcup_{x\in P}\mathscr{E}(X(d,x))$. Then it can be shown that $\mathrm{proj}_d(u)\in P(d),\mathrm{proj}_d(v)\in P$, and

$$\begin{aligned}u&\in\mathscr{E}(X(d,\mathrm{proj}_d(u))),\\ v&\in\mathscr{E}(X(d,\mathrm{proj}_d(v))).\end{aligned} \tag{4.16}$$

From Proposition 4.5, there exists $C>0$ such that

$$\Delta_H(\mathscr{E}(X(d,\mathrm{proj}_d(u))),\mathscr{E}(X(d,\mathrm{proj}_d(v))))\leq C||\mathrm{proj}_d(u)-\mathrm{proj}_d(v)||\cdot\max\left\{\frac{1}{|d_i|}:d_i\neq 0\right\}.$$

Using (4.16) and the definition of Hausdorff distance, we have

$$\inf_{v'\in\mathscr{E}(X(d,\mathrm{proj}_d(v)))}||u-v'||\leq C||\mathrm{proj}_d(u)-\mathrm{proj}_d(v)||\cdot\max\left\{\frac{1}{|d_i|}:d_i\neq 0\right\}.$$

Taking an infimum over v on both sides we obtain

$$\begin{aligned}&\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}\inf_{v'\in\mathscr{E}(X(d,\mathrm{proj}_d(v)))}||u-v'||\\ &\leq\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}C||\mathrm{proj}_d(u)-\mathrm{proj}_d(v)||\cdot\max\left\{\frac{1}{|d_i|}:d_i\neq 0\right\}.\end{aligned} \tag{4.17}$$

Note that the left-hand side of (4.17) can be simplified to

$$\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}\inf_{v'\in\mathscr{E}(X(d,\mathrm{proj}_d(v)))}||u-v'||=\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}||u-v||. \tag{4.18}$$

To simplify the right-hand side of (4.17), we bound the infimum of $||\mathrm{proj}_d(u)-\mathrm{proj}_d(v)||$. By construction of X^m, it can be shown that $\Delta_H(X^m,X)=\sqrt{2}\frac{u-l}{(m-1)}$. Since $u\in X$, there exists $u'\in X^m$ such that $||u-u'||\leq\sqrt{2}\frac{u-l}{(m-1)}$. Applying Proposition 2.1, we also have

$$||\mathrm{proj}_d(u-u')||\leq||u-u'||\leq\sqrt{2}\frac{u-l}{(m-1)}. \tag{4.19}$$

Then we may bound the following infimum by

$$\begin{aligned}\inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(u)-\text{proj}_d(v')|| &= \inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(u-u')-\text{proj}_d(v'-u')|| \\ &\leq \inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(u-u')||+||\text{proj}_d(v'-u')|| \\ &= ||\text{proj}_d(u-u')||+\inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(v'-u')|| \\ &\leq \sqrt{2}\frac{u-l}{(m-1)}+\inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(v'-u')||, \end{aligned} \tag{4.20}$$

where the first equality uses the linearity property of projection operators, the first inequality comes from the triangle inequality, and the second inequality comes from (4.19). To compute the infimum in (4.20), observe that $u' \in X^m$ implies that $\text{proj}_d(u') \in P$ and $u' \in X(d,\text{proj}_d(u'))$. Therefore, $u' \in \bigcup_{x\in P}\mathscr{E}(X(d,x))$ and hence

$$0 \leq \inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(v'-u')|| \leq ||\text{proj}_d(u'-u')|| = 0,$$

which implies the value of the above infimum is zero. Therefore, we may simplify (4.20) to obtain

$$\inf_{v'\in\bigcup_{x\in P}\mathscr{E}(X(d,x))}||\text{proj}_d(u)-\text{proj}_d(v')|| \leq \sqrt{2}\frac{u-l}{(m-1)}.$$

Applying the above inequality to simplify the right-hand side of (4.17), and using (4.18) to simplify the left-hand side of (4.17), we have

$$\inf_{v\in\bigcup\limits_{x\in P}\mathscr{E}(X(d,x))}||u-v|| \leq C\sqrt{2}\frac{u-l}{(m-1)}\cdot\max\left\{\frac{1}{|d_i|}:d_i\neq 0\right\}.$$

Since $u \in \bigcup_{x\in P(d)}\mathscr{E}(X(d,x))$ was arbitrary, then

$$\Delta_H\left(\bigcup_{x\in P}\mathscr{E}(X(d,x)),\bigcup_{x\in P(d)}\mathscr{E}(X(d,x))\right) \leq C\sqrt{2}\frac{u-l}{(m-1)}\cdot\max\left\{\frac{1}{|d_i|}:d_i\neq 0\right\}.$$

□

Theorem 4.1 indicates that the error in the computation of a set $\bigcup_{x\in P(d)}\mathscr{E}(X(d,x))$, depends on the MOP data and the algorithm parameters. The former includes the size of the feasible set and the Lipschitz constant associated with line search solutions, while the latter includes the number of discretization points and the optimization directions.

In the pseudocode, each iteration of the algorithm consists of two steps: computing a set U_i and updating the approximate efficient set, E. In Theorem 4.1, we quantify the error in computing the set U_i. However, evaluating the error associated with updating E is significantly more challenging. This difficulty arises because updating E essentially involves analyzing the intersection of two sets, U_1 and U_2, and determining the distance of their intersection to the true efficient set. Describing such intersections requires detailed information about the boundaries of these sets.

According to Theorem 2.1, the efficient line segments in the union U_i are determined by the solutions to NLP(ξ) for each objective function f_r, where $r \in [p]$. Consequently, the boundary

of U_i can, in principle, be described by the solution maps $(\hat{v}^r, \hat{\alpha}_r)$. However, these solution maps are highly problem-specific, and explicit formulas for them are not always obtainable. For this reason, the available information seems to be insufficient to determine the intersection error analytically. In the next section, we provide experimental results of the error of the approximate solution set E for various examples.

5. Computational Results

In this section, we apply the line decomposition algorithm to various bi-objective and single-objective test problems. The source code for all examples was executed on a MacBook Pro with Dual-Core Intel Core i7 processor. For the biobjective problems (BOPs) we use quadratic objective functions, and the single-objective problems are built on polynomial functions selected from [11]. For each example we report the algorithm input parameters including the number of discretization points, m, and the number of directions, k, as well as the error of the approximate solution. The metric used to measure error is the Hausdorff distance of the approximate efficient set, E, to the true efficient set, $\mathscr{E}(X)$.

$$\Delta_H(\mathscr{E}(X),E) = \max\left(\sup_{u\in\mathscr{E}(X)}\inf_{v\in E}||u-v||\ ,\ \sup_{v\in E}\inf_{u\in\mathscr{E}(X)}||v-u||\right).$$

For all our examples, we choose the set of line directions, S, so that it grows monotonically. Meaning that as the number of directions used increases from $k_1 \in \mathbb{N}$ to $k_2 \in \mathbb{N}$, the first k_1 directions in S will remain the same while $(k_2 - k_1)$ new directions are added. By consistently using the same directions at the beginning of the algorithm, we are able to observe the impact of performing additional iterations, and help the approximation error decrease as the parameter k increases.

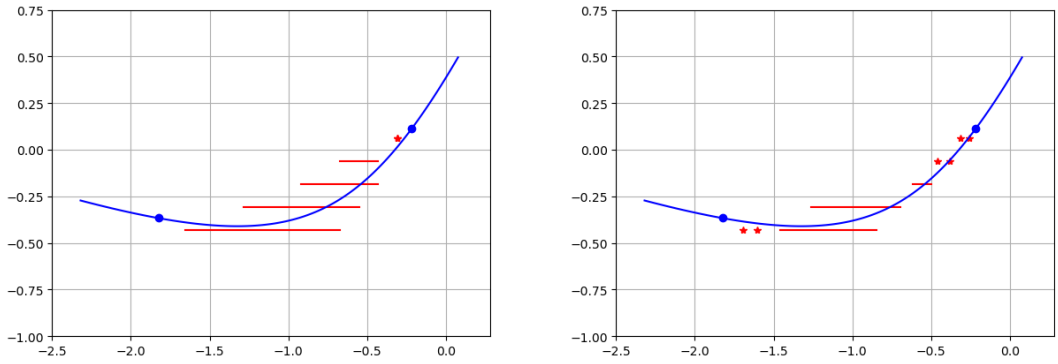
5.1. Biobjective examples in $\mathbb{R}^2$.

In Examples 5.1 - 5.4, we display pictures of the approximate solution sets generated by the algorithm. In each picture, the true efficient set is the portion of the blue curve between the two marked points, that are the individual minimizers to each objective function. This blue curve is drawn according to the formula in [12] for the efficient set of biobjective quadratic problems. The collection of red line segments is the approximate solution set produced by the algorithm. In our implementation, these red lines are always horizontal since the line direction we use to generate the initial approximation, U_1, is always $d^1 = (1,0)$.

