Annals of Operations Research
https://doi.org/10.1007/s10479-020-03820-4

ORIGINAL RESEARCH

Optimizing over the properly efficient set of convex multi-objective optimization problems

Kahina Ghazli[1,2] · **Nicolas Gillis**[3] · **Mustapha Moulaï**[1]

Accepted: 29 September 2020

Abstract
Optimizing over the efficient set of a multi-objective optimization problem is among the difficult problems in global optimization because of its nonconvexity, even in the linear case. In this paper, we consider only properly efficient solutions which are characterized through weighted sum scalarization. We propose a numerical method to tackle this problem when the objective functions and the feasible set of the multi-objective optimization problem are convex. This algorithm penalizes progressively iterates that are not properly efficient and uses a sequence of convex nonlinear subproblems that can be solved efficiently. The proposed algorithm is shown to perform well on a set of standard problems from the literature, as it allows to obtain optimal solutions in all cases.

Keywords Multi-objective programming · Convex optimization · Properly efficient set · Penalty approach

1 Introduction

Multi-objective optimization (also referred to as vector or multi-criteria optimization) deals with the situations where one wishes to minimize several conflicting objectives. Formally, a multi-objective optimization problem (MOP) can be formulated as follows

$$\min_{x} \Big(f_1(x), \ldots, f_p(x) \Big) \quad \text{such that} \quad x \in X, \tag{1}$$

where $p \geq 2$, $x = (x_1, \ldots, x_n)$ is the vector of decision variables, $X \subseteq \mathbb{R}^n$ represent the feasible set and $f_k : \mathbb{R}^n \to \mathbb{R}$ $(k = 1, \ldots, p)$ are the objective functions. A solution of (1) is efficient if it is impossible to improve the value of one objective function without deteriorating at least one of the others, and it is weakly efficient if it is impossible to simultaneously improve all the values of the objective functions. Solving (1) requires to find the set of all efficient

✉ Kahina Ghazli
kghazli@usthb.dz

1 LaROMad, Faculty of Mathematics (USTHB), Algiers, Algeria

2 University of Bejaia, Béjaïa, Algeria

3 Department of Mathematics and Operational Research, Faculté Polytechnique, University of Mons, Mons, Belgium

Published online: 06 October 2020

solutions (Pareto optimal solutions) or the set of all weakly efficient solutions (weak Pareto optimal solutions). The problem (1) has been widely studied in the literature both theoretically and practically; see for example Steuer (1986), Luc (1989), Miettinen (1999), Ehrgott (2005) and Chinchuluun and Pardalos (2007).

In many situations, the decision-making process does not require explicit enumeration of all efficient solutions, but only efficient solutions achieving the optimum of some scalar function expressing the decision maker's preferences within the set of (weak) Pareto optimal solutions. This is the problem of *optimizing over the (weakly) efficient set* introduced by Phillip (1972). This variant of multi-objective optimization avoids numerical difficulties related to generating all efficient solutions; see, e.g., Konno et al. (1997), Benson (1984), Benson (1986) and Benson (1992). For example, one may want to find the extreme values attained by the objective functions over the efficient set, which are of great practical use; see, e.g., Dessouky et al. (1986), Benson (1992). They are obtained by optimizing one of the objective functions of problem (1) over the efficient set. In this context, we can refer to Dessouky et al. (1979), Dessouky et al. (1986), Benson (1984), Isermann and Steuer (1987), Reeves and Reid (1988) and Alves and Costa (2009). Optimizing over the efficient set can also be used in other situations; see for example Benson (1984), Shigeno et al. (2003) and Thach and Thang (2014).

The difficulty of optimizing over the (weakly) efficient set is that it is not explicitly given by a system of inequalities and, in general, it is not convex. It is therefore a difficult non-convex optimization problem, even for the linear case where the objective functions f_k's are linear and the feasible set X is a polyhedron. Most of the proposed methods to tackle this problem focus on the linear case. Several methods were developed following the work of Phillip (1972); see for instance Dessouky et al. (1979), Benson (1984), Benson (1986), Benson (1992), Benson (1993), Ecker and Song (1994), Jorge (2005), and the survey by Yamamoto (2003) and the references therein. Other authors have been interested in optimizing a nonlinear function that is convex or quasi-convex; see for example Dauer (1991), Bolintinéanu (1993), Thach et al. (1996), Benson and Lee (1996), Horst and Thoai (1999), Muu (2000) and Luc (2001).

However, the literature dealing with this problem is relatively poor for nonlinear multi-objective problems due to its difficulty. Most of the algorithms proposed in the literature seek to globally solve this problem through global optimization techniques. In (Yamada et al. 2000) and (Yamada et al. 2001), two inner approximation algorithms were developed to optimize a convex function on the weakly efficient set where the objective functions in (1) are linear and X is a compact convex set. Thoai (2000b) has generalized the conical branch and bound algorithm proposed by Horst and Thoai (1999) to solve more general problems of minimizing a nonlinear function on the efficient set of (1) where the objective functions are nonlinear and X is a closed set. An outer approximation algorithm was proposed by Thoai (2000a) to solve the special case where the underlying multi-objective problem (1) is convex and the function to be optimized is a nondecreasing composite quasi-concave function of the criterion functions of (1). In (Tuyen and Muu 2001), an algorithm based on the branch and bound approach was proposed to minimize a convex function on the weakly efficient set of problem (1) where X is a compact convex set, and the objective functions, in (1), are affine fractional. Tuy and Hoai-Phuong (2006) proposed a new approach to solve more general problems, which consists in reformulating the optimization problem over the (weakly) efficient set as a non-convex optimization problem under composite monotonic constraints, which were then solved using monotonic optimization methods. An algorithm based on the branch and bound approach was proposed by Benson (2012) to minimize a convex function on the weakly efficient set of a convex nonlinear multi-objective problem. Recently, based on primal and dual variants of Benson's algorithm (Hamel et al. 2014), Liu

and Ehrgott (2018) proposed both primal and dual algorithms for optimizing a linear function over the efficient set of a convex multi-objective optimization problem. It is worth noting that most of these algorithms based on the techniques of global optimization are only able to solve moderate sized problems.

In addition to these algorithms, some authors have been interested in finding approximate solutions using numerical techniques. To the best of our knowledge, this has been considered for the first time by White (1996). The author discussed the theoretical properties of different formulations based on penalty functions to solve the linear case. In (Bolintinéanu and El Maghri 1997), an exact penalty algorithm was proposed to minimize a concave function on the weakly efficient set of a linear multi-objective problem. However, no numerical studies were reported. Recently, some works have focused on solving the semivectorial bilevel programming problem which is a generalization of the optimization over the efficient set; see Bonnel and Morgan (2006), Ankhili and Mansouri (2009), Zheng and Wan (2011), Calvete and Galé (2011) and Ren and Wang (2016). Only few algorithms have been developed and it is worth noting that most of them are based on a penalty approach.

In this paper, we consider the problem of minimizing a convex function over the efficient set of a convex nonlinear multi-objective optimization problem in which $f_k, k = 1, \ldots, p$, are convex quadratic functions and X is a compact convex set. Although we focus on quadratic convex objective functions, our approach can be generalized to smooth convex objective functions in a straightforward way. Problems of this type have several applications, including finance models such as the portfolio optimization problem (Konno and Inori 1989; Thach et al. 1996; Ogryczak 2000; Steuer et al. 2007.

In this paper, a new algorithm based on the Karush-Kuhn-Tucker optimality conditions of the weighted problem of (1) is proposed that is easy to implement. This method is inspired by the penalty approach for nonlinear optimization; see for instance Nocedal and Wright (1999). Although it does not necessarily find an optimal solution, it is guaranteed to provide an efficient solution. In all the tested cases from the literature, it was able to compute an optimal solution. Notice that our main focus is only on efficient solutions obtained with positive weight vectors, that is, the *properly efficient solutions*. The concept of proper efficiency was introduced firstly by Kuhn and Tucker (1951) and later redefined by Geoffrion (1968), in order to eliminate efficient solutions which display some undesirable properties; see Kuhn and Tucker (1951).

The paper is organized as follows. Section 2 describes the optimization problem over the efficient set and presents some basic definitions related to multi-objective optimization. In Sect. 3, two reformulations of the problem are given. The algorithm and its implementation are described in Sect. 4. In Sect. 5, the proposed algorithm is first tested on different small-scale cases from the literature (up to 20 variables and 3 objectives), and a sensitivity analysis of its parameters is performed. It is also tested on medium-scale problems (up to 150 variables and 3 objectives) motivated by the computation of nadir points. Some concluding remarks and some perspectives are given in the last section.

2 Preliminaries

In this paper, we consider (1) where the f_k's are convex quadratic functions

$$f_k(x) = \frac{1}{2}\, x^T Q_k x + c_k^T x, \;\; k = 1, \ldots, p$$

with $Q_k \in \mathbb{R}^{n\times n}$ is a symmetric, positive semidefinite matrix (which we denote $Q_k \succeq 0$) and $c_k \in \mathbb{R}^n$. As mentioned above, our approach will be easily generalized for smooth convex functions f_k's. We focus on the quadratic case for simplicity. Hence we focus on the following convex multi-objective optimization problem (CMOP)

$$\begin{aligned} \min_x \quad & \big(f_1(x), \ldots, f_p(x)\big) \\ \text{such that } & Ax = b, x \geq 0, G(x) \leq 0, \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$, with $m \leq n$ and $\text{rank}(A) = m$, and $b \in \mathbb{R}^m$. We will denote a_i the ith row of A so that the ith linear constraint is given by $a_i^T x - b_i = 0$. The function $G : \mathbb{R}^n \to \mathbb{R}^c$ is such that $G(x) = (g_1(x), \ldots g_c(x))$, and $G(x) \leq 0$ means that $g_\ell(x) \leq 0$ for each $\ell \in \{1, \ldots, c\}$. The feasible set is given

$$X = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0,\ G(x) \leq 0\},$$

and we will assume that

1. X is a compact and nonempty set,
2. for each $\ell = 1, \ldots, c$, the constraint function g_ℓ is continuously differentiable and convex, which implies that X is a convex set,
3. the gradients of the active constraints are linearly independent at any feasible point.

Unlike the case with a single objective, the optimal solution for multi-objective optimization is in general not a single point but a set of solutions where each solution represents a compromise between the different objectives. To solve this type of problems, one has to find all efficient solutions or weakly efficient solutions in the sense of the following definitions (Ehrgott 2005).

Definition 1 A feasible point $\hat{x} \in X$ is an efficient (Pareto optimal, nondominated) solution for problem (2) if and only if there does not exist $x \in X$ such that $f(x) \leq f(\hat{x})$ and $f(x) \neq f(\hat{x})$.

Definition 2 A feasible point $\hat{x} \in X$ is a weakly efficient (weakly Pareto optimal, weakly nondominated) solution for problem (2) if and only if there does not exists $x \in X$ such that $f(x) < f(\hat{x})$.

Some efficient solutions are well known for having some undesirable features such as allowing unbounded trade-offs between objectives (Kuhn and Tucker 1951; Geoffrion 1968). In order to avoid such solutions, a slightly restricted definition of efficiency was introduced, namely, the notion of *proper efficiency*. In this paper, we use the definition of Geoffrion.

Definition 3 (Geoffrion 1968) A feasible point $\hat{x} \in X$ is a properly efficient solution if it is efficient and if there exists some real number $M > 0$ such that, for each k and $x \in X$ satisfying $f_k(x) < f_k(\hat{x})$, there exists at least one f_j such that $f_j(\hat{x}) < f_j(x)$ and

$$\frac{f_k(\hat{x}) - f_k(x)}{f_j(x) - f_j(\hat{x})} \leq M.$$

In other words, properly efficient solutions are efficient solutions for which, given any objective, the trade-off between that objective and some other objective is bounded from above. An efficient point that is not properly efficient is said to be improperly efficient.

Let us denote X_{pE}, X_E and X_{wE} the set of properly efficient solutions, the set of all efficient solutions, and the set of all weakly efficient solutions of problem (2), respectively,

so that $X_{pE} \subseteq X_E \subseteq X_{wE}$. It is well known that in general the set X_{wE} is closed but X_E and X_{pE} are not necessarily closed. In the linear case the two sets X_{pE} and X_E are equal and closed. Note that under the convexity assumption, the sets X_{pE}, X_E and X_{wE} are connected. We can now define the problem which will be the focus in this paper, namely optimizing a convex function over the efficient set of problem (2):

$$\inf_x \ \phi(x) \quad \text{such that} \quad x \in X_E \subseteq X, \tag{3}$$

where $\phi : \mathbb{R}^n \to \mathbb{R}$ is convex and continuous function on X. We will focus on properly efficient solutions, and hence focus on the problem

$$\inf_x \ \phi(x) \quad \text{such that} \quad x \in X_{pE} \subseteq X, \tag{4}$$

Problems (3) and (4) are difficult optimization problem, even in the linear case where the objective function $\phi(x)$ is linear and (2) is a linear multi-objective optimization problem, because the feasible sets X_E and X_{pE} are not convex in general. The motivation for considering only properly efficient solutions is two-fold. First, this set of solutions is technically easier to deal with. Second, these solutions are usually more useful to the decision maker, as having bounded trade-offs between the objectives. The characterization of these solutions is established via positive combinations of the objective functions; see Theorem 1 below.

3 Reformulations of (4)

In this section, we give two reformulations of problem (4). First, we transform (4) into a nonlinear bilevel problem. Then, we transform this second problem into a single level problem using the Karush–Kuhn–Tucker (KKT) optimality conditions of the lower level problem. Both formulations will be useful in the design of our algorithm described in Sect. 4.

It is well known that any properly efficient solution of a convex multi-objective optimization problem can be obtained by varying the positive coefficients; see for example Geoffrion (1968), Miettinen (1999), Ehrgott (2005). Let us denote the set of possible weights as

$$\Lambda = \{\lambda \in \mathbb{R}^p \mid \lambda > 0 \text{ and } \sum_{k=1}^{p} \lambda_k = 1\}.$$

Given $\lambda \in \Lambda$, the weighted problem associated with (2) is defined by

$$\min_{x \in X} \ f_\lambda(x) \quad \text{with } f_\lambda(x) = \sum_{k=1}^{p} \lambda_k f_k(x), \tag{5}$$

where the λ_k's weight the different objective functions. In the remainder of the paper, we denote $Q_\lambda = \sum_{k=1}^{p} \lambda_k Q_k$ and $c_\lambda = \sum_{k=1}^{p} \lambda_k c_k$ so that $f_\lambda(x) = \frac{1}{2} x^T Q_\lambda x + c_\lambda^T x$. Note that (5) is a convex optimization problem since X is a convex set and the objective function is also convex since $Q_k \succeq 0$ for each k implies $Q_\lambda \succeq 0$.

Theorem 1 (Geoffrion 1968) *A feasible solution $\bar{x}$ is properly efficient for problem (2) if and only if there exists a weight vector $\lambda \in \Lambda$ such that $\bar{x}$ is an optimal solution of the weighted problem (5).*

Let us illustrate this result with a simple example.

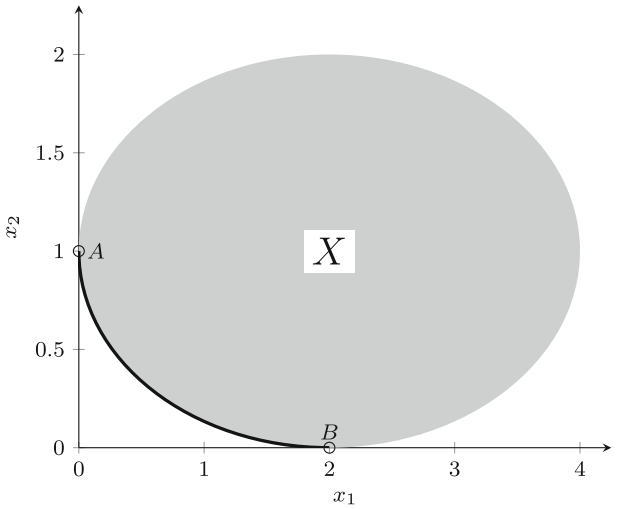

Fig. 1 The set X_{pE} of the CMOP from Example 1: X_{pE} does not contain the points A and B

Example 1 Let $f_1(x_1, x_2) = x_1$, $f_2(x_1, x_2) = x_2$ and

$$X = \left\{ (x_1, x_2) \in \mathbb{R}^2 \mid \frac{1}{4}(x_1 - 2)^2 + (x_2 - 1)^2 \leq 1 \right\}.$$

The efficient set in the decision space is the bold curve between A and B shown in Fig. 1. The two boundary points A and B are efficient, and correspond to the unique optimal solutions of (5) for $\lambda = (1, 0)$ and $\lambda = (0, 1)$, respectively. However, they are not properly efficient since there is no $\lambda > 0$ such that they are optimal solutions of (5); see Fig. 1.

Notice that if λ is allowed to have zero entries, any optimal solution of (5) may be weakly efficient but not necessarily efficient unless it is the unique optimal solution (this is the case for example for the boundary points A and B of X_E in Example 1). However, the use of $\lambda > 0$ in (5) leads to the following difficulty : since $\{\lambda \in \mathbb{R}^p | \lambda > 0\}$ is an open set, the minimum of $\phi(x)$ over all optimal solutions of (5) for all $\lambda > 0$ is not necessarily attained. So, to overcome this difficulty, the condition $\lambda_k \geq \epsilon$ will be used instead of $\lambda_k > 0$, for all $k = 1, \ldots, p$, where ϵ is a small positive number. Since (2) is convex, as will be clarified below, for ϵ sufficiently small, all the properly efficient solutions can be found by varying λ over Λ_ϵ (see Miettinen (1999)),

$$\Lambda_\epsilon = \left\{ \lambda \in \mathbb{R}^p \mid \sum_{k=1}^{p} \lambda_k = 1, \ \lambda_k \geq \epsilon \text{ for } k = 1, \ldots, p \right\}.$$

However, before we proceed further, let us briefly discuss the choice of the value of ϵ in the decision making process.

Relationship between ϵ and trade-offs As we mentioned above the value of ϵ must be sufficiently small to avoid missing some of the properly efficient solutions. However, in practice, ϵ should be specified in advance which is a difficult problem, as pointed out by Miettinen (1999) and Korhonen and Wallenius (1989). To alleviate this issue, some authors (see, e.g., Geromel and Ferreira (1991), Miettinen (1999) and Karimi and Karimi (2017)) showed that there is a relationship between the concept of trade-off and the ratios between the weight

coefficients that is, $\frac{f_k(\hat{x}) - f_k(x)}{f_j(x) - f_j(\hat{x})} \leq \frac{\lambda_j}{\lambda_k}$ (for all $k, j, k \neq j$) for every properly efficient solution $\hat{x}$, where λ the corresponding positive weighted vector. In particular, if we take the limit of the ratio as $\lambda_k \to 0$, the trade-off between f_k and f_j becomes unbounded, while imposing $\lambda_k \geq \epsilon$ implies that $1/\epsilon$ is an upper bound for all ratios λ_j/λ_k leading to an upper bound for all trade-offs between objectives. Geromel and Ferreira (1991) considered ϵ as a constant vector ($\epsilon \in \mathbb{R}^p$) where its components are computed such as to satisfy a given prescribed marginal rates of substitution.

They then proposed, in the convex case, a simple technique to compute a theoretical upper bound on the set of properly efficient solutions obtained for all $\lambda \geq \epsilon$. In this paper, the algorithm we propose tries to find an optimal solution on all properly efficient solutions generated for ϵ small enough (we will use $\epsilon = 10^{-4}$ in our experiments) getting the properly efficient set as larger as possible (As will be shown below) while avoiding improperly efficient solutions. However, since ϵ is considered here as an input parameter (see Sect. 5), the algorithm can also be used to solve the cases where a common upper bound on all trade-offs between objectives or maximal rates of substitution are provided by the decision maker.

Based on the interesting aspect that $1/\epsilon$ is an appropriate upper bound on all trade-offs between objectives, we observe that any properly efficient solution obtained with $\lambda \geq \epsilon e$ (where $e \in \mathbb{R}^p$ is the vector of all ones) may be defined as ϵ-*properly efficient solution* in the sense of Wierzbicki; see Wierzbicki (1980); Miettinen (1999).

Definition 4 (Wierzbicki 1980) A feasible point $\hat{x} \in X$ is $\epsilon-$properly efficient if

$$\left(f(\hat{x}) - \mathbb{R}^p_\epsilon \setminus \{0\}\right) \cap f(X) = \emptyset,$$

where $f(X) = \{f(x) \in \mathbb{R}^p \mid x \in X\}$ and $\mathbb{R}^p_\epsilon = \{y \in \mathbb{R}^p \mid \text{dist}(y, \mathbb{R}^p_+) \leq \epsilon \|y\|\}$ where dist denotes the Euclidean distance.

Let us consider the set

$$X_{\epsilon pE} = \left\{x \in X \mid \exists \lambda \in \Lambda_\epsilon \text{ such that } \ f_\lambda(x) \leq f_\lambda(y) \ \forall\, y \in X\right\} \tag{6}$$

$$= \left\{x \in X \mid x \in \arg\min_{x \in X} f_\lambda(x), \ \lambda \in \Lambda_\epsilon\right\}, \tag{7}$$

which is, for any given $\epsilon > 0$, a nonempty, compact and connected subset of the properly efficient set that contains all the properly efficient solutions generated with weight vectors in Λ_ϵ.

In the next paragraph, we will show how the value of ϵ can affect the properly efficient subset $X_{\epsilon pE}$ generation by solving problem (5) for all $\lambda \in \Lambda_\epsilon$.

Let $\{\epsilon^{(i)}\}$ be a sequence of parameters such that $\epsilon^{(i)} > \epsilon^{(i+1)}$ for all $i = 1, 2, \ldots$ By solving for each i the weighted problem (5) for λ in $\Lambda_{\epsilon^{(i)}}$, we obtain a family of subsets of properly efficient solutions $X_{\epsilon^{(1)}pE}, X_{\epsilon^{(2)}pE}, \ldots$ possessing the following property.

Lemma 1 *The sequence of subsets $\{X_{\epsilon^{(i)}pE}\}$ corresponding to a decreasing sequence of parameters $\{\epsilon^{(i)}\}$ is nested, i.e., $X_{\epsilon^{(i)}pE} \subseteq X_{\epsilon^{(i+1)}pE}$ for all i.*

Proof This follows from the fact that $\Lambda_{\epsilon^{(i)}} \subseteq \Lambda_{\epsilon^{(i+1)}}$ since $\epsilon^{(i)} > \epsilon^{(i+1)}$ for all i. □

Lemma 1 is illustrated in Fig. 2, where some nested properly efficient subsets of the properly efficient set of the CMOP in Example 1 corresponding to certain ϵ values are shown. For $\epsilon = 0.4, 0.2, 0.1$ and 0.01, we get the curves FG, EH, DI and CJ, respectively, which are subsets of the bold curve connecting A and B. For $\epsilon = 0$, we get $X_{\epsilon pE} = X_{wE}$.

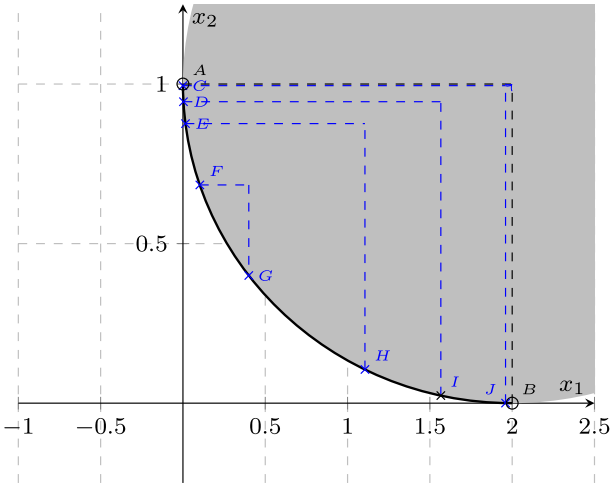

Fig. 2 Nested properly efficient subsets for some different values of ϵ

We observe that the properly efficient solutions with large ϵ values are located in the central part, and as ϵ decreases, the set $X_{\epsilon pE}$ becomes larger. As ϵ gets smaller, we obtain a better approximation of the properly efficient set of (2).

In this paper, we will focus on the problem

$$\min_{x} \ \phi(x) \quad \text{such that} \quad x \in X_{\epsilon pE} \subseteq X, \tag{8}$$

which is a parametric optimization problem which admits an optimal solution for every parameter $\epsilon > 0$, which follows by Weierstrass theorem (the objective function ϕ is continuous and $X_{\epsilon pE}$ is a compact set).

3.1 Connection between Problems (4) and (8)

In this section, we discuss the relationship between the choice of the parameter ϵ and the minimum (or infimum) of ϕ on all properly efficient solutions. Let us consider the original problem (4).

Assumption 1 Problem (4) admits an optimal solution $x^\star \in X_{pE}$.

This is obvious when (2) is a linear multi-objective problem, since X_{pE} is a closed set. In the general case, Assumption 1 guarantees that $x^\star$ is not a boundary point, but $x^\star$ does not necessarily belong to $X_{\epsilon pE}$.