Example 5.1. Consider the BOP below

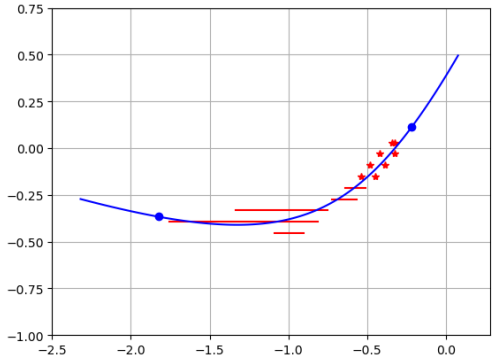
$$\begin{aligned}
\min\ f_1(x) &= 0.5\cdot x^T \begin{bmatrix} 12 & 3 \\ 3 & 26 \end{bmatrix} x + \begin{bmatrix} 23 & 15 \end{bmatrix} x \\
f_2(x) &= 0.5\cdot x^T \begin{bmatrix} 20 & -6 \\ -6 & 15 \end{bmatrix} x + \begin{bmatrix} 5 & -3 \end{bmatrix} x \\
\text{s.t. } & -3 \le x_i \le 3\ , i = 1,2.
\end{aligned}$$

Using various values for parameters m and k, we run the line decomposition algorithm to produce the approximate solutions that are depicted in Figure 3.

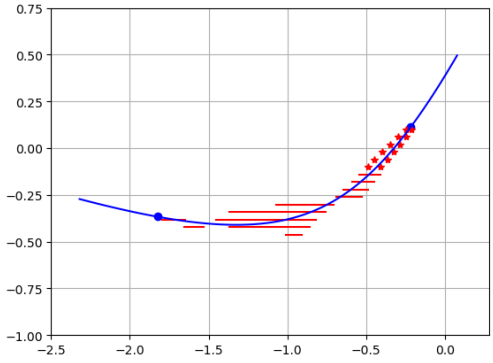


(A) $k = 4, m = 50$

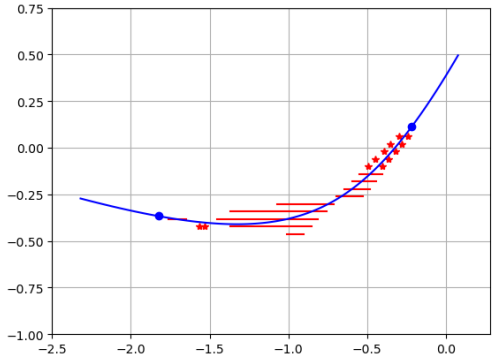
(B) $k = 8, m = 50$



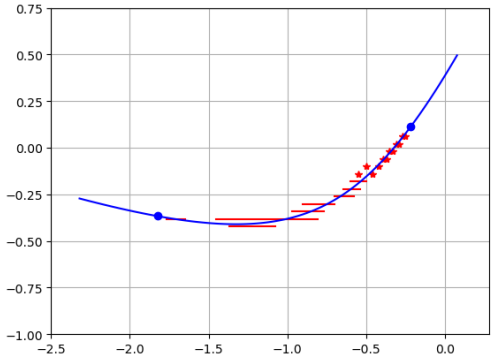
(C) $k = 10, m = 100$



(D) $k = 10, m = 150$



(E) $k = 12, m = 150$



(F) $k = 16, m = 150$

FIGURE 3. Approximate efficient sets for Example 5.1

The error for each approximate solution produced is given in Table 1.

TABLE 1. Example 5.1 results summary

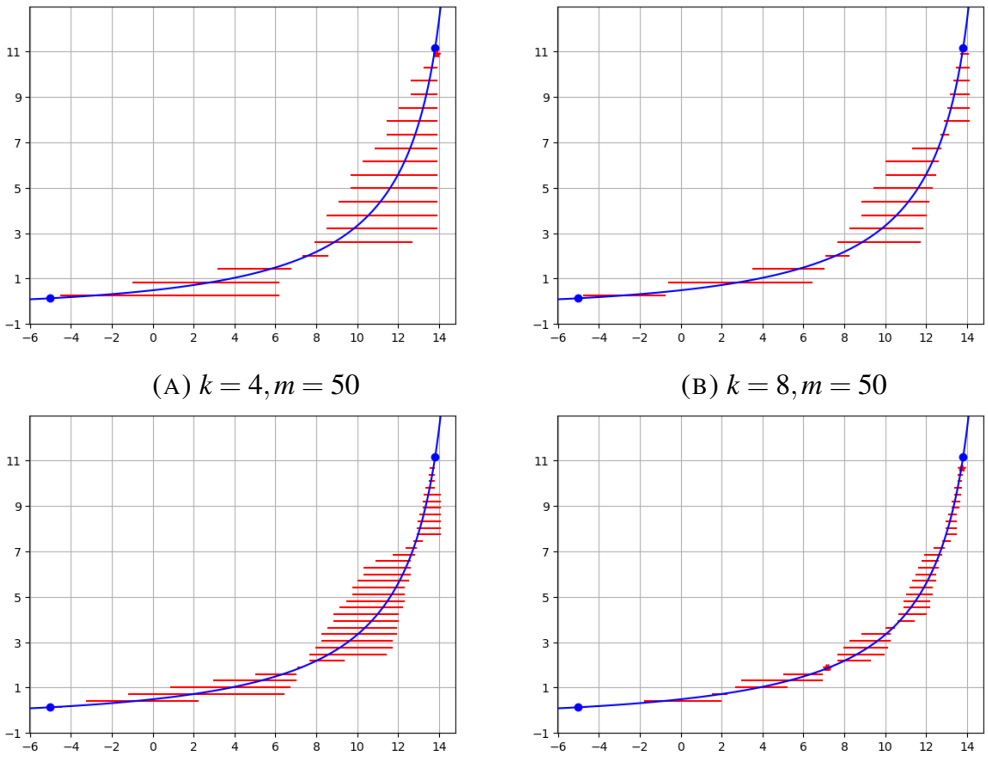
k	4	8	10	10	12	16
m	50	50	100	150	150	150
approximation error	0.1794	0.1440	0.1368	0.1012	0.1021	0.0986

In Table 1, we observe the approximation error consistently decreases as k increases.

Example 5.2. Consider the BOP below

$$\begin{aligned} \min\ f_1(x) &= 0.5 \cdot x^T \begin{bmatrix} 0.1 & 0.028 \\ 0.004 & 0.6 \end{bmatrix} x + \begin{bmatrix} 0.5 & 0 \end{bmatrix} x \\ f_2(x) &= 0.5 \cdot x^T \begin{bmatrix} 0.35 & 0.015 \\ 0.015 & 0.25 \end{bmatrix} x + \begin{bmatrix} -5 & -3 \end{bmatrix} x \\ \text{s.t. } & -11 \leq x_i \leq 18\ , i = 1,2. \end{aligned}$$

Using various parameter values, we run the line decomposition algorithm to produce the approximate solution sets in Figure 4.



(A) $k = 4, m = 50$ (B) $k = 8, m = 50$

(C) $k = 12, m = 100$ (D) $k = 16, m = 100$

FIGURE 4. Approximate efficient sets for Example 5.2

A summary of the computational results is given in Table 2.

TABLE 2. Example 5.2 results summary

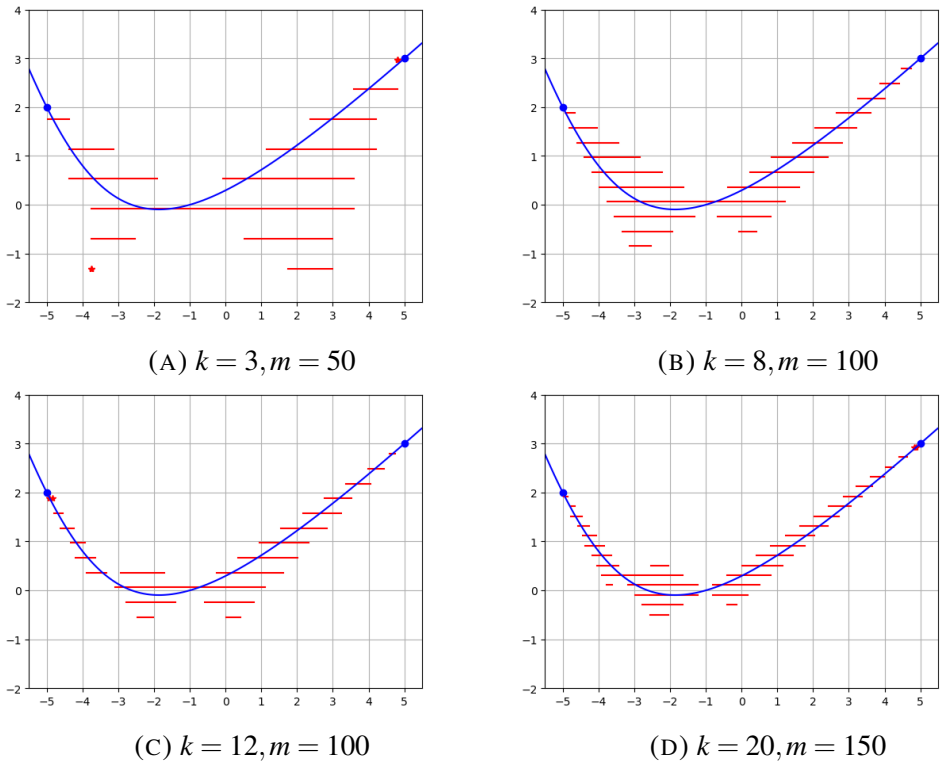
k	4	8	12	16
m	50	50	100	100
approximation error	2.8728	1.9371	1.7502	1.4528

In Table 2, we again observe the approximation error decreases as k increases. However, the magnitude of the approximation error is much larger than in Example 5.1. This results from the fact that the feasible region in Example 5.2 is larger than in Example 5.1, but the number of points used to represent X remains the same (i.e., we use similar values of m in both examples).