Hence, since $X_{\epsilon pE} \subseteq X_{pE}$, we have

$$\phi(\epsilon) \geq \phi^\star, \tag{9}$$

where $\phi(\epsilon)$ and $\phi^\star$ denote the optimal values of (8) and (4), respectively. We also have $\phi(\epsilon^{(i)}) \geq \phi(\epsilon^{(i+1)})$ for all $\epsilon^{(i)} > \epsilon^{(i+1)}$ since $X_{\epsilon^{(i)}pE} \subseteq X_{\epsilon^{(i+1)}pE}$ (Lemma 1). The equality in (9) holds if the optimal solution $x^\star$ belongs to $X_{\epsilon pE}$. Since ϵ is chosen small in practice, this happens when the weight vector $\lambda^\star$ corresponding to the optimal solution $x^\star$ is sufficiently far from the extreme points of Λ (note that $\lambda^\star$ is not necessarily unique).

The connection between the original problem (4) and the problem (8) is established formally in the following property.

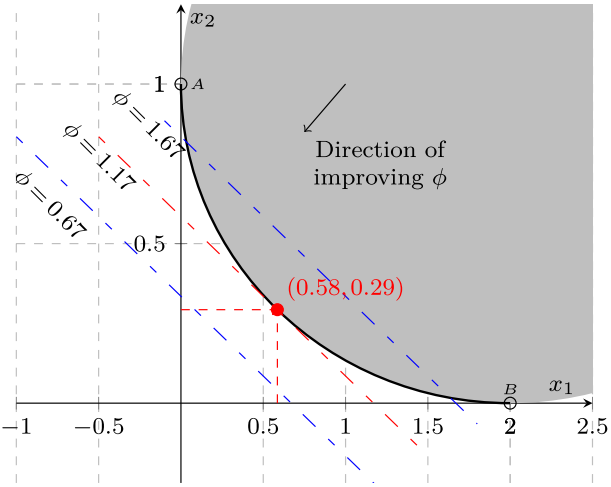

Fig. 3 Illustration of an optimal solution of (4) obtained with some ϵ; see Example 2

Property 1 Under Assumption 1, for any optimal solution of (4) there exists a threshold value $\epsilon^\star > 0$ such that

$$\phi^\star = \phi(\epsilon) \quad \text{for all } \epsilon \in]0, \epsilon^\star].$$

Proof By Assumption 1, there exists a threshold value $\epsilon^\star > 0$ such that $\phi^\star = \phi(\epsilon^\star)$. The result then follows from Lemma 1, since $\phi^\star \leq \phi(\epsilon) \leq \phi(\epsilon^\star)$ for any $\epsilon < \epsilon^\star$. □

The property stated above guarantees that the original problem (4) can actually be solved by means of problem (8), given that Assumption 1 is satisfied and the parameter ϵ is sufficiently small.

This is illustrated in the following example.

Example 2 Let us consider the properly efficient set given in Example 1, and let

$$\phi(x_1, x_2) = x_1 + 2x_2.$$

For $\epsilon = 0.2$, the set $X_{\epsilon pE}$ with $\epsilon = 0.2$ is shown in Fig 2 by the curve EH, joining the points E and H on the bold curve. The optimal value of (8) is given by $\phi(\epsilon) = 1.17$, and is attained at the properly efficient point $x^\star = (0.58, 0.29)$ with the corresponding weight vector $\lambda^\star = (0.33, 0.66)$. This optimal solution remains unchanged for smaller values of ϵ. In this example, the optimal value $\phi(\epsilon)$ is not sensitive to the value of ϵ as $\phi(\epsilon) = \phi^\star = 1.17$ for all $0 < \epsilon \leq 0.33$. This is also the optimal value of ϕ on the (weakly) efficient set, i.e., for $\epsilon = 0$; see Fig. 3 for an illustration.

However, as indicated above, the choice of ϵ is problem dependent, and difficult to choose in advance. Hence it may happen that ϵ is not sufficiently small, excluding the weights $\lambda^\star$ corresponding to the optimal solution $x^\star$ of problem (4), which leads to a strict inequality in (9). Such a situation arises in general when $\lambda^\star$ is close to the extreme points of Λ. However, under Assumption 1, there must exist an ϵ for which $x^\star$ is containing in $X_{\epsilon pE}$. Hence, based on Lemma 1, a simple way to avoid this situation, and thus finding the optimal solution on the entire properly efficient set X_{pE}, is to minimize the function ϕ on a sequence of expanding subsets $X_{\epsilon^{(1)} pE} \subseteq X_{\epsilon^{(2)} pE} \subseteq \cdots$ generated with $\epsilon^{(1)} > \epsilon^{(2)} > \cdots$, approximating the set

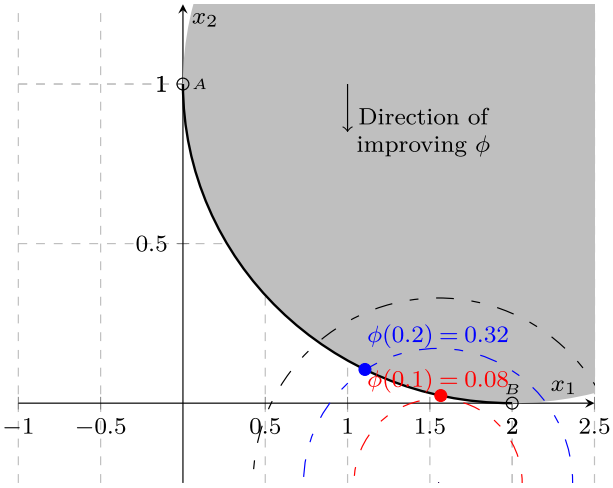

Fig. 4 Illustration of a finite sequence of properly efficient points obtained as ϵ decreases; see Example 3

Table 1 Values of $\phi(\epsilon)$ for the problem of Example 1 and $\phi(x_1, x_2) = -x_1 + x_2$

ϵ	0.2	0.1	10^{-2}	10^{-3}
$\phi(\epsilon)$	-1	-1.54	-1.96	-1.99

X_{pE} from the inside such that $\phi(\epsilon^{(i)}) \to \phi^\star$. If the optimal value of (8) remains unchanged, then we stop the process. Note that this process is finite since there must exist an $\epsilon > 0$ such that $\phi(\epsilon) = \phi^\star$ under Assumption 1, and in practice it convergences very fast. Let us illustrate this on a simple example.

Example 3 Let us consider the properly efficient set given in Example 1, and let

$$\phi(x_1, x_2) = (x_1 - 1.55)^2 + (x_2 + 0.25)^2.$$

For $\epsilon = 0.2$, $\phi(\epsilon) = 0.32$ is the optimal value of (8), but not on the entire properly efficient set. Indeed, setting $\epsilon = 0.1$, we get a better solution $x^\star = (1.57, 0.02)$ (shown in red in Fig. 4) with $\lambda^\star = (0.1, 0.9)$. The value $\phi(\epsilon) = 0.08$ remains unchanged for all $0 < \epsilon \leq 0.1$ (we get the same solution even if $\epsilon = 0$). This is illustrated in Fig. 4.

It is worth noting that if Assumption 1 does not hold, i.e., the optimal value of (4) is not attained at any properly efficient point but rather at an improperly efficient point, then by performing the process described above, the formulation (8) may converge to the closest properly efficient solution to this point, as shown in Example 4 below.

Example 4 Let us consider the properly efficient set given in Example 1, and let

$$\phi(x_1, x_2) = -x_1 + x_2.$$

The optimal value of (4) is $\phi^\star = -2$, attained by the (boundary) efficient point $(2, 0) \notin X_{pE}$. Table 1 gives the optimal value $\phi(\epsilon)$ obtained for $\epsilon = 0.2, 0.1, 10^{-2}$ and 10^{-3}. The sequence $\{\phi(\epsilon^{(i)})\}$ converges to $\phi^\star = -2$ as $\epsilon^{(i)} \to 0$. An illustration of this case is shown in Fig. 5.

Note that all optimal values given in the above examples were computed using our Algorithm 1 which will be presented in the next section. Note also that the results obtained in our

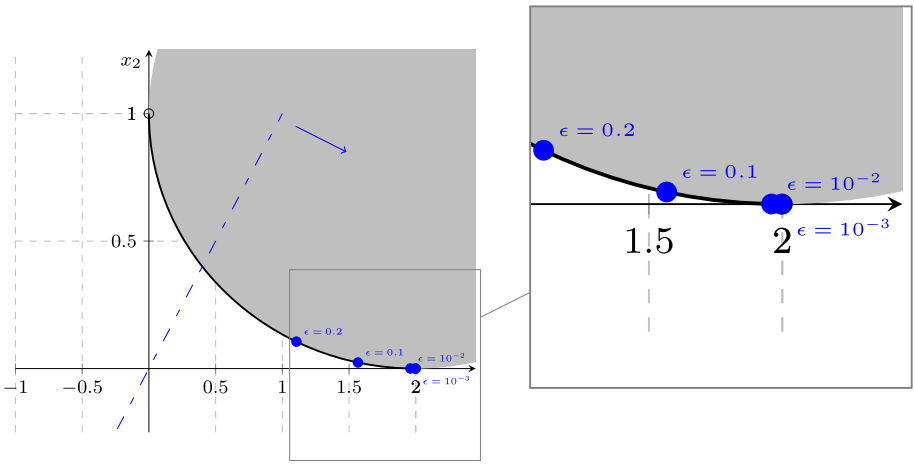

Fig. 5 Illustration of an improperly efficient point, and the sequence of properly efficient points obtained as ϵ decreases; see Example 4

experiments in Sect. 5 use the value $\epsilon = 10^{-4}$ in all cases. As we will see, we always obtain the same results as obtained in the literature with other methods. We did not need to perform the process described above (that is, decreasing ϵ progressively). However, Algorithm 1 could be easily adapted to tune ϵ which is, for now, a fixed parameter.

In the sequel, we will assume that Assumption 1 holds and hence Property 1 applies so that there exists a threshold value $\epsilon^\star$ for the parameter ϵ, and hence the two problems (4) and (8) are equivalent for any ϵ smaller than $\epsilon^\star$. Therefore, instead of problem (4), we focus on the problem (8).

Problem (8) can be tackled via the following nonlinear bilevel programming problem (NBLP):

$$\begin{array}{ll} \min\limits_{\lambda \in \Lambda_\epsilon} & \phi(x) \\ \text{such that } x \in \arg\min_{x \in X} f_\lambda(x), & \end{array} \tag{10}$$

where λ represents the variables of the first level (upper level variables) and x represents the variables of the second level (lower level variables).

Since (5) is convex and under the linear independence constraint qualification, every solution of problem (8) is a solution of the Karush-Kuhn-Tucker (KKT) optimality conditions of (5).

Proposition 1 *A necessary and sufficient condition for $x^* \in X$ to be a solution of problem (5) is that there exists a vector $(\lambda^*, z^*, s^*, v^*) \in \Lambda_\epsilon \times \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^c_+$ such that*

$$\begin{array}{l} Q_{\lambda^*} x^* + c_{\lambda^*} + A^T z^* - s^* + \nabla G(x^*)^T v^* = 0, \\ (s^*)^T x^* = 0, \\ (v^*)^T G(x^*) = 0, \\ \lambda^* \in \Lambda_\epsilon,\ s^* \geq 0,\ v^* \geq 0,\ z^* \in \mathbb{R}^m, \end{array} \tag{11}$$

where ∇G is the Jacobian matrix of G. The vector $(\lambda^*, z^*, s^*, v^*) \in \Lambda_\epsilon \times \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^c_+$ is not necessarily unique: to each efficient point, there corresponds a subdomain of Λ_ϵ, but for each λ in this domain, the Lagrange multipliers (z, s, v) are unique.

The previous result can be used to check whether a given feasible point $x \in X$ is in the set $X_{\epsilon pE}$, either by resolving the linear system (11) in (λ, z, s, v), or equivalently by solving

$$\begin{aligned} \min_{\lambda,z,s,v} \quad & L(\lambda, z, s, v) = \left\| Q_\lambda x^* + c_\lambda + A^T z - s + \nabla G(x^*)^T v \right\|_2^2 \\ \text{such that } & s^T x^* = 0, \\ & v^T G(x^*) = 0, \\ & \lambda \in \Lambda_\epsilon,\ s \geq 0,\ v \geq 0,\ z \in \mathbb{R}^m, \end{aligned} \tag{12}$$

and checking whether the optimal value is equal to zero. The problem (12) is a linearly constrained quadratic programming problem (LCQP), and can be used as an efficiency test.

Property 2 The problem (12) has an optimal solution.

Proof Notice that L is a convex quadratic function bounded below on the nonempty polyhedron $\Lambda_\epsilon \times \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^c_+$ since $L(\lambda, z, s, v) \geq 0$ for any $(\lambda, z, s, v) \in \Lambda_\epsilon \times \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^c_+$. Therefore, the fundamental theorem of Frank and Wolfe implies that the optimal solution exists; see for example Belousov and Klatte (2002). □

The characterization of the properly efficient set by the KKT approach leads to the following equivalent reformulation of problem (8).

Proposition 2 *A point x^* is an optimal solution to (8) if and only if there exists a vector $(\lambda^*, z^*, s^*, v^*)$ such that $(x^*, \lambda^*, z^*, s^*, v^*)$ is an optimal solution to the following problem*

$$\begin{aligned} \min_{x,\lambda,z,s,v} \quad & \phi(x) \\ \textit{such that } & Q_\lambda x + c_\lambda + A^T z - s + \nabla G(x)^T v = 0, \\ & s^T x = 0, \\ & v^T G(x) = 0, \\ & x \in X,\ \lambda \in \Lambda_\epsilon,\ s \geq 0,\ v \geq 0,\ z \in \mathbb{R}^m. \end{aligned} \tag{13}$$

This problem is a nonconvex optimization problem which is difficult to solve due to the nonlinear constraints $Q_\lambda x + c_\lambda + A^T z - s + \nabla G(x)^T v = 0$, $s^T x = 0$ and $v^T G(x) = 0$. Indeed, to solve such problems, several methods were proposed in the literature using penalty approaches; for example to solve semivectorial bilevel problems in which the the multiobjective optimization problem is linear (Ankhili and Mansouri 2009; Zheng and Wan 2011; Ren and Wang 2016). In general, these algorithms use a classical penalty function that consists to append the bilinear equality constraint (i.e., the complementary constraint $s^T x^* = 0$ in the linear case) in the objective function weighted by a positive penalty parameter. In the nonlinear case, there are more nonlinear constraints (namely, $Q_\lambda x + c_\lambda + A^T z - s + \nabla G(x)^T v = 0$ and $v^T G(x) = 0$), which makes penalty approaches rather numerically unstable. In fact, we were not able to obtain a stable method able to solve all small-scale problems presented in Sect. 5 with such an approach, which also was rather sensitive to the parameters. Also, we tried to solve (13) using a block coordinate descent method, optimizing alternatively over the variable x and the block of variables (λ, z, s, v) for which the subproblems are convex. However, although it worked well in some cases, this approach was not be able to generate good solutions in all scenarios.

4 Proposed method

Our idea for tackling (8) is to construct a penalty function that is a combination of the objective functions of the upper and the lower levels of the equivalent bilevel reformulation (10). The

problem we consider is the following

$$\min_{x \in X} \ \frac{1}{\gamma}\phi(x) + f_\lambda(x) \tag{14}$$

where $\lambda \in \Lambda_\epsilon$ and $\gamma > 0$ is a penalty parameter. Note that for $\gamma \to +\infty$, solutions x of (14) tend to properly efficient solutions; see Theorem 2 below.

Theorem 2 *Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function. For any given $\lambda \in \Lambda_\epsilon$ and for some $\gamma > 0$, let x_γ be an optimal solution of (14). As $\gamma \to +\infty$ and λ is fixed, any optimal solution x_γ of (14) tends to a properly efficient solution of (2).*

Proof Let x_γ be an optimal solution of (14). Using the KKT conditions, there exists (z, s, v) such that $s^T x_\gamma = 0$, $v^T G(x_\gamma) = 0$, $s \geq 0$, $v \geq 0$ and

$$\frac{1}{\gamma}\nabla\phi(x_\gamma) + \nabla f_\lambda(x_\gamma) + A^T z - s + \nabla G(x_\gamma)^T v = 0,$$

where $\nabla f_\lambda(x_\gamma) = Q_\lambda x_\gamma + c_\lambda$. It follows that

$$\frac{1}{\gamma}\nabla\phi(x_\gamma) + Q_\lambda x_\gamma + c_\lambda + A^T z - s + \nabla G(x_\gamma)^T v = 0,$$

which implies that

$$\frac{1}{\gamma}\left\|\nabla\phi(x_\gamma)\right\|_2 = \left\|Q_\lambda(x_\gamma) + c_\lambda + A^T z - s + \nabla G(x_\gamma)^T v\right\|_2.$$

Taking the limit as $\gamma \to +\infty$, we find that $\frac{1}{\gamma}\nabla\phi(x_\gamma) \to 0$ since $\nabla\phi(x_\gamma)$ is bounded, which concludes the proof as (λ, z, s, v) tends to be a solution of (11). □

For $\lambda \in \Lambda_\epsilon$ and γ fixed, (14) is a convex nonlinear problem and can be solved effectively. For x fixed, it does not make much sense to optimize over λ since x heavily depends on the value of λ. Moreover, since $f_\lambda(x)$ is a linear function in λ and Λ_ϵ is a simplex, only one entry of λ will generically be different from Λ_ϵ. Our idea is to update λ by solving (12). This means that, given x, we will compute λ in order to satisfy the efficiency test for x as best as possible. As we will see, it turns out that this new alternating strategy works remarkably well in practice. Conceptually, the method is simple to describe and can be summarized as follows (see the next section for more details):

0. Choose an initial $\lambda \in \Lambda_\epsilon$ and $\gamma > 0$.
1. While x is not properly efficient (up to some given precision):
   - 1.1 Solve (14) for λ fixed to obtain a new x.
   - 1.2 Solve (12) to obtain a new λ. If the objective function is equal to zero, x is properly efficient.
   - 1.3 If x is not properly efficient, compute the efficient solution $x_e \in \arg\min_{x \in X} f_\lambda(x)$ corresponding to λ. Keep the best such solution in memory.
   - 1.4 Increase γ.

This scheme has two nice properties:

– Since γ increases at each step, it is guaranteed to generate a limit point which is a properly efficient solution; see Theorem 2 above.

– It generates at each step a properly efficient solution corresponding to the current value of λ and keep the best solution generated by the algorithm. Moreover, since the corresponding λ was computed by solving (12) for an optimal solution x of (14) (for the previous value of λ), it usually has a small value for $\phi(x)$. In fact, we have observed that usually $\phi(x)$ monotonically increases at each step, until reaching the efficient set. (The reason $\phi(x)$ increases is because it is given less and less importance in the objective function as γ increases.)

However, the algorithm does not guarantee to generate an optimal solution of the difficult problem (8): the optimal solution of the surrogate problem (14) as γ tends to infinity might not coincide with the optimal solution of (4), especially since we are not guaranteed that λ will converge to weights corresponding to an optimal solution of (8). However, we will see in the numerical experiments that this approach performs well in practice. Note that this scheme is well defined since (12) always admits an optimal solution (Property 2), while the problems (14) and $\min_{x\in X} f_\lambda(x)$ for λ fixed also admits at least one optimal solution by Weierstrass theorem (the objective functions are continuous and X is a compact set).

4.1 Algorithmic details and implementation

We present in this section the details of the algorithm described above; the pseudo code is given in Algorithm 1.

Initialization and update of γ

A key choice for Algorithm 1 is the value of γ. In fact, if the chosen penalty parameter γ is too large, then the algorithm deals mainly with the efficiency of the solution without taking into account the objective function ϕ, and therefore may stops at an efficient solution far from the solution of problem (8). Conversely, if γ is too small, then the algorithm will be slow as it will take many iterations to get close to an efficient solution.

We propose a simple way to update γ using two parameters and an initial vector $\lambda^{(0)} \in \Lambda_\epsilon$. The two parameters are $\alpha > 0$ that will be used to initialize γ, and $\beta > 1$ that will be a multiplicative factor to increase γ after each iteration of Algorithm 1.

Initially, we solve (5) using $\lambda = \lambda^{(0)}$ to obtain x_e. Then, $\gamma^{(1)}$ is chosen to balance the two objective functions ϕ and f_λ according to the parameter $\alpha > 0$:

$$\gamma^{(1)} = \frac{|\phi(x_e)|}{\alpha |f_{\lambda^{(0)}}(x_e)|}.$$

Hence, the larger the α, the smaller $\gamma^{(1)}$. As we will see, the following choice works very well in practice: $\alpha = 10$, $\beta = 1.1$, and $\lambda^{(0)} = e/p$ where e is the vector of all ones.

Stopping criterion

At each iteration, the algorithm solves three convex subproblems. First, it computes an optimal solution $x^{(k)}$ to (14) for $\gamma = \gamma^{(k)}$ using the previous value of λ. Second, it checks whether $x^{(k)}$ is properly efficient by solving (12), which generates a new λ. Third, it computes the properly efficient solution corresponding to the current value of λ (and keeps the best solution computed so far). The first condition for Algorithm 1 to stop is that $x^{(k)}$ is properly efficient, up to some precision ε, that is, when the optimal value of (12) is smaller than ε. However, it may happen that $x^{(k)}$ is properly efficient while Algorithm 1 has not yet converged. In order to avoid stopping the algorithm in this situation, and possibly allow it to obtain a better solution, we add a second stopping criterion based on distance between two

Algorithm 1: Algorithm for optimizing over the properly efficient set

Data: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, positive semidefinite matrices $Q_i \in \mathbb{R}^{n\times n}$ and vectors $c_i \in \mathbb{R}^n$ for $1 \le i \le p$, a convex function $\phi : \mathbb{R}^n \to \mathbb{R}$.
Parameters: $\epsilon > 0$, $\lambda^{(0)} \in \Lambda_\epsilon$, maxiter, $\varepsilon > 0$, $\tau > 0$, $\alpha > 0$, $\beta > 1$.
Result: Approximate solution x^* to (8)

`/* Initialization */`

Choose the initial weights $\lambda^{(0)} \in \Lambda_\epsilon$.

Compute $x_e \in \operatorname{argmin}_{x\in X} f_\lambda(x)$ for $\lambda = \lambda^{(0)}$.

Set $x^* \leftarrow x_e$, $\lambda^* \leftarrow \lambda^{(0)}$, $\phi^* \leftarrow \phi(x_e)$, $k \leftarrow 1$, $\gamma^{(1)} = \dfrac{|\phi(x_e)|}{\alpha\,|f_{\lambda^{(0)}}(x_e)|}$.

`/* Main loop of the algorihm */`

repeat

  Compute an optimal solution $x^{(k)}$ of (14) for $\gamma = \gamma^{(k)}$ and for $\lambda = \lambda^{(k-1)}$ fixed.

  Compute $(\lambda^{(k)}, z^{(k)}, s^{(k)}, v^{(k)})$, an optimal solution of (12) for $x^* = x^{(k)}$ with objective function $L^{(k)}$.

  `/* This step stops the algorithm when two consecutive iterates are properly efficient while` ϕ `has increased: */`

  if $\max(L^{(k)}, L^{(k-1)}) \le \varepsilon$ **and** $\phi(x^{(k)}) > \phi(x^{(k-1)})$ **then**

    STOP, **return** $x^{(k-1)}$.

  end

  if $L^{(k)} > 0$ **then**

    compute $x_e \in \operatorname{argmin}_{x\in X} f_\lambda(x)$ for $\lambda = \lambda^{(k)}$.

  else

    $x_e = x^{(k)}$.

  end

  `/* Keep the best properly efficient solution found */`

  if $L^{(k)} \le \varepsilon$ **and** $\phi(x^{(k)}) < \phi^*$ **then**

    $x^* \leftarrow x^{(k)}$, $\lambda^* \leftarrow \lambda^{(k)}$, $\phi^* \leftarrow \phi(x^{(k)})$.

  end

  if $\phi(x_e) < \phi^*$ **then**

    $x^* \leftarrow x_e$, $\lambda^* \leftarrow \lambda^{(k)}$, $\phi^* \leftarrow \phi(x_e)$.

  end

  `/* Detect fixed points (see Sect. 4.2) */`

  if $||x_e - x^{(k)}|| < \varepsilon$ **then**

    if $\Delta\mu \ge 0$ **then**

      STOP, **return** $x^{(k)}$, a global optimal solution.

    else

      if *the fixed point has already been attained* **then**

        STOP

      else

        $\gamma^{(1)} \leftarrow \frac{\gamma^{(k)}}{10(1+\bar{\delta})}$ where $\bar{\delta}$ is given in (16), $k \leftarrow 1$.

      end

    end

  else

    `/* Increase the penalty parameter` γ `*/`

    $\gamma^{(k+1)} \leftarrow \beta\gamma^{(k)}$, $k \leftarrow k+1$.

  end

until $k >$ *maxiter* **or** $(L^{(k)} \le \varepsilon$ **and** $d^{(k)} \le \tau)$

iterates:

$$d^{(k)} = \left\| (x^{(k)}, \lambda^{(k)}, z^{(k)}, s^{(k)}, v^{(k)}) - (x^{(k-1)}, \lambda^{(k-1)}, z^{(k-1)}, s^{(k-1)}, v^{(k-1)}) \right\|_2 \leq \tau,$$

for some small τ.