Example 5.3. Consider the BOP below

$$\begin{aligned} \min\ f_1(x) &= 0.5 \cdot x^T \begin{bmatrix} 6 & 1 \\ 1 & 2 \end{bmatrix} x + \begin{bmatrix} 28 & 1 \end{bmatrix} x \\ f_2(x) &= 0.5 \cdot x^T \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} x + \begin{bmatrix} -7 & -1 \end{bmatrix} x \\ \text{s.t. } & -16 \leq x_i \leq 14\ , i = 1,2. \end{aligned}$$

Using various parameter values, we run the line decomposition algorithm to produce the approximate solution sets in Figure 5.



(A) $k = 3, m = 50$ (B) $k = 8, m = 100$

(C) $k = 12, m = 100$ (D) $k = 20, m = 150$

FIGURE 5. Approximate efficient sets for Example 5.3

A summary of the computational results is given in Table 3.

TABLE 3. Example 5.3 results summary

k	3	8	12	20
m	50	100	100	150
approximation error	2.6926	0.9593	0.9311	0.6003

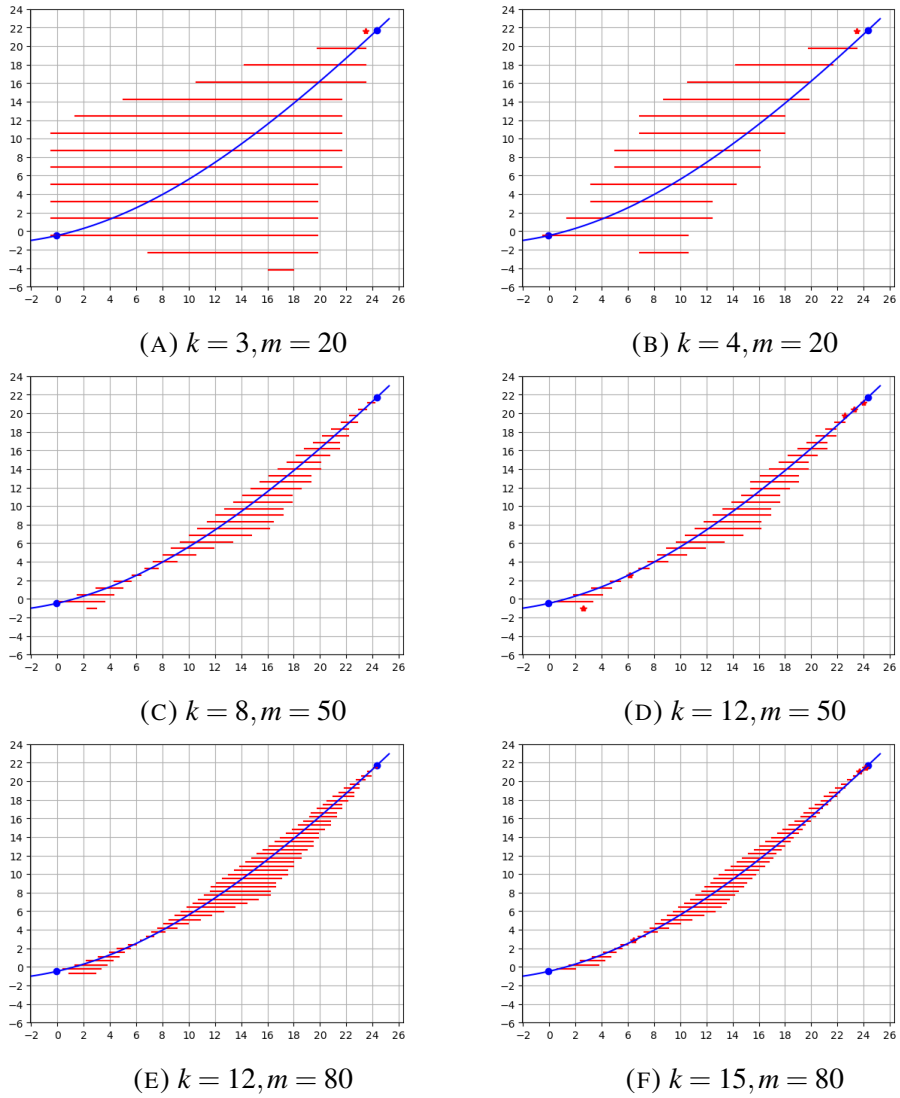
Observe that the errors reported in Table 3 for $k = 8$ and $k = 12$ are very close in decimal value to each other. Visually, however, we observe a significant improvement in the approximate solution sets from Figures 5 b) and 5 c). Specifically, the approximation in 5 c) closer fits the blue curve, $\mathscr{E}(X)$, on its left half. The reason the error does not reflect a significant improvement is

because the Hausdorff distance is measured by taking the farthest point in the red approximation to the blue curve. Therefore the error will only decrease significantly if the worst points in the approximation get much closer to $\mathscr{E}(X)$.

Example 5.4. Consider the BOP below

$$\begin{aligned} \min\ f_1(x) &= 0.5 \cdot x^T \begin{bmatrix} 16 & 3 \\ 1 & 10 \end{bmatrix} x + \begin{bmatrix} 2 & 5 \end{bmatrix} x \\ f_2(x) &= 0.5 \cdot x^T \begin{bmatrix} 0.5 & -0.1 \\ -0.1 & 0.25 \end{bmatrix} x + \begin{bmatrix} -10 & -3 \end{bmatrix} x \\ \text{s.t. } & -6 \leq x_i \leq 29\ , i = 1,2. \end{aligned}$$

The approximate solution sets produced by the algorithm are depicted in Figure 6.



(A) $k = 3, m = 20$ (B) $k = 4, m = 20$

(C) $k = 8, m = 50$ (D) $k = 12, m = 50$

(E) $k = 12, m = 80$ (F) $k = 15, m = 80$

FIGURE 6. Approximate efficient sets for Example 5.4

A summary of the computational results is given in Table 4.

TABLE 4. Example 5.4 results summary

k	3	4	8	12	12	15
m	20	20	50	50	80	80
approximation error	12.5797	7.1546	2.8210	2.8378	2.7305	1.5037

In Table 4, we observe an instance where the error decreases when m increases but k remains constant at $k = 12$.

This occurs because as m increases, the set X^m more closely represents the feasible set X. Consequently more lines are used to decompose X which improves accuracy of the algorithm.

Note that in Figure 6 the approximate efficient set improves greatly when increasing k from 3 to 8. However in Figure 5, the shape of the approximate efficient set in Example 5.3 does not change significantly when using $k = 3$ versus using $k = 8$.

Therefore the line directions d^4 through d^8 have a greater impact in improving the approximate efficient set in Example 5.4 versus Example 5.3.

From this, we can conclude that the best choice of line directions might be problem dependent and can greatly influence the accuracy of the approximation.

In addition to Examples 5.1 - 5.4, we test the line decomposition algorithm on randomly generated biobjective quadratic problems of the form

$$\begin{aligned} \min\ & f_1(x) = 0.5 \cdot x^T Q^1 x + (p^1)^T x \\ & f_2(x) = 0.5 \cdot x^T Q^2 x + (p^2)^T x \\ \text{s.t. } & l \leq x_i \leq u\ , i = 1,2, \end{aligned} \tag{BOQP}$$

where $Q^1, Q^2 \in \mathbb{R}^{n\times n}$ are symmetric positive definite matrices, $p^1, p^2 \in \mathbb{R}^n$, and $l, u \in \mathbb{R}$. After the objective functions are defined, the variable bounds l and u are chosen so that the feasible region contains the global individual minimizers of f_1 and f_2.

We run the line decomposition algorithm using $k = 4, 8$, and 16 line directions, and $m = 50, 100$, and 150 discretization points respectively. Depending of the size of the feasible region, the set X^m may be a sparse or dense representation of the hypercube X, which will directly influence the magnitude of the error for most problems.

Therefore, to better compare approximation error of different examples, we state the length of the interval $[l, u]$ for each test problem in the summary of the computational results.

Tables 5 - 7 present the computational results of these test problems. See Table 16 in the appendix for the specific instances of the randomly generated BOPs that we solve.

TABLE 5. Line decomposition algorithm with parameters $k = 4$ and $m = 50$

BOP #	length of interval $[l,u]$	approximation error
9	3	0.0651
2	5	0.2474
3	7	0.3624
4	8	0.3329
8	8	0.6926
5	9	0.2247
7	13	0.7750
6	15	1.1359
10	19	2.0513
1	28	1.1890

TABLE 6. Line decomposition algorithm with parameters $k = 8$ and $m = 100$

BOP #	length of interval $[l,u]$	approximation error
9	3	0.0394
2	5	0.1125
3	7	0.2596
4	8	0.2005
8	8	0.5403
5	9	0.1615
7	13	0.3629
6	15	0.5203
10	19	0.8073
1	28	1.0986

TABLE 7. Line decomposition algorithm with parameters $k = 16$ and $m = 150$

BOP #	length of interval $[l,u]$	approximation error
9	3	0.0247
2	5	0.0864
3	7	0.0891
4	8	0.1992
8	8	0.3690
5	9	0.1145
7	13	0.2984
6	15	0.2974
10	19	0.5223
1	28	0.8112

In Tables 5 - 7 we observe that the error for each BOP decreases as the parameter k increases. Another key observation is that the approximation error is not necessarily smaller when the feasible region is smaller (i.e., when the length of $[l,u]$ is smaller). For instance in Table 7, the error of BOP 8 has a larger approximation error than BOP 6, even though the length of the interval $[l,u]$ is nearly half the size of BOP 6. This can occur because a direction $d^i \in S$ may or may not produce a set U_i which improves the current approximation E during the update. The impact a set U_i has on improving the approximation E depends on both the objective function

and the particular direction d^i. We also observe this behavior when comparing Examples 5.3 and 5.4. As we will see even more clearly in Example 5.6, the choice of direction vectors in S can heavily influence the accuracy of the approximate solution.