We will also stop the algorithm if two consecutive iterates are properly efficient but the value of ϕ has increased (see step 8 of Algorithm 1). The reason is that since γ monotonically increases, it is not likely for ϕ to decrease during the next iterations in this situation (and, in fact, we have never observed this behavior in practice).

4.2 Fixed points of Algorithm 1

In this section, we address the case when Algorithm 1 terminates early. More precisely, let us analyze the following situation: for some iteration k the algorithm generates

$$x^{(k)} = \operatorname*{argmin}_{x \in X} f_{\lambda^{(k-1)}}(x) + \frac{1}{\gamma^{(k)}} \phi(x)$$

such that $x^{(k)}$ is properly efficient for $\lambda^{(k-1)}$, that is,

$$x^{(k)} = \operatorname*{argmin}_{x \in X} f_{\lambda^{(k-1)}}(x),$$

and solving (12) gives $\lambda^{(k)} = \lambda^{(k-1)}$. Note that this situation can be easily identified within Algorithm 1 as *it will happen when* $x_e = x^{(k)}$. In that case, the algorithm has converged to a fixed point. In fact, it can be easily shown that for any $\gamma' > \gamma$, $x^{(k)}$ will be an optimal solution of $\min_{x \in X} f_{\lambda^{(k)}}(x) + \frac{1}{\gamma'} \phi(x)$ (see below for a proof).

Although we have not observed this particular situation in the numerical examples presented in the next section for the default parameters of Algorithm 1, it is interesting to investigate it and characterize such solutions that could be generated by[1] Algorithm 1. In particular, it will allow us to define a safety procedure to escape such a points.

To simplify the presentation, let us write $X = \{x \in \mathbb{R}^n \mid h_i(x) \geq 0, 1 \leq i = 1, 2, \ldots, t\}$ where the constraints $h_i(x) \geq 0$ represent $Ax = b$, $x \geq 0$ and $G(x) \leq 0$ so that $t = 2m + n + c$. Let us also define the fixed point as $\lambda^* = \lambda^{(k-1)}$ and $x^* = x^{(k)}$, and $\gamma = \gamma^{(k)}$. By the optimality of x^*, we have

$$\nabla f_{\lambda^*}(x^*) = \sum_i \mu_i \nabla h_i(x^*), \mu_i \geq 0, h_i(x^*) \geq 0, \mu_i h_i(x^*) = 0 \text{ for all } i,$$

and

$$\nabla f_{\lambda^*}(x^*) + \frac{1}{\gamma} \nabla \phi(x^*) = \sum_i \mu_i' \nabla h_i(x^*), \mu_i' \geq 0, h_i(x^*) \geq 0, \mu_i' h_i(x^*) = 0 \text{ for all } i.$$

This implies that

$$\frac{1}{\gamma} \nabla \phi(x^*) = \sum_i (\mu_i' - \mu_i) \nabla h_i(x^*).$$

Let us denote the set of active constraints $\mathcal{A} = \{i \mid h_i(x^*) = 0\}$ so that $\mu_i = \mu_i' = 0$ for $i \notin \mathcal{A}$, and $\Delta\mu = \mu' - \mu$. We make the following observations

[1] In particular, we have observed this situation for problem P_9 when $\beta \geq 100$. The reason is that γ increases too fast hence $\phi(x)$ is not given enough importance in the objective function.

- If $\Delta\mu \geq 0$, then x^* is also an optimal solution of $\min_{x\in X}\phi(x)$. Which is the unlikely situation where the optimal solution of $\min_{x\in X}\phi(x)$ is efficient. Hence x^* is a global optimal solution.
- If $|\mathcal{A}| < n$, that is, x^* is not at the intersection of n constraints of X, then $\nabla\phi(x^*)$ and $\nabla f_{\lambda^*}(x^*)$ belong to the same low-dimensional subspace spanned by $\{\nabla h_i(x^*)\}_{i\in\mathcal{A}}$. This will generically not happen (meaning that this happens with probability zero for h_i's, f and ϕ chosen randomly). Hence it is much more likely for a fixed point to be at the intersection of n constraints (see an example below).
- As claimed above, by increasing γ, x^* remains optimal for the penalized problem. In fact, let $\gamma' > \gamma$, and let us define $\delta > 0$ such that $\gamma = \gamma'\frac{1}{(1+\delta)}$. Since $\mu \geq 0$ and $\mu' = \mu + \Delta\mu \geq 0$, we have $\mu'' = \mu + \frac{1}{(1+\delta)}\Delta\mu \geq 0$ so that

$$\nabla f_{\lambda^*}(x^*) + \frac{1}{\gamma'}\nabla\phi(x^*) = \nabla f_{\lambda^*}(x^*) + \frac{1}{(1+\delta)\gamma}\nabla\phi(x^*) = \sum_i \mu_i''\nabla h_i(x^*), \quad (15)$$

  which proves optimality of x^* (it is a KKT point of the penalized convex problem).
- By decreasing γ sufficiently, we can escape x^*, assuming the gradient $\{\nabla h_i(x^*)\}_{i\in\mathcal{A}}$ are linearly independent. In fact, unless x^* is a global optimal solution (see above), there exists j such that $\mu'_j - \mu_j < 0$. Moreover, using the same derivations as above but with $\gamma = \gamma'(1+\delta)$, we have that $\mu'' = \mu' + (1+\delta)\Delta\mu$ is the unique multiplier that satisfies (15) (by the linear independence assumption). Moreover, we have that

$$\begin{aligned}\mu''_j = \mu'_j + (1+\delta)\Delta\mu_j < 0 &\iff \mu'_j < -(1+\delta)\Delta\mu_j \\ &\iff \delta > \frac{\mu'_j}{-\Delta\mu_j} - 1 = \frac{\mu'_j}{\mu_j - \mu'_j} - 1 > 0.\end{aligned}$$

  Therefore, for $\gamma' \leq \frac{\gamma}{(1+\bar{\delta})}$ where

$$\delta > \bar{\delta} = \min_{j,\Delta\mu_j<0} \frac{\mu'_j}{-\Delta\mu_j} - 1, \quad (16)$$

  x^* is no longer an optimal solution of the penalized problem. For Algorithm 1, we use $\gamma' = \frac{\gamma}{10(1+\bar{\delta})}$.

Example 5 Let us consider the problem where $X = \mathbb{R}_+$,

$$f_1(x) = \frac{1}{2}(x+1)^2,\ f_2(x) = \frac{1}{2}(x-1)^2 \text{ and } \phi(x) = -x.$$

We have $X_E = [0, 1]$ hence the optimal solution of $\min_{x\in X_E}\phi(x)$ is $x^\dagger = 1$. However, the point $x^* = 0$ for any $\lambda_1 > 1/2$ and $\frac{1}{\gamma} > 2\lambda_1 - 1$ is a fixed point of Algorithm 1. In fact, the efficient point corresponding to $\lambda_1 > 1/2$ is $x^* = 0$ while, for any $\frac{1}{\gamma} > 2\lambda_1 - 1$, we have

$$\operatorname*{argmin}_{x\geq 0} \lambda_1\frac{1}{2}(x+1)^2 + (1-\lambda_1)\frac{1}{2}(x-1)^2 - \frac{1}{\gamma}x = \max\left(0, \frac{1}{\gamma} - 2\lambda + 1\right) = 0.$$

Hence, without the safety procedure, if Algorithm 1 is run with inappropriate parameters, it may converge to the global maximum $x^* = 0$. For example, using $\lambda^{(0)} = (0.99, 0.01)$ and $\alpha \leq 0.1$ will generate the solution 0.

The situation of converging to a suboptimal fixed point was observed on the 12 small-size problems analyzed in Sect. 5.1 only when α is chosen too small or β too large. On the 45

larger problems from Sect. 5.3, it sometimes happens even when Algorithm 1 is run with its defaults parameters. However, this situation can be easily detected and such non-optimal fixed points can be easily escaped by decreasing γ sufficiently (see above). We have added a safety procedure in Algorithm 1 to do so which allows Algorithm 1 to converge to $x^\dagger = 1$ for any choice of the parameters (in particular, for $\lambda^{(0)} = (0.99, 0.01)$ and $\alpha \leq 0.1$). Note that it may happen that the fixed point is optimal. In that case Algorithm 1 will take more iterations as it will restarted with a smaller value of γ. To avoid this situation, the detection of fixed points is not always activated within Algorithm 1. We will only activate it on the 45 examples from Sect. 5.3 when it was not able to compute the true nadir value with the proposed parameter values.

5 Computational results

To illustrate the performance and efficiency of the proposed method both in linear and non-linear cases, we consider some test problems from the literature. We use the following values for the parameters of Algorithm 1:

- $\varepsilon = 10^{-6}$, the efficiency tolerance,
- $\tau = 10^{-4}$, the tolerance for the distance between two iterates which we will refer to as the stationarity condition,
- $\alpha = 10$, $\beta = 1.1$, $\epsilon = 10^{-4}$, and
- $\lambda^{(0)} = \frac{e}{p} \in \Lambda_\epsilon$.

The algorithm was implemented in Matlab and the solutions of the convex problems are obtained by using CVX (Grant and Boyd 2017, 2008), using the interior-point method SDPT3 (version 4.0) (Toh et al. 1999; Tütüncü et al. 2003). Experiments have been performed on a PC Intel(R) CORE(TM) i3-2310M CPU @ 2.10GHz 4GO RAM. The code is available from

http://bit.ly/OptimEfficientSetv2

We first perform numerical experiments on small-scale problems from the literature for which the optimal solutions are known (Sect. 5.1). Then we perform a sensitivity analysis of Algorithm 1 to the above parameters on the same set of problems (Sect. 5.2). Finally, we apply Algorithm 1 to medium-scale problems motivated by the computation of nadir points (Sect. 5.3).

Note that the limitation to use Algorithm 1 is the size of the three convex subproblems to be solved. Using CVX on a standard laptop would allows us to solve problems of size up to a few hundred variables (for example, solving a LCQP with 500 variables and 100 constraints requires about 10 seconds).

5.1 Small-scale problems from the literature

We first illustrate the performance of Algorithm 1 on several small-scale problems from the literature.

Example 6 (Horst and Thoai 1999) This example, which we will denote P_1, is a linear multi-objective problem with $m = 5$, $n = 7$ and $p = 2$, with

$$A = \begin{pmatrix} 1 & -2 & & & \\ -1 & 1 & & & \\ 2 & 1 & I_5 & & \\ 2 & 5 & & & \\ -1 & -1 & & & \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \\ 4 \\ 10 \\ -1.5 \end{pmatrix}, \begin{pmatrix} c_1^T \\ c_2^T \end{pmatrix} = \begin{pmatrix} 0.2 & -1 & 0_{1\times 5} \\ -1 & -0.1 & 0_{1\times 5} \end{pmatrix},$$

where I_u is an identity matrix of size u and $0_{c\times d}$ is the zero matrix of dimension c-by-d, and Q_1 and Q_2 are zero matrices. The function $\phi(x) = (x_1 - 1.2)^2 - 0.4x_2$ is quadratic.

After four iterations (including the initialization step) and a CPU time of 3.34 seconds, both stopping criteria are satisfied : $L^{(3)} = 1.95 \cdot 10^{-9} < 10^{-6}$ and $d^{(3)} = 3.84 \cdot 10^{-6} < 10^{-4}$, and the algorithm stops, where the best efficient solution found (with 2 digits of accuracy) is

$$x^* = (1.12, 1.55, 2.98, 0.57, 0.21, 0, 1.17) \quad \text{with} \quad \phi(x^*) = -0.614400.$$

This coincide with the optimal solution proposed by Horst and Thoai (1999).

Example 7 (Horst and Thoai 1999) This example, which we will denote P_2, is a linear multi-objective problem with $m = 3$, $n = 7$ and $p = 3$, with

$$A = \begin{pmatrix} 0.476623 & 0.577845 & 0.230673 & 0.834146 & \\ -0.463549 & 0.056380 & -0.952541 & 0.932330 & I_3 \\ 0.541297 & -0.337843 & -0.230437 & -0.804641 & \end{pmatrix},$$

$$\begin{pmatrix} c_1^T \\ c_2^T \\ c_3^T \end{pmatrix} = \begin{pmatrix} -0.462837 & -0.354616 & -0.326963 & 0.129599 & 0_{1\times 3} \\ 0.199892 & -0.013460 & 0.317473 & -0.315895 & 0_{1\times 3} \\ 0.013082 & 0.635298 & 0.412242 & -0.469719 & 0_{1\times 3} \end{pmatrix},$$

$b^T = (3.657495, 1.048547, -0.540701)$, and Q_1, Q_2 et Q_3 are zero matrices. The convex function ϕ is given by

$$\phi(x) = (x_1 - 1.2)^2 - 0.4x_2 + 1.3^{0.1x_3} - \log(1 + 0.2x_4).$$

After five iterations and a CPU time of 4.30 seconds, the algorithm stops and returns the solution

$$x^* = (1.03, 5.47, 0, 0, 0, 1.22, 0.75) \quad \text{with} \quad \phi(x^*) = -1.163113.$$

Note that the algorithm stops at iteration 5 although the stationarity condition is not satisfied: $d^{(4)} = 3.47 \cdot 10^{-3} \not< 10^{-4}$. The reason is that the algorithm does not find a better solutions at iteration 4, so that the step 8 of Algorithm 1 is entered. Note also that the algorithm generates the same solution for $\epsilon = 10^{-9}$ with the same number of iterations.

The optimal solution given by Horst and Thoai (1999) is $\overline{x} = (1.03, 5.48, 0, 0)$ (omitting the slack variables) with $\phi(\overline{x}) = -1.163074$. This solution is obtained after six iterations and for a precision of 10^{-3}. Our algorithm shows a high efficiency by converging to the optimal solution in only four iterations and with a higher precision as $\phi(\bar{x}) - \phi(x^*) \approx 4 \cdot 10^{-5}$.

Example 8 (Benson and Lee 1996) This is also a linear multi-objective problem, with $m = 10$, $n = 20$, which we will denote P_3. The matrix A is given by $A = (I_{10}, I_{10})$ and $b = e$ (the vector of all ones). We have $p = 2$ with c_1 and c_2 being the rows of

$$\begin{pmatrix} 1 & 1 & 1 & 1 & -0.667 & -0.667 & -0.667 & -0.667 & 0.75 & 0.75 & 0_{1\times 10} \\ -1 & -1 & -1 & -1 & 0.333 & 0.333 & 0.333 & 0.333 & -0.25 & -0.25 & 0_{1\times 10} \end{pmatrix},$$

and Q_1 and Q_2 are zero matrices. The function $\phi(x) = d^T x$ is linear with

$$d = (1, 1, -1, 1, -1, 1, -1, 1, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T.$$

After 44 iterations and a CPU time of 32 seconds, the algorithm stops and returns the solution

$$x^* = (0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1)^T \quad \text{with} \quad \phi(x^*) = -1.$$

This solution corresponds to the solution proposed by Benson and Lee (1996).

The second problem proposed by Benson and Lee (1996) minimizes instead a quadratic function given by $\phi(x) = \sum_{j=1}^{10}(11 - j)(x_j - 0.25)^2$, over the same efficient set. We will refer to the corresponding problem as P_4. After 44 iterations and a CPU time of 52 seconds, the algorithm terminates with the efficient solution

$$x^* = (0.25, 0.25, 0.25, 0.25, 1, 1, 1, 1, 0, 0)^T \quad \text{with} \quad \phi(x^*) = 10.312500$$

(slack variables x_j, $j = 11, \ldots, 20$ are omitted), which corresponds to the optimal solution proposed by Benson and Lee (1996).

Example 9 (Benson 2012) Consider the nonlinear multi-objective problem given by

$$\min_{x \in X} \left(x_1^2 + x_2^2 + 0.4x_1 - 4x_2, -\min\{0.5x_1 + 0.25x_2 + 0.2, 2x_1 - 4.6x_2 + 5.8\}\right),$$

where X is defined with $g(x) = 0.5(x_1 - 1)^2 + 1.4(x_2 - 0.5)^2 - 1.1$ is a convex quadratic function, $A = (M, I_7)$,

$$M = \begin{pmatrix} 1 & -2 & 0 \\ -1 & 1 & 0 \\ 2 & 1 & 0 \\ 2 & 5 & 0 \\ -1 & -1 & 0 \\ -0.5 & -0.25 & 1 \\ -2 & -4.6 & 1 \end{pmatrix}, \; b = \begin{pmatrix} 1 \\ 1 \\ 4 \\ 10.5 \\ -1.5 \\ 0.2 \\ 5.8 \end{pmatrix}, \begin{pmatrix} c_1^T \\ c_2^T \end{pmatrix} = \begin{pmatrix} 0.4 & -4 & 0 & 0_{1\times 7} \\ 0 & 0 & -1 & 0_{1\times 7} \end{pmatrix},$$

Q_1 et Q_2 are 10-by-10 matrices, where

$$Q_1(i, j) = \begin{cases} 2 \text{ if } i = j = 1 \text{ or } i = j = 2 \\ 0 \text{ otherwise} \end{cases},$$

and Q_2 is the zero matrix. The objective is given by $\phi(x) = x_1 + x_2^2$. We will refer to this problem as P_5. After 51 iterations and a CPU time of 48 seconds, Algorithm 1 terminates with the properly efficient solution

$$x^* = (0.26, 1.24)^T \quad \text{with} \quad \phi(x^*) = 1.7920,$$

(the variables x_j for $j \geq 3$ are omitted). We obtain the same solution after the same number of iterations and CPU time when we choose $\epsilon = 0$, that is, if the function ϕ is optimized over the weakly efficient set as it is considered by Benson (2012), and we obtain the same optimal solution.

It is worth noting that for all the above problems in this section, Algorithm 1 provides the same optimal solution for $\epsilon = 0$, i.e., when the function ϕ is optimized on the weakly efficient set.

Example 10 (Jorge 2005) We have $m = 4$, $n = 7$, $p = 3$,

$$A = \begin{pmatrix} 1\,1\,0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,0\,1 \end{pmatrix}, b = \begin{pmatrix} 5 \\ 3 \\ 3 \\ 1 \end{pmatrix}, \begin{pmatrix} c_1^T \\ c_2^T \\ c_3^T \end{pmatrix} = \begin{pmatrix} -I_3 & 0_{3\times 4} \end{pmatrix},$$

Q_1, Q_2 and Q_3 are zero matrices.

The objective function is $\phi(x) = x_1$, and we refer to the corresponding problem as P_6.

Recall that X_{wE} is the weakly efficient set of (2).

Setting $\epsilon = 0$ in Algorithm 1, after 3 iterations and a CPU time of 2.70 seconds, the algorithm returns the solution

$$x_{we}^* = (0, 3, 1, 2, 3, 0, 0)^T \quad \text{with} \quad \phi(x_{we}^*) = 0,$$

which is a weakly efficient solution according to Definition 2, since $\min_{x\in X} f_2(x) = f_2(x_{we}^*) = -3$, so it is not possible to find a solution $x \in X$ which is better than x_{we}^* on this criteria. It is also clear that x_{we}^* is an optimal solution of (4) since $\phi(x) \geq 0$ for all $x \geq 0$.

If we set $\epsilon = 10^{-4}$ in Algorithm 1, then it terminates after 122 iterations and a CPU time of 88 seconds, with the efficient solution

$$x^* = (2, 3, 1, 0, 1, 0, 0)^T \quad \text{with} \quad \phi(x^*) = 2,$$

which corresponds to the optimal solution obtained by Jorge (2005). The solution x_{we}^* is not efficient as it is dominated by x^* since $f_1(x^*) = -2 < f_1(x_{we}^*) = 0$.

5.2 Sensitivity analysis

In the previous section, we have observed that for the selected parameters, Algorithm 1 was able to compute the optimal solution for all test problems (comparing the results obtained with those in the references). In this section, we study the impact of the parameters on the efficiency and robustness of our method. Let us first recall the parameters of our method:

- the multiplicative factor β to increase γ,
- the positive constant α to introduce a trade-off between ϕ and f_λ to initialize $\gamma^{(1)}$,
- the initial weight vector $\lambda^{(0)}$.

Note that the last parameter determine the choice of the initial solution $x^{(0)}$. Since there are quite a few parameters, we will only vary one parameter at a time, keeping the others at their default values.

In this section, we introduce two new test problems inspired by the cases tested by Benson and Lee (1996). Using the same data as in Example 8, but replace the vectors c_i's by

$$\begin{pmatrix} c_1^T \\ c_2^T \end{pmatrix} = \begin{pmatrix} 1 & -0.667 & 0.75 & 0_{1\times 17} \\ -1 & 0.333 & -0.25 & 0_{1\times 17} \end{pmatrix},$$

two new problems are obtained and denoted P_7 and P_8, respectively. Our choice to introduce these two problems is motivated by the fact that our algorithm is more sensitive for these cases as shown below.

We also introduce other test problems from the literature, namely P_9, P_{10}, P_{11} and P_{12} that are taken respectively from Ecker and Song (1994), Benson (1993), Benson (1990) and Sayin (2000). For these additional test cases, our method with the default parameters (that is, the parameters given in the previous section) also generates the optimal solutions (compared

to the above references). It is worth noting that, for P_{10}, the optimal solution $\overline{x} = (0, 2, 4)$, with $\phi(\overline{x}) = -2$ where $\phi(x) = x_1 + x_2 - x_3$, proposed by Benson (1993) is not unique, and our algorithm found another optimal solution, namely $x^* = (1, 1, 4)$.

Sensitivity to β Let us start by investigating the influence of parameter β. We are interested in evaluating the gap function $\delta\phi(x) = \phi(x) - \phi(x^*)$, where x is the solution found by our algorithm for a given value of β and x^* is the optimal solution of the problem under consideration. The results are shown in Table 2, where the variation in the number of iterations and in the CPU time are also given.

The results show that the method is sensitive in terms of identifying an optimal solution to the choice of parameter β for problems P_3, P_4, P_5, P_7, P_8 and P_9. In fact, we observe that for β too large, the method converged to a suboptimal solution. This is not surprising and can be explained as follows: by taking β too large, the term $f_\lambda(x)$ takes precedence over the penalty term $\frac{1}{\gamma}\phi(x)$, so our method tries to move fast to an efficient but suboptimal solution. Note that for problem P_7 the method is more sensitive as even for β too small (=1.1), it does not find an optimal solution. This can be explained by the fact that the problem has several local solutions in the neighbourhood of the optimal solution (this will be confirmed in the sensitivity analyses for the other parameters). For the other test cases, Algorithm 1 is very stable: it has always converged to the optimal solution even for β quite large.

In terms of number of iterations and CPU time, not surprisingly, the smaller β, the larger the number of iterations is needed (hence a larger CPU time). The reason is that γ increases faster as β increases, hence Algorithm 1 converges faster to an efficient solution.

Sensitivity to α

For this experiment, we use the following values for α: 0.1, 1, 10, 10^2 and 10^3. The results are shown in Table 3.

More surprisingly, Algorithm 1 is less sensitive to the choice of α than β in terms of identifying the optimal solution. In fact, only for P_2, P_{11}, P_{12} and P_7, it did not find an optimal solution in all cases. For P_2, P_9, P_{11} and P_{12} the explanation is that $\alpha = 0.1$ is too small: γ is initialized at a large value so that Algorithm 1 converges quickly to an efficient solution which is not optimal (with respectively $\delta\phi = 0.56, \delta\phi = 8, \delta\phi = 1.5$ and $\delta\phi = 4$) as ϕ does not have enough importance in the objective function initially. For P_7, the sensitivity is rather due to the sensitivity of P_7 itself, as already noted for the parameter β.

As for β, the choice of α influences the number of iterations and the CPU time: the larger α, the larger the number of iterations as γ is initially chosen smaller.

Sensitivity to $\lambda^{(0)}$ In our last experiment, we investigate the effect of the initial parameter $\lambda^{(0)}$. For this purpose , we have used the following set of vectors in $\mathbb{R}^p$: e/p (e is the vector of all ones), $e_1 = (1, 0, \ldots, 0)$, $e_2 = (0, 1, \ldots, 0)$, $\ldots$ and $e_p = (0, 0, \ldots, 1)$. The results are shown in Table 4.

We observe that the method converges to the optimal solution for all test cases, except for the problem test P_7. As we will see in the next section, Algorithm 1 can be more sensitive to the choice of the initial vector $\lambda^{(0)}$ for larger problems.

Based on our numerical results, we conclude that our algorithm performs well for reasonable choices of the parameters. For the 12 tested problems, Algorithm 1 converges quickly to an optimal solution. Moreover, it is quite robust as it is not very sensitive to its parameters. Note that all of the problems above were solved without the use of the procedure to detect fixed points and most of the non-optimal solutions obtained with the inappropriate parameters could be avoided if the procedure was activated; for example P_9 with $\beta = 100$ in Table 2 and $\alpha = 0.1$ in Table 3, P_{12} with $\alpha = 0.1$ in Table 3.

In the next section, we perform more numerical experiment on medium-scale multi-objective linear optimization problems to confirm these observations.