In summary, we observe in Examples 5.1 - 5.4, as well as the randomly generated problems in Tables 5 - 7, that the approximation error decreases as we increase the parameter k. We also observe some instances, such as in Example 5.4, where increasing the parameter m causes a decrease in approximation error.

Additionally, in Figures 3 - 6 we can see that the approximate efficient set closely fits the true efficient set and matches its curvature when using the largest value for k. Despite observing a close fit in all of these figures, the magnitude of the approximation error can be greater when the feasible set is larger. This is observed in Examples 5.1 and 5.2. On the other hand, we show in Tables 5 and 7 that just having a larger feasible set does not guarantee that the approximation error will be bigger.

5.2. **Biobjective examples in $\mathbb{R}^3$.** We now test the line decomposition algorithm on biobjective quadratic problems in $\mathbb{R}^3$. Unlike in $\mathbb{R}^2$, we observe that arbitrarily choosing k line directions will generally not produce good approximations of the efficient set in $\mathbb{R}^3$. However, for specially structured problems it is possible to identify which line directions produce good approximations. Example 5.6 demonstrates this by using an analysis from [8] to produce a very good approximate efficient set by only using 6 line directions.

An additional challenge we face in higher dimensions is that the size of discrete set X^m increases exponentially as the dimension of the decision space increases. For many line directions $d^i \in \mathbb{S}^{n-1}$, each point in X^m defines a unique point in P.

Therefore m^3 lines are used to decompose X. However if d^i is a coordinate direction, then only m^2 lines are used to decompose X since not every point in X^m has a unique projection in P.

For this reason we use two different values of m in the algorithm, denoted m_{dense} and m_{sparse}. If d^i is a coordinate direction we use $m_{dense} \in \mathbb{N}$ and $P := \text{proj}_{d^i}(X^{m_{dense}})$ for that iteration of the algorithm. Otherwise, $m_{sparse} \in \mathbb{N}$ and $P := \text{proj}_{d^i}(X^{m_{sparse}})$ is used. We select m_{dense} and m_{sparse} so that $m_{dense}^2 \approx m_{sparse}^3$, which ensures that roughly the same number of lines are used to decompose X in each iteration.

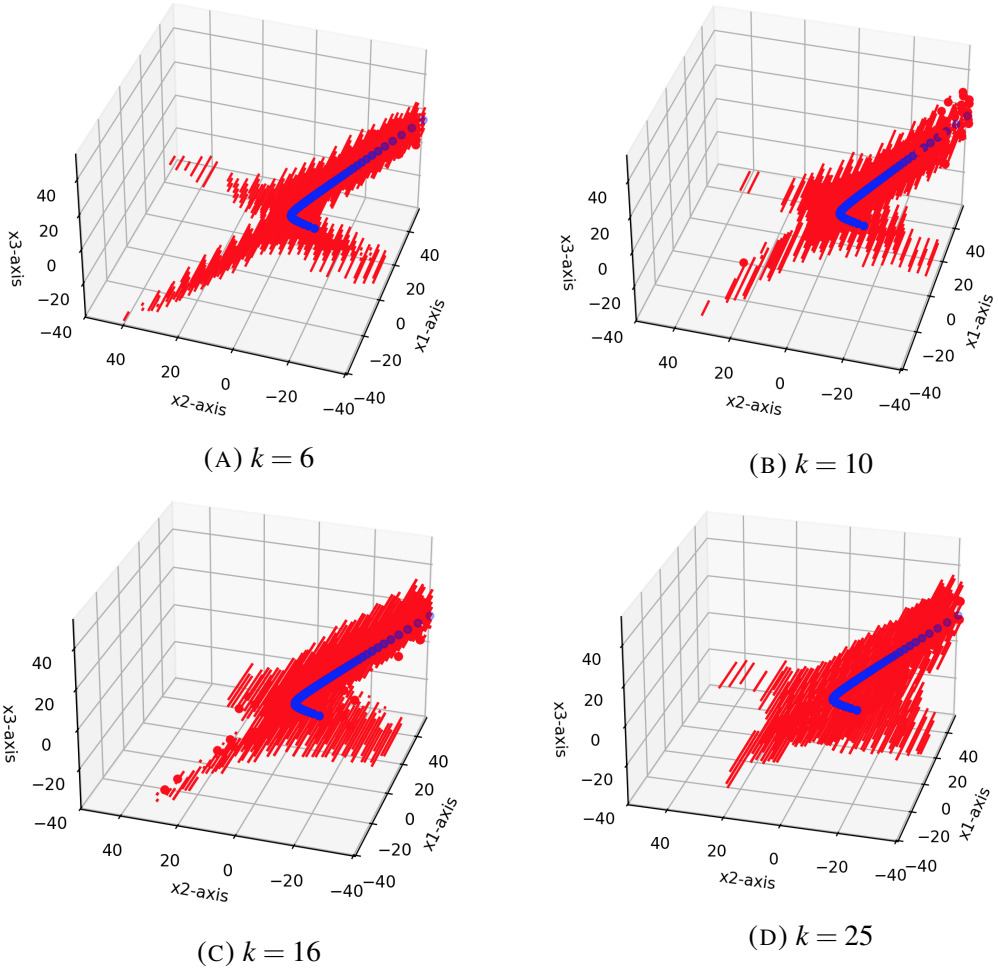
Example 5.5. Consider the BOP below.

$$\min f_1(x) = 0.5 \cdot x^T \begin{bmatrix} 0.5 & 0.025 & -0.1 \\ 0.025 & 0.08 & -0.17 \\ -0.1 & -0.17 & 0.9 \end{bmatrix} x + \begin{bmatrix} -1 \\ 0 \\ 4 \end{bmatrix}^T x$$

$$f_2(x) = 0.5 \cdot x^T \begin{bmatrix} 0.26 & 0.4 & 0.16 \\ 0.4 & 0.85 & 0.5 \\ 0.16 & 0.5 & 0.7 \end{bmatrix} x + \begin{bmatrix} 0 \\ 6 \\ 2 \end{bmatrix}^T x$$

$$\text{s.t. } -40.6413 \leq x_i \leq 53.9613 \ , i = 1,2,3.$$

Using various parameter values, we run the line decomposition algorithm to produce the approximate solution sets in Figure 7.



(A) $k = 6$

(B) $k = 10$

(C) $k = 16$

(D) $k = 25$

FIGURE 7. Approximate efficient sets for Example 5.5 using $m_{sparse} = 10, m_{dense} = 32$.

A summary of the computational results is given in Table 8.

TABLE 8. Example 5.5 results summary

k	6	10	16	25
m_{dense}	32	32	32	32
m_{sparse}	10	10	10	10
approximation error	70.9422	61.9973	60.4589	54.9325

In Table 8 we observe the error decreases consistently as k increases, however visually in Figure 7 the approximation does not appear to improve much as k increases. Comparing Figures 7 a) and 7 d), we see that 7 a) better captures the curvature of the efficient set but this is lost in 7 d).

This can be caused by the tolerance we allow during the execution of the intersection method used to update E (refer to Section 5.4 for details). Unlike the examples in $\mathbb{R}^2$, the algorithm struggles to get a close approximation of $\mathscr{E}(X)$. The approximate solution sets in this example only show the general neighborhood where $\mathscr{E}(X)$ lies, but we cannot get a good sense of the true solution from these approximations.

Example 5.6. Consider the BOP below

$$\begin{aligned}
\min\ & f_1(x) = 0.5 \cdot x^T \begin{bmatrix} 10 & 1 & 1 \\ 1 & 15 & -10 \\ 1 & -10 & 12 \end{bmatrix} x + \begin{bmatrix} 0 \\ 5 \\ -3 \end{bmatrix}^T x \\
& f_2(x) = 0.5 \cdot x^T \left(3 \cdot \begin{bmatrix} 10 & 1 & 1 \\ 1 & 15 & -10 \\ 1 & -10 & 12 \end{bmatrix} \right) x + \begin{bmatrix} 20 \\ -10 \\ 70 \end{bmatrix}^T x \\
\text{s.t. } & -4.1486 \leq x_i \leq 0.2465\ , i = 1,2,3.
\end{aligned}$$

Using an arbitrary selection of $k = 7$ direction vectors, the algorithm produced the solution set depicted in Figure 8.