Table 2 Impact of the parameter β on the distance to optimality $\delta\phi(x)$ of the solution found by Algorithm 1, its number of iterations (including the initialization step) and the CPU time (in seconds)

Problem		β=1.1	β=1.5	β=1.8	β=2	β=10	β=100	β=1000
P_1	Niter	4	4	4	4	4	4	4
($n = 7$)	CPU	3.3	3.4	3.3	3.1	3.3	3.4	3.3
	$\delta\phi$	0	0	0	0	0	0	0
P_2	Niter	5	4	8	5	5	4	4
($n = 7$)	CPU	4.3	3.6	6.7	4.6	4.6	3.7	3.7
	$\delta\phi$	0	0	0	0	0	0	0
P_3	Niter	44	13	10	9	5	4	4
($n = 20$)	CPU	32.7	9.4	7.8	6.8	3.9	3.5	3.3
	$\delta\phi$	0	0	0	0	**2**	**2**	**2**
P_4	Niter	44	14	11	10	6	4	4
($n = 20$)	CPU	53.0	16.2	12.5	11.3	6.6	4.5	4.3
	$\delta\phi$	0	0	0	0	0	**2.13**	**2.13**
P_5	Niter	51	34	32	26	14	8	10
($n = 10$)	CPU	49.1	34.2	31.1	25.0	13.5	8.4	10.9
	$\delta\phi$	0	0	0	**0.10**	**0.10**	**0.10**	**0.10**
P_6	Niter	122	31	23	20	8	6	5
($n = 7$)	CPU	90.4	23.9	16.6	15.7	5.9	4.5	4.2
	$\delta\phi$	0	0	0	0	0	0	0
P_7	Niter	23	8	7	7	4	4	4
($n = 20$)	CPU	17.7	6.5	6.3	5.9	3.6	4.7	4.2
	$\delta\phi$	**1**	**1**	**1**	0	**2**	**2**	**2**
P_8	Niter	38	13	12	9	5	4	4
($n = 20$)	CPU	43.9	15.4	14.0	10.3	5.8	5.2	4.7
	$\delta\phi$	0	0	0	0	**2.38**	**2.38**	**2.38**
P_9	Niter	26	9	7	7	4	4	4
($n = 5$)	CPU	18.5	7.1	5.4	5.1	2.9	3.4	3.3
	$\delta\phi$	0	0	0	0	0	**8**	**8**
P_{10}	Niter	107	28	20	18	8	6	5
($n = 5$)	CPU	81.5	21.3	15.7	13.7	6.4	4.6	4.0
	$\delta\phi$	0	0	0	0	0	0	0
P_{11}	Niter	129	33	24	21	9	6	7
($n = 8$)	CPU	100.8	26.3	19.1	16.1	7.0	5.0	5.5
	$\delta\phi$	0	0	0	0	0	0	0
P_{12}	Niter	119	31	22	19	8	6	5
($n = 9$)	CPU	90.9	24.2	16.6	14.8	6.1	4.7	4.2
	$\delta\phi$	0	0	0	0	0	0	0

In bold we indicate the cases when Algorithm 1 did not find an optimal solution

Table 3 Impact of the parameter α on the distance to optimality $\delta\phi(x)$ of the solution found by Algorithm 1, its number of iterations (including the initialization step) and the CPU time (in seconds)

Problem		$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 100$	$\alpha = 1000$
P_1	Niter	5	4	4	3	3
$(n = 7)$	CPU	3.9	3.9	3.3	2.8	3.0
	$\delta\phi$	0	0	0	0	0
P_2	Niter	5	8	5	27	52
$(n = 7)$	CPU	5.0	7.2	4.4	22.8	44.1
	$\delta\phi$	**0.56**	0	0	0	0
P_3	Niter	3	20	44	68	92
$(n = 20)$	CPU	2.9	14.6	32.5	50.1	69.4
	$\delta\phi$	0	0	0	0	0
P_4	Niter	3	22	44	69	92
$(n = 20)$	CPU	3.4	26.2	51.1	81.3	111.4
	$\delta\phi$	0	0	0	0	0
P_5	Niter	5	28	51	72	98
$(n = 10)$	CPU	4.7	27.1	50.6	68.2	98.3
	$\delta\phi$	0	0	0	0	0
P_6	Niter	3	98	122	146	170
$(n = 7)$	CPU	2.5	71.4	91.3	109.3	128.6
	$\delta\phi$	0	0	0	0	0
P_7	Niter	3	3	23	47	71
$(n = 20)$	CPU	2.6	2.5	18.1	36.6	55.6
	$\delta\phi$	**1**	**1**	**1**	**1**	**1**
P_8	Niter	3	13	38	61	85
$(n = 20)$	CPU	3.3	14.6	43.6	72.1	99.0
	$\delta\phi$	0	0	0	0	0
P_9	Niter	3	3	26	50	74
$(n = 5)$	CPU	2.6	2.6	19.1	36.4	55.2
	$\delta\phi$	**8**	0	0	0	0
P_{10}	Niter	59	83	107	132	156
$(n = 5)$	CPU	44.3	63.43	82.2	100.0	119.5
	$\delta\phi$	0	0	0	0	0
P_{11}	Niter	3	105	129	153	177
$(n = 8)$	CPU	2.6	82.4	100.9	117.0	141.9
	$\delta\phi$	**1.5**	0	0	0	0
P_{12}	Niter	3	95	119	143	167
$(n = 9)$	CPU	2.4	73.7	90.7	110.0	128.8
	$\delta\phi$	**4**	0	0	0	0

In bold we indicate the cases when Algorithm 1 did not find an optimal solution

Table 4 Impact of the parameter $\lambda^{(0)}$ on the distance to optimality $\delta\phi(x)$ of the solution found by Algorithm 1, its number of iterations (including the initialization step) and the CPU time (in seconds)

Problem		$\lambda^{(0)} = e/p$	$\lambda^{(0)} = e_1$	$\lambda^{(0)} = e_2$	$\lambda^{(0)} = e_3$
P_1	Niter	4	4	4	–
($n = 7$)	CPU	3.6	3.5	3.7	–
	$\delta\phi$	0	0	0	–
P_2	Niter	5	25	6	7
($n = 7$)	CPU	4.6	21.3	5.4	6.2
	$\delta\phi$	0	0	0	0
P_3	Niter	44	131	57	-
($n = 20$)	CPU	32.2	98.3	47.0	–
	$\delta\phi$	0	0	0	-
P_4	Niter	44	57	58	-
($n = 20$)	CPU	51.9	69.6	69.9	–
	$\delta\phi$	0	0	0	–
P_5	Niter	51	53	43	–
($n = 10$)	CPU	47.8	51.9	40.7	-
	$\delta\phi$	0	0	0	–
P_6	Niter	122	124	129	115
($n = 7$)	CPU	91.4	92.9	93.3	85.0
	$\delta\phi$	0	0	0	0
P_7	Niter	23	40	58	–
($n = 20$)	CPU	17.4	29.7	41.7	–
	$\delta\phi$	**1**	**1**	**1**	–
P_8	Niter	38	52	53	-
($n = 20$)	CPU	43.0	62.6	62.2	–
	$\delta\phi$	0	0	0	–
P_9	Niter	26	28	43	–
($n = 5$)	CPU	18.8	21.0	30.4	–
	$\delta\phi$	0	0	0	–
P_{10}	Niter	107	115	115	–
($n = 5$)	CPU	87.8	83.3	87.3	–
	$\delta\phi$	0	0	0	–
P_{11}	Niter	129	126	137	–
($n = 8$)	CPU	101.5	99.1	105.6	–
	$\delta\phi$	0	0	0	–
P_{12}	Niter	119	129	124	121
($n = 9$)	CPU	90.6	97.6	95.9	89.4
	$\delta\phi$	0	0	0	0

In bold we indicate the cases when Algorithm 1 did not find an optimal solution

Table 5 Numerical results for the computation of the nadir values from Alves and Costa (2009), where r indicates the problem number

$(n \times m \times r)$	(β, α)=(1.1,10)				$(\lambda^{(0)}, \beta, \alpha)$	Best
	e/p	e_1	e_2	e_3		
$(90 \times 30 \times 1)$	**−162.79**	**−162.79**	**−162.79**	**−162.79**	-	-162.79
	384.79	384.79	384.79	384.79	$(e/p$,100,1)+FP	321.21
	90.04	90.04	90.04	90.04	$(e/p$,1.5,10)	64.68
2	**567.24**	**567.24**	**567.24**	**567.24**	-	567.24
	−188.21	**−188.21**	**−188.21**	**−188.21**	-	-188.21
	191.15	191.15	**94.75**	191.15	-	94.75
3	639.39	639.39	639.39	639.39	$(e/p$,1.2,10)	637.23
	662.45	662.45	629.02	662.45	$(e/p$,100,10^3)	570.97
	701.91	701.9 1	701.91	701.91	$(e/p$,1.5,10)	666.56
4	**−6.41**	**−6.41**	**−6.41**	**−6.41**	-	-6.41
	301.33	**301.33**	**301.33**	**301.33**	-	301.33
	−196.93	**−196.93**	**−196.93**	**−196.93**	-	-196.93
5	**290.41**	**290.41**	**290.41**	**290.41**	-	290.41
	511.77	511.77	511.77	501.18	$(e/p$,1.1,10)+FP	460.65
	310.35	**310.35**	**310.35**	**310.35**	-	310.35
$(120 \times 40 \times 1)$	**−133.40**	**−133.40**	**−133.40**	**−133.40**	-	-133.40
	154.09	154.09	154.09	154.09	$(e_2$,10,0.1)+FP	152.24
	−102.33	**−102.33**	**−102.33**	**−102.33**	-	-102.33
2	**836.79**	**836.79**	**836.79**	**836.79**	-	836.79
	982.04	**772.08**	982.04	982.04	-	772.08
	1291.31	**1279.48**	1291.31	1291.31	-	1279.48
3	448.83	448.83	448.83	448.83	$(e_2$,1.8,10)	448.05
	505.57	**505.57**	**505.57**	**505.57**	-	505.57
	728.67	885.79	1017.33	885.79	-	728.67
4	**491.73**	**491.73**	**491.73**	**491.73**	-	491.73
	272.07	272.07	272.07	272.07	$(e/p$,10,100)	262.28
	−209.03	**−209.03**	**−209.03**	**−209.03**	-	-209.03
5	813.20	813.20	**629.36**	813.20	-	629.36
	886.93	**886.93**	**886.93**	**886.93**	-	886.93
	402.32	402.32	**182.96**	402.32	-	182.96
$(150 \times 50 \times 1)$	**927.85**	**927.85**	**927.85**	**927.85**	-	927.85
	781.00	762.08	762.08	**558.80**	-	558.80
	405.29	**405.29**	**405.29**	**405.29**	-	405.29
2	**1179.75**	**1179.75**	**1179.75**	**1179.75**	-	1179.75
	552.69	552.69	552.69	186.53	$(e/p$,1.2,10)	185.50
	430.95	**430.95**	**430.95**	**430.95**	-	430.95

Table 5 continued

$(n \times m \times r)$	(β, α)=(1.1,10)				$(\lambda^{(0)}, \beta, \alpha)$	Best
	e/p	e_1	e_2	e_3		
3	**−5.74**	**−5.74**	**−5.74**	**−5.74**	-	-5.74
	−22.73	**−22.73**	**−22.73**	**−22.73**	-	-22.73
	64.45	**64.45**	**64.45**	**64.45**	-	64.45
4	**483.31**	**483.31**	**483.31**	**483.31**	-	483.31
	623.76	623.76	**597.92**	**597.92**	-	597.92
	467.29	**467.29**	**467.29**	**467.29**	-	467.29
5	**−8.29**	**−8.29**	**−8.29**	**−8.29**	-	-8.29
	489.55	**489.55**	**489.55**	**489.55**	-	489.55
	−247.51	**−247.51**	**−247.51**	**−247.51**	-	-247.51

In bold we indicate the cases when Algorithm 1 found the optimal value for the corresponding parameters. The last column indicates the optimal value reported in Alves and Costa (2009). The column before last gives parameters for which Algorithm 1 was able to compute an optimal solution (when it was not able to do so for the parameters of the four preceding columns). The symbol FP indicates that the fixed point procedure was used

5.3 Medium-scale problems: application to finding the nadir values

In this section, we apply Algorithm 1 to the problem of computing the nadir values for medium-scale multi-objective linear optimization problems. The nadir point $y = (y_1, \ldots, y_p) \in \mathbb{R}^p$ of a multi-objective optimization problem is characterized by the component-wise maximal values of each objective function over the efficient set: for $k = 1, \ldots, p$,

$$y_k = \max_x f_k(x) \text{ such that } x \in X_E.$$

This problem is in general very hard, as it requires solving p optimization problem over the efficient set. Note that for the bi-objective case ($p = 2$), the nadir values can be efficiently computed; see for example (Ehrgott 2005). Besides the algorithms optimizing over the efficient set, many other approaches have been proposed to find the nadir point; see for example (Wang et al. 2015) and the references therein. Computing nadir values is motivated by several applications; see for example (Deb et al. 2010) and the references therein.