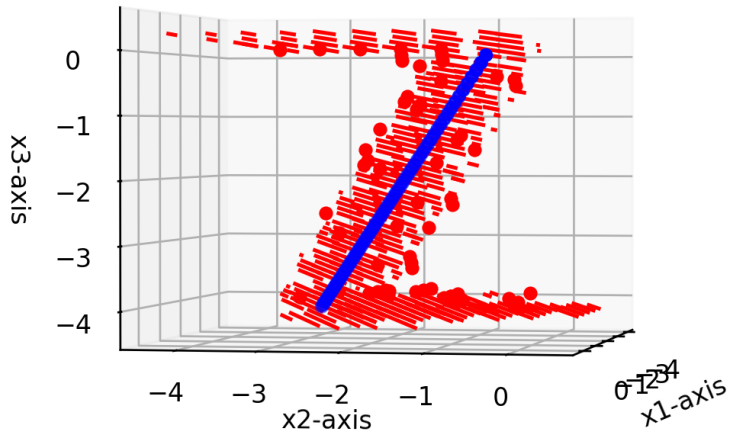
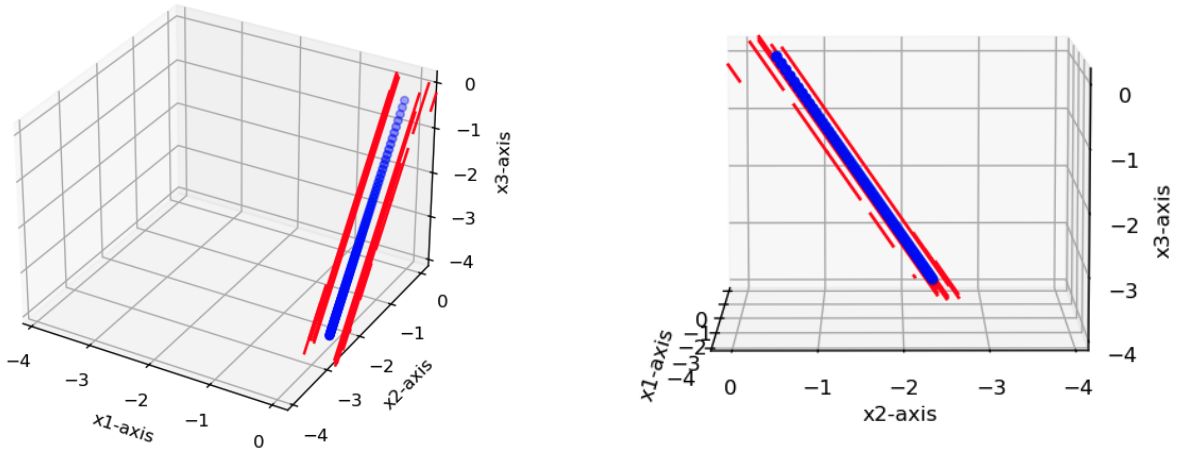


FIGURE 8. Approximate solution for Example 5.6 using $k = 7$ and $m_{sparse} = 10, m_{dense} = 32$.

Note that this BOP has a special structure since the quadratic terms of f_1 and f_2 are multiples. Applying the analysis done in Theorem 6.1 from [8], we take advantage of this special structure to identify the line directions that will produce a better approximation. Let Q^1 denote the matrix in f_1 defining the quadratic terms. We use line directions which form an orthonormal basis with respect to the Q^1-inner product defined by $\langle u,v\rangle_{Q^1} := u^T Q^1 v$, and the standard coordinate directions. Specifically this Q^1-basis includes the direction vector connecting the individual minimizers of f_1 and f_2. Using these 6 direction vectors, the algorithm produces the following solution set depicted in Figure 9.



(A) (B)

FIGURE 9. Approximate solution for Example 5.6 using $k = 6$ and $m_{sparse} = 10, m_{dense} = 32$.

A summary of the computational results is given in Table 9.

TABLE 9. Example 5.6 results summary

k	6	7
m_{dense}	32	32
m_{sparse}	10	10
approximation error	0.6852	4.8048

We observe that the approximate solution set produced by using 7 arbitrary directions has a significantly larger error than the solution set produced using the 6 specially selected directions. Therefore we may conclude that a large number of line directions is not necessarily needed to produce a close approximation of the true efficient set.

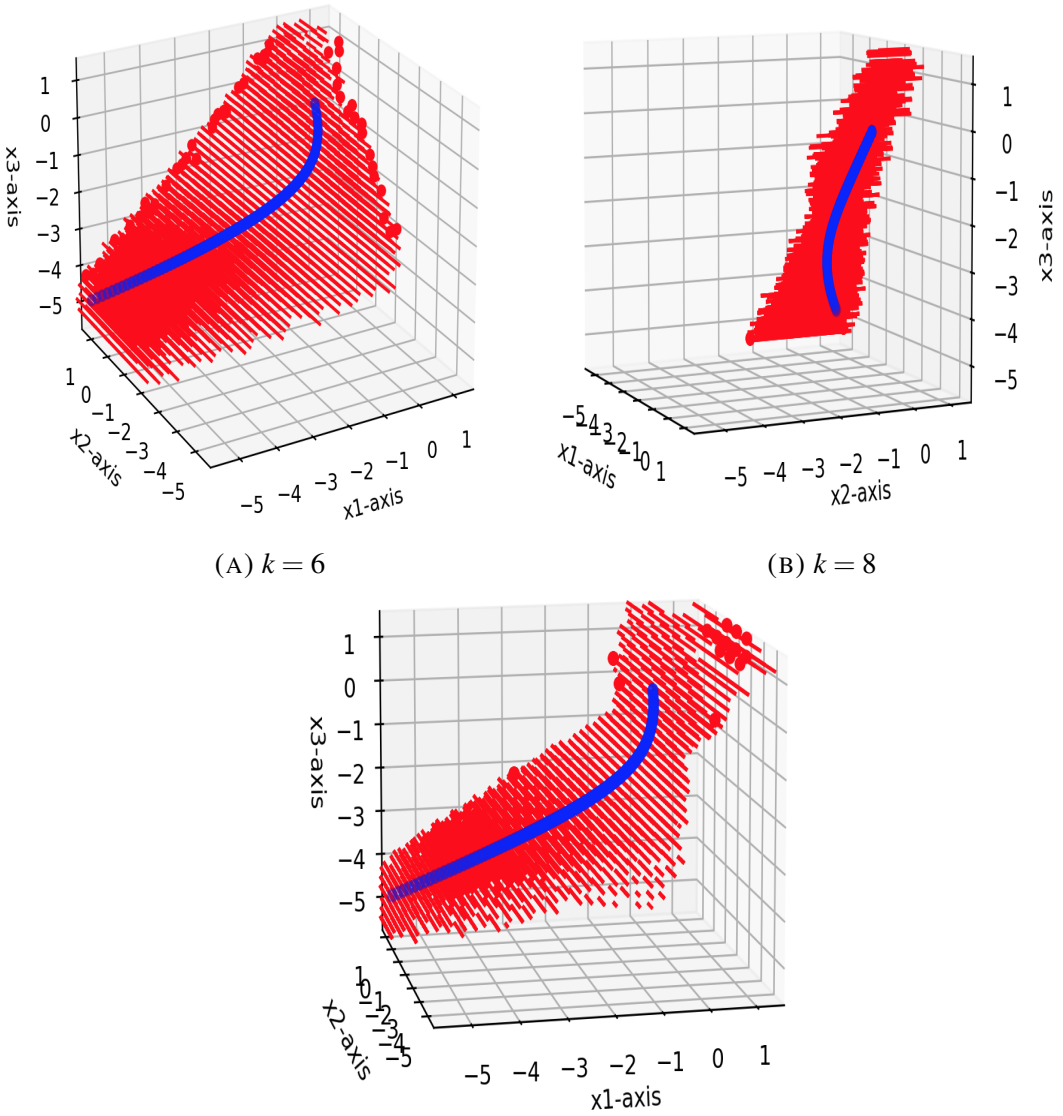
Example 5.7. We approximate the efficient set for the BOP below.

$$\min f_1(x) = 0.5 \cdot x^T \begin{bmatrix} 10 & 1 & 1 \\ 1 & 15 & -10 \\ 1 & -10 & 12 \end{bmatrix} x + \begin{bmatrix} 0 \\ 5 \\ -3 \end{bmatrix}^T x$$

$$f_2(x) = 0.5 \cdot x^T \begin{bmatrix} 8 & 0 & -5 \\ 0 & 9 & 0.5 \\ -5 & 0.5 & 20 \end{bmatrix} x + \begin{bmatrix} 2 \\ -1 \\ 7 \end{bmatrix}^T x$$

$$\text{s.t. } -5.7812 \leq x_i \leq 1.585 \ , i = 1,2,3.$$

Using various parameter values, we run the line decomposition algorithm to produce the approximate solution sets in Figure 10.



(A) $k = 6$ (B) $k = 8$

(C) $k = 16$

FIGURE 10. Approximate efficient sets for Example 5.7 using $m_{sparse} = 10, m_{dense} = 32$.

A summary of the computational results is given in Table 10.

TABLE 10. Example 5.7 results summary

k	6	8	16
m_{dense}	32	32	32
m_{sparse}	10	10	10
approximation error	3.5608	3.2990	4.7792

Although the error decreases in Table 10 as k increases, in Figure 10 we observe that the approximate solution struggles to get a tighter fit of the true efficient set, as is the case in Example 5.5. We may conclude that it becomes necessary in higher dimensions to implement a different

method for selecting line directions. Choosing a random set of directions in $\mathbb{S}^{n-1}$ will generally not produce a good approximate solution.

In summary, we observe in Examples 5.5 - 5.7 that the algorithm struggles to obtain a close approximation of the efficient set when using an arbitrary selection of line directions. In some cases, such as in Example 5.7, the approximate efficient set begins to capture the curvature of the true efficient set but the overall fit around $\mathscr{E}(X)$ remains wide. This differs from Examples 5.1 - 5.4 in $\mathbb{R}^2$ where we are able to obtain a tight fit around $\mathscr{E}(X)$ by simply increasing k. The only case in $\mathbb{R}^3$ where we are able to obtain a tight approximation around $\mathscr{E}(X)$ is in Example 5.6 where we select appropriate line directions by analyzing the specific objective functions. We can therefore conclude that in higher dimensions it becomes necessary to implement a method for selecting line directions which depends on the objective functions.

5.3. **Single-objective examples in $\mathbb{R}^2$.** In Examples 5.8- 5.11, we test the line decomposition algorithm on both convex and nonconvex single-objective problems. We no longer require convexity of the objective function for single-objective problems since this assumption is only needed in Theorem 2.1 that is used to determine the efficient set on a line segment.

In the single-objective setting we found that only two line directions, $k = 2$, are needed to produce a good approximation, whereas the multiobjective case requires numerous line directions. We use less line directions for single-objective optimization since the set E is always a collection of points, rather than line segments. Therefore we can easily determine the best approximate minimizer by evaluating the objective function at these points in E.

To improve efficiency and accuracy of the line decomposition algorithm for single-objective problems, we apply a small modification to the final update of the set E. The first iteration of the algorithm is exactly the same for both single and multiobjective problems. In the second iteration when performing the final update of E, we determine the intersection points of the set $U_1 \cap U_2$, and evaluate the objective function at these points.

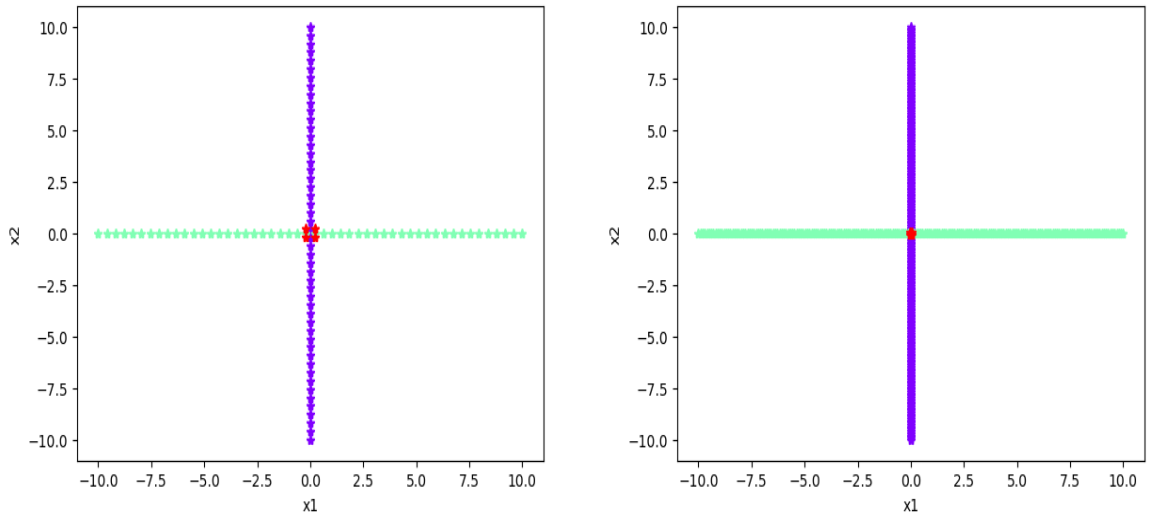
Then E is set equal to the collection of intersection points which have the smallest objective value. This differs from the multiobjective case where E is just set equal to the intersection $U_1 \cap U_2$. For all single-objective examples, the two line directions used in the algorithm are $S = \{(1,0),(0,1)\}$, but in theory any two line directions can be used.

In Examples 5.8 - 5.11 we display pictures of the sets U_1 and U_2, as well as their intersection points from which we select the final approximate minimizers. The sets U_1 and U_2 are represented by purple and green points respectively, and the red points are their computed intersection. The final approximation is a subset of the red points which have the smallest objective value.

Example 5.8. Consider the sum of squares problem below.

$$\begin{aligned} \min\ & f(x) = \sum_{i=1}^{2} i x_i^2 \\ \text{s.t. } & -10 \leq x_i \leq 10. \end{aligned}$$

The sum squares function is strictly convex. Thus, it has no local minimizers except the global one at $x^* = (0,0)$. Using various values of m, we run the line decomposition algorithm to produce Figure 11.



(A) $m = 50$ (B) $m = 200$

FIGURE 11. Intersection points, $\bigcap_{i=1}^{2} \bigcup_{x \in P} \mathscr{E}(X(d^i, x))$ for Example 5.8.

A summary of the computational results is given in Table 11.

TABLE 11. Example 5.8 results summary

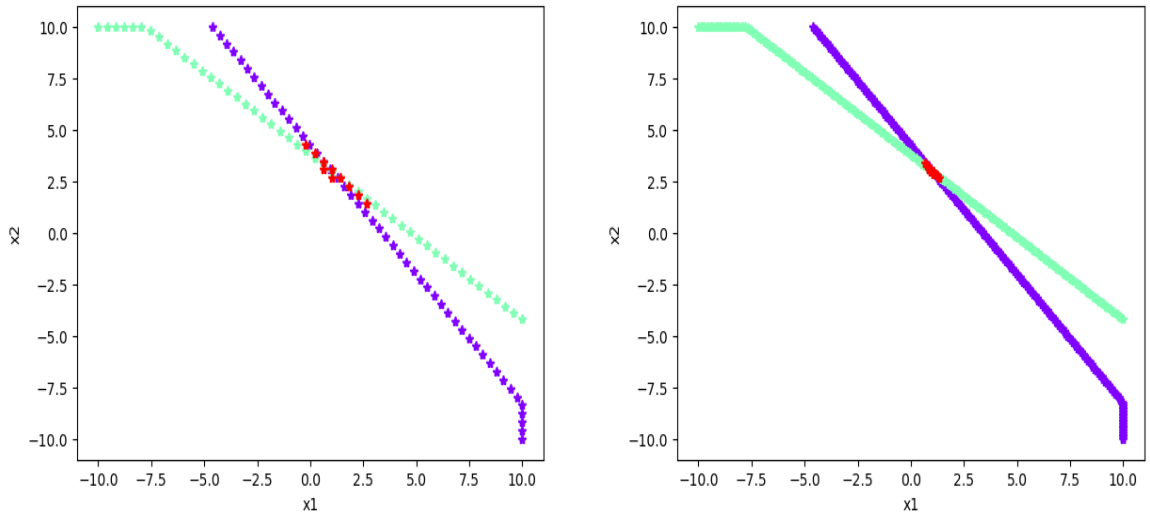
k	m	approximate minimizers	decision space error	image space error
2	50	$\{(0.2041,-0.2041)\ ,\ (0.2041,0.2041)\ ,$ $(-0.2041,0.2041)\ ,\ (-0.2041,-0.2041)\}$	0.2886	0.1249
2	100	$\{(0.101,-0.101)\ ,\ (0.101,0.101)\ ,$ $(-0.101,0.101)\ ,\ (-0.101,-0.101)\}$	0.1428	0.0306
2	200	$\{(0.0503,-0.0503)\ ,\ (0.0503,0.0503)\ ,$ $(-0.0503,0.0503)\ ,\ (-0.0503,-0.0503)\}$	0.0711	0.0076

In Table 11, we observe that both the decision space error and image space error decrease as m increases. In this example, all of the red intersection points have the same objective value (up to 6 decimal places), and therefore are all included in the final approximation.

Example 5.9. Consider the problem of minimizing the Booth function.

$$\begin{aligned} &\min\ f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \\ &\text{s.t.}\ -10 \leq x_i \leq 10. \end{aligned}$$

This function is strictly convex and has a global minimizer at $x^* = (1,3)$. Using various values of m we run the line decomposition algorithm to produce Figure 12.



(A) $m = 50$ (B) $m = 200$

FIGURE 12. Intersection points, $\bigcap_{i=1}^{2} \bigcup_{x \in P} \mathscr{E}(X(d^i, x))$ for Example 5.9.

A summary of the computational results is given in Table 12.

TABLE 12. Example 5.9 results summary

k	m	approximate minimizers	decision space error	image space error
2	50	$\{(1.0204, 3.0612)\}$	0.0645	0.0308
2	100	$\{(1.1111, 2.9293)\}$	0.1317	0.0239
2	200	$\{(1.0553, 2.9648)\}$	0.0655	0.0059

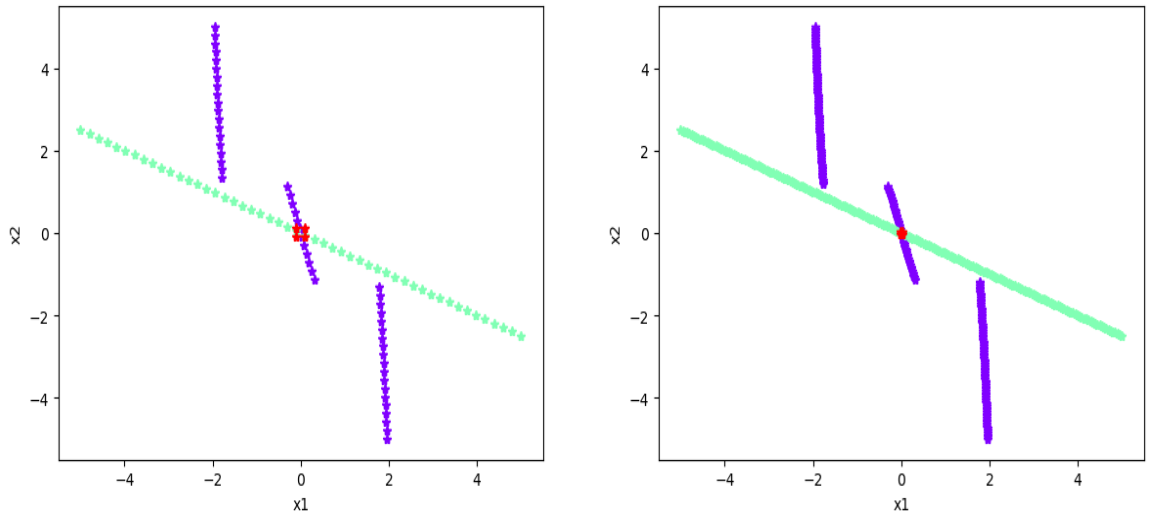
In Table 12, we observe that the decision space error increases when we increase m to be greater than 50. However, the image space error decreases consistently as m increases. This occurs because the final update of E is determined by evaluating the objective value at each intersection point.