To evaluate the performance of Algorithm 1 in finding the true nadir values, we use all the problems with three objectives used in (Alves and Costa 2009): for each size $(n, m) = (90, 30)$, $(120, 40)$ and $(150, 50)$, there are five different problems for a total of 15 test problems hence 45 nadir values to be computed. Table 5 reports the numerical results. The last column reports the optimal values reported in (Alves and Costa 2009). The second column reports the value computed by Algorithm 1 with its default parameters. We observe that, in 28 out of the 45 cases, it computes an optimal solution. Moreover, in most cases, the value is not far away from optimal, comparing the second and last column of Table 5. The third to fifth columns report the values computed by Algorithm 1 for different initial $\lambda^{(0)}$, namely, e_1, e_2 and e_3. In 35 out of the 45 cases, these three initializations allow to compute the optimal nadir value. The sixth column gives, for each problem, a set of parameters that allowed to compute the optimal nadir values in the 10 cases where the default values of α and β did not allow to compute these values. We observe that increasing the values of α and/or β allows to compute these optimal nadir values in all cases.

Finally, Algorithm 1 was able to find all optimal nadir values obtained in (Alves and Costa 2009) for a good choice of the parameters. In practice, since we do not know the optimal values, we recommend to run Algorithm 1 with different parameters, in particular, $\lambda^{(0)} = e/p, e_1, e_2, e_3, \alpha = 10^k$ for $k = -1, 0, 1, 2, 3$ and $\beta = 1.1, 1.2, 1.5, 1.8$.

6 Conclusion

In this paper, we have considered the problem of optimizing a convex function over the properly efficient set of a convex nonlinear multi-objective problem. This class of problems is very difficult to solve due mainly to the nonconvexity of the properly efficient set. We have proposed a numerical method based on a penalty approach; see Algorithm 1. Although the idea behind our algorithm is relatively simple, it is rather novel, updating the weight vector λ using a modified efficiency test (12), and allows us to obtain optimal solutions in all the tested cases from the literature: 12 small-scale problems coming from 8 different papers and 45 larger-scale problems from Alves and Costa (2009) where the goal is to compute the nadir point of 15 multi-objectives problems with three objectives. Although we have only considered quadratic objective functions in the multi-objective optimization problem, the method can be adapted when these objective functions are smooth and convex, so that the first-order optimality KKT conditions can be written. A direction for further research would be to design a meta-heuristic using Algorithm 1 in order to avoid local solutions and automatically tune the values of the parameters.

Acknowledgements The first and third authors acknowledge the support of the Direction Générale de la Recherche Scientifique et du Développement Technologique (DGRSDT) Grant ID: C0656104. We are grateful to the reviewer whose comments allowed us to improve the manuscript significantly.

References

Alves, M. J., & Costa, J. P. (2009). An exact method for computing the nadir values in multiple objective linear programming. *European Journal of Operational Research*, *198*, 637–646.

Ankhili, Z., & Mansouri, A. (2009). An exact penalty on bilevel programs with linear vector optimization lower level. *European Journal of Operational Research*, *197*, 36–41.

Belousov, E., & Klatte, D. (2002). A Frank–Wolfe type theorem for convex polynomial programs. *Computational Optimization and Applications*, *22*, 37–48.

Benson, H. P. (1984). Optimization over the efficient set. *Journal of Mathematical Analysis and Applications*, *98*, 562–580.

Benson, H. P. (1986). An algorithm for optimizing over the weakly-efficient set. *European Journal of Operational Research*, *25*, 192–199.

Benson, H. P. (1990). An all-linear programming relaxation algorithm for optimizing over the efficient set. *Journal of Global Optimization*, *1*, 83–104.

Benson, H. P. (1992). A finite, nonadjacent extreme-point search algorithm for optimization over the efficient set. *Journal of Optimization Theory and Applications*, *73*, 47–64.

Benson, H. P. (1993). A bisection-extreme point search algorithm for optimizing over the efficient set in the linear dependence case. *Journal of Global Optimization*, *3*, 95–111.

Benson, H. P. (2012). An outcome space algorithm for optimization over the weakly efficient set of a multiple objective nonlinear programming problem. *Journal of Global Optimization*, *52*, 553–574.

Benson, H. P., & Lee, D. (1996). Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. *Journal of Optimization Theory and Applications*, *88*, 77–105.

Bolintinéanu, S. (1993). Minimization of a quasi-concave function over an efficient set. *Mathematical Programming*, *61*, 89–110.

Bolintinéanu, S., & El Maghri, M. (1997). Pénalisation dans l'optimisation sur l'ensemble faiblement efficient. *RAIRO Recherche opérationnelle*, *31*, 295–310.

Bonnel, H., & Morgan, J. (2006). Semivectorial bilevel optimization problem: Penalty approach. *Journal of Optimization Theory and Applications*, *131*, 365–382.
Calvete, H. I., & Galé, C. (2011). On linear bilevel problems with multiple objectives at the lower level. *Omega*, *39*, 33–40.
Chinchuluun, A., & Pardalos, P. M. (2007). A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, *154*(1), 29–50.
Dauer, J. P. (1991). Optimization over the efficient set using an active constraint approach. *Mathematical Methods of Operations Research*, *35*, 185–195.
Deb, K., Miettinen, K., & Chaudhuri, S. (2010). Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation*, *14*, 821–841.
Dessouky, M., Ghiassi, M., & Davis, W.J. (1979). Determining the worst value of an objective function within the nondominated solutions in multiple objective linear programming, working Paper, Department of Mechanical and Industrial Engineering, University of Illinois, Urbana, Illinois.
Dessouky, M. I., Ghiassi, M., & Davis, W. J. (1986). Estimates of the minimum nondominated criterion values in multiple-criteria decision-making. *Engineering Costs and Production Economics*, *10*, 95–104.
Ecker, J. G., & Song, J. H. (1994). Optimizing a linear function over an efficient set. *Journal of Optimization Theory and Applications*, *83*, 541–563.
Ehrgott, M. (2005). *Multicriteria optimization*. Berlin: Springer.
Geoffrion, A. M. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, *22*(3), 618–630.
Geromel, J. C., & Ferreira, P. A. V. (1991). An upper bound on properly efficient solutions in multiobjective optimization. *Operations Research Letters*, *10*, 83–86.
Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In *Recent advances in learning and control* (pp. 95–110).
Grant, M.C., & Boyd, S.P. (2017). CVX: Matlab software for disciplined convex programming. http://cvxr.com/cvx.
Hamel, A. H., Löhne, A., & Rudloff, B. (2014). Benson type algorithms for linear vector optimization and applications. *Journal of Global Optimization*, *59*(4), 811–836.
Horst, R., & Thoai, N. V. (1999). Maximizing a concave function over the efficient or weakly-efficient set. *European Journal of Operational Research*, *117*, 239–252.
Isermann, H., & Steuer, R. E. (1987). Computational experience concrning payoff tables and minimum criterion values over the efficient set. *European Journal of Operational Research*, *33*, 91–97.
Jorge, J. M. (2005). A bilinear algorithm for optimizing a linear function over the efficient set of a multiple objective linear programming problem. *Journal of Global Optimization*, *31*, 1–16.
Karimi, M., & Karimi, B. (2017). Linear and conic scalarizations for obtaining properly efficient solutions in multiobjective optimization. *Mathematical Sciences*, *11*(4), 319–325.
Konno, H., & Inori, M. (1989). Bond portfolio optimization by bilinear fractional programming. *Journal of the Operations Research Society of Japan*, *32*, 143–158.
Konno, H., Thach, P. T., & Tuy, H. (1997). *Optimization on low rank nonconvex structures*. Boston: Springer.
Korhonen, P., & Wallenius, J. (1989). A careful look at efficiency and utility in multiple criteria decision making: A tutorial. *Asia-Pacific Journal of Operational Research*, *6*(1), 46–62.
Kuhn, H.W., & Tucker, A.W. (1951). Nonlinear programming. In *Proceedings of the second Berkeley symposium on mathematical statistics and probability* (pp. 481–492). University of California Press, Berkeley and Los Angeles.
Liu, Z., & Ehrgott, M. (2018). Primal and dual algorithms for optimization over the efficient set. *Optimization*, *67*(10), 1661–1686.
Luc, D. T. (1989). *Theory of vector optimization*. Berlin: Springer.
Luc, L. T. (2001). Reverse polyblock approximation for optimization over the weakly efficient set and efficient set. *Acta Mathematica Vietnamica*, *26*, 65–80.
Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Boston: Kluwer Academic Publishers.
Muu, L. D. (2000). A convex-concave programming method for optimizing over the efficient set. *Acta Mathematica Vietnamica*, *25*, 67–85.
Nocedal, J., & Wright, S. (1999). *Numerical optimization*. New York: Springer.
Ogryczak, W. (2000). Multiple criteria linear programming model for portfolio selection. *Annals of Operations Research*, *97*(1), 143–162.
Phillip, J. (1972). Algorithms for the vector maximization problem. *Mathematical Programming*, *2*, 207–229.
Reeves, G. R., & Reid, R. C. (1988). Minimum values over the efficient set in multiple objective decision making. *European Journal of Operational Research*, *36*, 334–338.

Ren, A., & Wang, Y. (2016). A novel penalty function method for semivectorial bilevel programming problem. *Applied Mathematical Modelling*, *40*, 135–149.
Sayin, S. (2000). Optimizing over the efficient set using a top-down search of faces. *Operations Research*, *48*, 65–72.
Shigeno, M., Takahashi, I., & Yamamoto, Y. (2003). Minimum maximal flow problem: An optimization over the efficient set. *Journal of Global Optimization*, *25*, 425–443.
Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and applications*. New York: Wiley.
Steuer, R. E., Qi, Y., & Hirschberger, M. (2007). Suitable-portfolio investors, nondominated frontier sensitivity, and the effect of multiple objectives on standard portfolio selection. *Annals of Operations Research*, *152*(1), 297–317.
Thach, P. T., & Thang, T. V. (2014). Problems with resource allocation constraints and optimization over the efficient set. *Journal of Global Optimization*, *58*, 481–495.
Thach, P. T., Konno, H., & Yokota, D. (1996). Dual approach to minimization on the set of pareto-optimal solutions. *Journal of optimization theory and applications*, *88*, 689–707.
Thoai, N. V. (2000a). A class of optimization problems over the efficient set of a multiple criteria nonlinear programming problem. *European Journal of Operational Research*, *122*, 58–68.
Thoai, N. V. (2000b). Conical algorithm in global optimization for optimizing over efficient sets. *Journal of Global Optimization*, *18*, 321–336.
Toh, K. C., Todd, M., & Tütüncü, R. (1999). SDPT3–A MATLAB software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, *11*(1–4), 545–581.
Tütüncü, R., Toh, K., & Todd, M. (2003). Solving semidefinite-quadratic-linear programs using SDPT3. *Math Program*, *95*(2), 189–217.
Tuy, H., & Hoai-Phuong, N. T. (2006). Optimization under composite monotonic constraints and constrained optimization over the efficient set. In L. Liberti & N. Maculan (Eds.), *Global optimization: From theory to implementation* (pp. 3–31)., Nonconvex optimization and its applications New York: Springer.
Tuyen, H. Q., & Muu, L. D. (2001). Biconvex programming approach to optimization over the weakly efficient set of a multiple objective affine fractional problem. *Operations Research Letters*, *28*, 81–92.
Wang, H., He, S., & Yao, X. (2015). Nadir point estimation for many-objective optimization problems based on emphasized critical regions. *Methodologies and Application*, *21*, 2283–2295.
White, D. J. (1996). The maximization of a function over the efficient set via a penalty function approach. *European Journal of Operational Research*, *94*, 143–153.
Wierzbicki, A. P. (1980). The use of reference objectives in multiobjective optimization. In *Multiple criteria decision making theory and application* (pp. 468–486). Berlin, Heidelberg, Lecture Notes in Economics and Mathematical Systems: Springer.
Yamada, S., Tanino, T., & Inuiguchi, M. (2000). An inner approximation method for optimization over the weakly efficient set. *Journal of Global Optimization*, *16*, 197–217.
Yamada, S., Tanino, T., & Inuiguchi, M. (2001). An inner approximation method incorporating a branch and bound procedure for optimization over the weakly efficient set. *European Journal of Operational Research*, *133*, 267–286.
Yamamoto, Y. (2003). Optimization over the efficient set: overview. *Journal of Global Optimization*, *22*, 285–317.
Zheng, Y., & Wan, Z. (2011). A solution method for semivectorial bilevel programming problem via penalty method. *Journal of Applied Mathematics and Computing*, *37*, 207–219.