Consequently there could be points which are farther from the true minimizer in the decision space, but have smaller objective value in the image space. This objective function's behavior causes an increase in decision space error but a decrease in image space error.

Example 5.10. Consider the problem of minimizing the three-hump camel function.

$$\begin{aligned} \min\ & f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2 \\ \text{s.t.}\ & -5 \leq x_i \leq 5. \end{aligned}$$

The function is nonconvex and has three local minimizers and one global minimizer at $x^* = (0,0)$. Using various values of m we run the line decomposition algorithm to produce Figure 13.



(A) $m = 50$ (B) $m = 200$

FIGURE 13. Intersection points, $\bigcap_{i=1}^{2} \bigcup_{x \in P} \mathscr{E}(X(d^i, x))$ for Example 5.10.

A summary of the computational results is given in Table 13.

TABLE 13. Example 5.10 results summary

k	m	approximate minimizers	decision space error	image space error
2	50	$\{(-0.102, 0.102)\ ,\ (0.102, -0.102)\}$	0.1443	0.0207
2	100	$\{(-0.0505, 0.0505)\ ,\ (0.0505, -0.0505)\}$	0.0714	0.0051
2	200	$\{(-0.0251, 0.0251)\ ,\ (0.0251, -0.0251)\}$	0.0355	0.0013

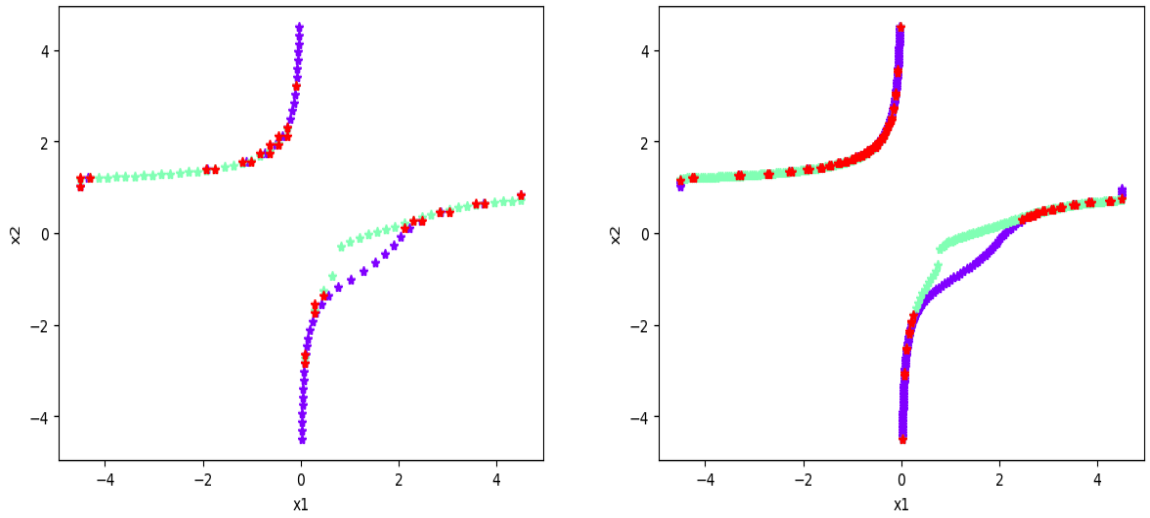
In Table 13, we observe that the decision space and image space error decrease consistently as m increases. Also, we note that the intersection points occur only near the global minimizer, and not at any of the local minimizers.

Lastly, looking at Figure 13 we can conclude that the sets U_1 and U_2 do not always represent lines like in Examples 5.8 and 5.9, rather they can represent piecewise curves. Additionally, we observe that U_1 and U_2 get closer to each other around the local minimizers at $(-1.9509, 0.9753)$ and $(1.9509, -0.9753)$.

Example 5.11. Consider the problem of minimizing the Beale function.

$$\begin{aligned} &\min f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2 \\ &\text{s.t. } -4.5 \leq x_i \leq 4.5 \end{aligned}$$

The Beale function is multimodal, with sharp peaks at the corners of the input domain. The global minimizer occurs at $x^* = (3, 0.5)$. Using various values of m we run the line decomposition algorithm to produce Figure 14.



(A) $m = 50$ (B) $m = 200$

FIGURE 14. Intersection points, $\bigcap_{i=1}^{2} \bigcup_{x \in P} \mathscr{E}(X(d^i, x))$ for Example 5.11.

A summary of the computational results is given in Table 14.

TABLE 14. Example 5.11 results summary

k	m	approximate minimizers	decision space error	image space error
2	50	$\{(2.8469, 0.4592)\}$	0.1584	0.0045
2	100	$\{(2.9545, 0.5)\ ,\ (3.0455, 0.5)\}$	0.0455	0.0033
2	200	$\{(2.9171, 0.4749)\}$	0.0866	0.0015

Observe that, in Figure 14, there are many red intersection points which occur in different regions of the decision space. We see many intersection points since the objective function is very flat in the interior of the domain, and therefore many points in X have similar objective values. Despite this, in Table 14 both measures of error decrease as m increases.

We additionally observe that only one or two intersection points are ever selected to be in the final approximation. Examples 5.8 - 5.10 also use a similar number of points for the final approximation, even though they all have much fewer intersection points to choose from. This demonstrates that the algorithm can recognize small differences between points with very similar objective values.

In summary, we observe that the line decomposition algorithm is able to generate approximate minimizers that are close to the global minimizer in Examples 5.8 - 5.11. Even in the presence of numerous local minimizers, such as in Example 5.10, the algorithm is able to approximate the global minimizer. Another benefit of the line decomposition algorithm is that it provides insight into the behavior of the objective function across the entire domain. This is illustrated through the sets U_1 and U_2, and their intersection points. In particular, the intersection points can imply the existence of local minimizers or small changes in objective values.

5.4. **Implementation details.** Having presented multiple examples of the line decomposition algorithm applied in different settings, we now describe relevant details of the implementation. In particular, we describe the process of computing a set $\mathscr{E}(X(d,x))$ in detail, and the modified intersection method used to update the approximate solution E.

Given a line segment $X(d,x)$, we compute the efficient set by solving multiple single-objective line search problems as stated in the pseudocode. To solve these optimization problems, we use the `minimize_scalar` solver from the `scipy` package. After the line search problems are solved, their solutions are used to determine the feasible step size range for the corresponding efficient line segment. Then the efficient line segment is saved in the list U_i as an `efficientLine` object which we defined in Python. This object class represents a line segment in $\mathbb{R}^n$ and has the following attributes listed in Table 15.

TABLE 15. `efficientLine` object class

Attribute	Type	Description
x	numpy.ndarray	Anchor point for line segment
d	numpy.ndarray	Direction vector for line segment
a_min	float	Smallest feasible step size of line segment
a_max	float	Largest feasible step size of line segment
e1	numpy.ndarray	Left-most endpoint of line segment
e2	numpy.ndarray	Right-most endpoint of line segment
c	color name (string) or code (numpy.ndarray)	Color used to plot the line segment. All line segments with the same direction vector should be assigned the same color.

Once each subproblem is solved for a new line direction, the approximate solution set E must be updated by intersecting it with the new collection of efficient line segments, U_i. We preform this update as follows. For each line segment in the current approximation E, we identify points on the line which intersect U_i. However, we allow a tolerance when checking whether two line segments intersect. Specifically, we extend the length of both line segments by the tolerance amount, and then determine whether they intersect. In $\mathbb{R}^3$, there is the possibility that the line segments are skew lines which nearly intersect at a point. In this case, we want to count the point where the lines nearly intersect. To accomplish this, we solve a least-squares problem to determine the closest points on each line segment, and then check whether the distance between the points is less than the tolerance amount.

Once the intersection points are determined, we must update each line segment in E. For a fixed line segment in E, if there are no intersection points with U_i, then the entire line segment is removed from E. Otherwise, we shrink the line segment so that it contains exactly the points where U_i intersect it. Additionally, if U_i intersects the line segment at points which are far apart, then we break up the line segment into multiple shorter line segments to be added to E. Each of these shorter line segments then contains a cluster of the intersection points.

6. Conclusion

In this paper, we presented an implementation of a line decomposition algorithm that produces an approximate solution to strictly convex MOPs, and convex and nonconvex single-objective problems. Our study has two parts, theoretical and computational. We first outlined the steps of the algorithm and then analyzed the theoretical error accumulated during the main steps of the algorithm, which consists of computing a collection of line search solutions, U_i. Once U_i is computed, the final step in an iteration is to update the approximate efficient set, E, by intersecting it with U_i. However, the computation of this intersection appears to be highly problem specific and therefore we provided experimental values of the final approximation error, $\Delta_H(E, \mathscr{E}(X))$, through various examples.

We applied the line decomposition algorithm to three classes of problems: bi-objective quadratic problems in $\mathbb{R}^2$, biobjective quadratic problems in $\mathbb{R}^3$, and single-objective problems in $\mathbb{R}^2$. We found that the algorithm was able to successfully produce close approximate solutions for biobjective and single-objective problems in $\mathbb{R}^2$. However, the approximate solutions became less accurate when the dimension of the decision space increased to $\mathbb{R}^3$. In higher dimensions, the approximate solutions struggled to get a close fit of the true efficient set, despite increasing the number of iterations. There was a special case we considered in Example 5.6, where we were able to utilize the structure of the objective functions to identify better line directions to use for decomposition in the algorithm. After this modification of selecting line directions, we were able to produce a very close approximation of the efficient set using only a few iterations. Therefore we may conclude that the algorithm has the potential to perform well in higher dimensions, but a more precise method for selecting the line directions must be developed.

While an increase in dimension of the decision space poses challenges for the algorithm, we note that an increase in objective functions is easy to handle. In particular, the number of objective functions only impacts how many single-objective line search problems are performed in an iteration of the algorithm.

We recognize that there exist methods to compute the efficient set of multiobjective convex quadratic problems in $\mathbb{R}^n$ with any number of objectives and to provide a complete parametric description of this set [13, 14]. Despite this fact, for our numerical study we chose quadratic problems for two reasons. Their convexity is easy to establish and the obtained approximations can be compared to the true efficient sets that are easy to compute.

The presented implementation is an initial effort to approximate the efficient set by decomposition and opens several avenues for improvement. In future research, we may investigate an adaptive method for selecting line directions where we first sample a set of random line directions, and analyze their performance to help select new line directions to be used in the next iterations. Alternatively, we may analyze more general MOPs to identify the appropriate line directions as we did in Example 5.6. Another research question we hope to answer is whether a set of perfect line directions, such as these in Example 5.6, exists for every MOP.

Acknowledgments

This work has benefited from the third author's participation in Dagstuhl Seminar 23361 "Multiobjective Optimization on a Budget."

7. APPENDIX

TABLE 16. Randomly generated BOQP problems

BOP	Q^1	Q^2	p^1	p^2	l	u
1	$\begin{bmatrix} 2.8129 & 0.3970 \\ 0.3970 & 0.9756 \end{bmatrix}$	$\begin{bmatrix} 2.5044 & -0.1798 \\ -0.1798 & 0.4136 \end{bmatrix}$	$\begin{bmatrix} -0.0720 \\ 0.2104 \end{bmatrix}$	$\begin{bmatrix} -0.9452 \\ 9.6834 \end{bmatrix}$	-26	2
2	$\begin{bmatrix} 1.8688 & -0.8755 \\ -0.8755 & 0.7934 \end{bmatrix}$	$\begin{bmatrix} 2.3647 & 0.6164 \\ 0.6164 & 0.6733 \end{bmatrix}$	$\begin{bmatrix} 0.5589 \\ 0.9975 \end{bmatrix}$	$\begin{bmatrix} 1.3459 \\ 0.1360 \end{bmatrix}$	-4	1
3	$\begin{bmatrix} 2.0676 & 0.4309 \\ 0.4309 & 0.2559 \end{bmatrix}$	$\begin{bmatrix} 2.2992 & 0.6681 \\ 0.6681 & 0.5988 \end{bmatrix}$	$\begin{bmatrix} -1.2783 \\ -0.5062 \end{bmatrix}$	$\begin{bmatrix} 0.3122 \\ 0.6912 \end{bmatrix}$	-4	3
4	$\begin{bmatrix} 2.5253 & 0.4945 \\ 0.4945 & 0.7845 \end{bmatrix}$	$\begin{bmatrix} 2.2388 & -0.1947 \\ -0.1947 & 0.2764 \end{bmatrix}$	$\begin{bmatrix} -0.2060 \\ -0.8123 \end{bmatrix}$	$\begin{bmatrix} 1.1874 \\ 0.4191 \end{bmatrix}$	-5	3
5	$\begin{bmatrix} 2.4598 & 0.0048 \\ 0.0048 & 0.3469 \end{bmatrix}$	$\begin{bmatrix} 2.6421 & 0.4326 \\ 0.4326 & 0.7927 \end{bmatrix}$	$\begin{bmatrix} -0.1495 \\ -1.6455 \end{bmatrix}$	$\begin{bmatrix} 1.2873 \\ -0.5504 \end{bmatrix}$	-3	6
6	$\begin{bmatrix} 2.5945 & -0.1544 \\ -0.1544 & 0.3888 \end{bmatrix}$	$\begin{bmatrix} 2.0421 & -0.5971 \\ -0.5971 & 0.4286 \end{bmatrix}$	$\begin{bmatrix} 1.3854 \\ -0.2803 \end{bmatrix}$	$\begin{bmatrix} 4.8906 \\ 1.2917 \end{bmatrix}$	-13	2
7	$\begin{bmatrix} 2.4435 & -0.2607 \\ -0.2607 & 0.4392 \end{bmatrix}$	$\begin{bmatrix} 2.1306 & 0.3086 \\ 0.3086 & 0.2282 \end{bmatrix}$	$\begin{bmatrix} 0.5427 \\ 1.1444 \end{bmatrix}$	$\begin{bmatrix} 0.1665 \\ -1.4758 \end{bmatrix}$	-4	9
8	$\begin{bmatrix} 2.3749 & -0.7193 \\ -0.7193 & 0.8697 \end{bmatrix}$	$\begin{bmatrix} 2.3812 & 0.3150 \\ 0.3150 & 0.3892 \end{bmatrix}$	$\begin{bmatrix} -1.1411 \\ -1.0444 \end{bmatrix}$	$\begin{bmatrix} 1.7958 \\ -0.6339 \end{bmatrix}$	-4	4
9	$\begin{bmatrix} 2.6402 & -0.2018 \\ -0.2018 & 0.5425 \end{bmatrix}$	$\begin{bmatrix} 2.5324 & 0.2013 \\ 0.2013 & 0.5697 \end{bmatrix}$	$\begin{bmatrix} 1.5206 \\ -0.3964 \end{bmatrix}$	$\begin{bmatrix} 0.9038 \\ -0.0985 \end{bmatrix}$	-2	1
10	$\begin{bmatrix} 2.6693 & 0.5435 \\ 0.5435 & 0.9891 \end{bmatrix}$	$\begin{bmatrix} 2.7051 & 0.0701 \\ 0.0701 & 0.6549 \end{bmatrix}$	$\begin{bmatrix} -0.1277 \\ 1.4207 \end{bmatrix}$	$\begin{bmatrix} -17.1902 \\ 6.2965 \end{bmatrix}$	-12	7

REFERENCES

[1] A.J. Conejo, E. Castillo, R. Mínguez, R. García-Bertrand, Decomposition Techniques in Mathematical Programming, Springer Berlin, Heidelberg, 2006.

[2] M.M. Wiecek, P.J. de Castro, Decomposition and coordination for many-objective optimization, In: S. Greco, V. Mousseau, J. Stefanowski, C. Zopounidis, (ed.), Intelligent Decision Support Systems: Combining Operations Research and Artificial Intelligence - Essays in Honor of Roman Słowiński, pp. 307–329, Springer, 2022.

[3] D. Li, Y.Y. Haimes, The envelope approach for multiobjeetive optimization problems, IEEE Transactions on Systems, Man, and Cybernetics, 17 (1987), 1026-1038.

[4] M. Gardenghi, T. Gómez, F. Miguel, M.M. Wiecek, Algebra of efficient sets for multiobjective complex systems, J. Optim. Theory Appl. 149 (2011), 385-410.

[5] J.A.C. Mira, F.M. García, On the parametric decomposition theorem in multiobjective optimization, J. Optim. Theory Appl. 174 (2017), 945-953.

[6] A. Raith, K. Dächert, B. Doerr, J.D. Knowles, A. Neumann, F. Neumann, A. Schöbel, M.M. Wiecek, Modeling and decomposition – biobjective block-coordinate descent, In: R. Allmendinger, C.M. Fonseca, S. Sayin, M.M. Wiecek, M. Stiglmayr, (ed.), Multiobjective Optimization on a Budget (Dagstuhl Seminar 23361), vol. 13 of Dagstuhl Reports, pp. 45–55, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

[7] D. Bertsekas, Nonlinear Programming, vol. 4, Athena Scientific, 2016.

[8] E. Soriano, M. Mitkovski, M.M. Wiecek, Decision space decomposition for multiobjective optimization, https://optimization-online.org/?p=29670, 2024. Accessed 23 March 2025.

[9] D.A.G. Vieira, R.H.C. Takahashi, R.R. Saldanha, Multicriteria optimization with a multiobjective golden section line search, Math. Program. 131 (2012), 131-161.
[10] I. Subotić, A. Hauswirth, F. Dörfler, Quantitative sensitivity bounds for nonlinear programming and time-varying optimization, IEEE Trans. Auto. Control, 67 (2021), 2829-2842.
[11] D. Bingham, S. Surjanovic, Virtual library of simulation experiments: Test functions and datasets, https://www.sfu.ca/~ssurjano/optimization.html, 2013. Accessed 22 Jan 2025.
[12] O.B. Augusto, F. Bennis, S. Caro, Multiobjective optimization involving quadratic functions, J. Optim. 2014 (2014), 406092.
[13] P.L.W. Jayasekara, A. Pangia, M.M. Wiecek, On solving parametric multiobjective quadratic programs with parameters in general locations, Ann. Oper. Res. 320 (2023), 123-172.
[14] N. Adelgren, Advancing Parametric Optimization On Multiparametric Linear Complementarity Problems with Parameters in General Locations, Springer Briefs in Optimization, Springer, 2021.