



Optimizing a linear function over an efficient set

Hadjer Belkhiri¹ · Mohamed El-Amine Chergui¹ · Fatma Zohra Ouail¹

Received: 1 August 2020 / Revised: 17 June 2021 / Accepted: 14 July 2021 /
Published online: 5 August 2021

© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

In this work, we deal with a global optimization problem (P) for which we look for the most preferred extreme point (vertex) of the convex polyhedron according to a new linear criterion, among all efficient vertices of a multi-objective linear programming problem. This problem has been studied for decades and a lot has been done since the 70's. Our purpose is to propose a new and effective methodology for solving (P) using a branch and bound based technique, in which, at each node of the search tree, new customized bounds are established to delete uninteresting areas from the decision space. In addition, an efficiency test is performed considering the last simplex tableau corresponding to the current visited vertex. A comparative study shows that the proposed method outperforms the most recent and performing method dedicated to solve (P).

Keywords Multi-objective optimization · Efficient solution · Global optimization · Branch and bound · Efficiency test

1 Introduction

The concept of optimal solution in linear programming with single objective function is no longer valid in multi-objective linear programming (MOLP). Indeed, several objective functions, often conflicting, must be optimized simultaneously and as a consequence; the improvement of one of the functions generates the deterioration of at least one another. Hence, the term optimal solution is replaced by efficient solution in the decision space, or non-dominated solution in the criteria space of the program to be solved.

However, in multi-objective optimization, the set S_E of efficient solutions is generally of considerable size and the choice of a solution that suits the decision

✉ Hadjer Belkhiri
hbelkhiri@usthb.dz

¹ RECITS Laboratory, Faculty of Mathematics, USTHB, Bab Ezzouar, PO 32, 16111 Algiers, Algeria

maker's preferences according to a new criterion, becomes a difficult task. Indeed, it is well known that this problem (henceforth denoted (P)) is a difficult global optimization problem (see for instance Benson (1991), Fülöp and Muu (2000), Thoai (2000), Yamamoto (2002)) since the set S_E of efficient solutions of (P) is nonconvex. Notice that, in this case, the efficient set S_E has two interesting properties. First, S_E is connected and second, it contains extreme points of the polyhedron. An extensive literature review allowed us to notice that several studies have been carried out during the end of the last century. In fact, one of the pioneers in this field is Philip (1972) who studied this problem and suggested for solving it an algorithm that moves from an efficient vertex to an efficient adjacent one yielding an increase in the objective function and a cutting plane technique is used to overcome the difficulty of non-convexity of the efficient set. A lot of papers followed his work using somehow the same technique; we cite for example those proposed by Isermann (1974), Bolinteanu (1993), Ecker and Song (1994) and Fülöp and Muu (2000). In a different way, Benson (1993) proposed a nonadjacent extreme point search algorithm which dispenses with the enumeration.

Not long ago, a new study to obtain all efficient extreme points of MOLP with only three objectives has been published Piercy and Steuer (2019). This approach operates in the decision space and especially useful when working with such MOLPs possessing large numbers of efficient extreme points. Whereas, in this case, working in the criteria space can be very beneficial in terms of execution time.

Recently, Liu et al. (2018) proposed two procedures to solve (P) in the criteria space, which the authors called Primal and Dual Algorithms. In fact, problems of interest are divided in two special cases of (P) where in the first case, they consider S_E as the efficient set of MOLP problem (multi-objective linear program) and in the second case, they consider a CMOP problem (convex multi-objective program). Based on a revised version of Benson's outer approximation algorithm, they first propose a new primal algorithm for MOLP problem by incorporating a bounding procedure in the primal algorithm described in Hamel et al. (2013). Then, this primal algorithm is extended to solve problem (P) in the criteria space.

In the dual algorithm, they introduce dual algorithms to solve cases one and two in dual objective space. The dual algorithms are derived from dual variants of Benson's algorithm Hamel et al. (2013), Löhne et al. (2014). Comparison is done with several state of the art algorithms from the literature on a set of randomly generated instances to demonstrate that they are considerably faster than the competitors.

In Boland et al. (2017) the authors described a new algorithm for optimizing a linear function over the set of efficient solutions of a multi-objective integer linear program (MOILP). To generate the non-dominated solutions, the authors developed a novel criteria space decomposition scheme resulting from a limited number of search sub-spaces created as well as that of disjunctive constraints required to define the single-objective integer program that searches for a non-dominated solution. A computational study shows that the proposed algorithm clearly outperforms Jorge's algorithm Jorge (2009) that defined a sequence of progressively more constrained single-objective integer problems to eliminate not only all the previously generated

efficient solutions, but also any other feasible solutions whose criterion vectors are dominated.

In the present study, we propose a branch and bound based method to solve the global optimization problem (P), named BCO-method. In a structured search tree, a node is created to determine an extreme point in the polytope of the MOLP problem, and a branch defines adjacent edges of the polytope along which we move from an extreme point to a neighbour one to improve at least one criterion. Several evaluation process rules to fathom a node are developed; the corresponding extreme point is efficient or has already been visited, no criterion of the MOLP problem can be improved or the value of the function f to be optimized is less than the best value of f previously encountered (for maximization problem). This has the effect to prune uninteresting branches, which allows us to find the optimal solution for (P) without having to check the entire set of efficient solutions that correspond to efficient extreme points of the MOLP problem. Moreover, results in the simplex tableau are used to launch the current node efficiency test run. In the following section (Sect. 2), we give mathematical formulations and necessary notations for a good understanding of the algorithm developed. Also, we provide some preliminaries on multi-objective optimization. In Sect. 3, the principle of the proposed BCO-method is set forth and the different steps of the algorithm are processed. In Sect. 4, theoretical results and proofs on which the algorithm is based are established. Furthermore, in Sect. 5, an illustrative example is performed to exhibit the resolution mechanism. In Sect. 6, a comparative study is carried out and shows that the BCO-method performs better than the Primal method recently described in reference Liu and Ehrgott (2018). Finally, some concluding remarks and prospects are given in Sect. 7.

2 Mathematical formulation

Let $S = \{x \in \mathfrak{R}^n | Ax \leq b, x \geq 0\}$ be a nonempty compact polyhedron in \mathfrak{R}^n , where A is an $(m \times n)$ -real matrix, b an $(m \times 1)$ -real vector and let c^i be an $(1 \times n)$ - real vectors for $i = 1, \dots, r$; $r \geq 2$. Let C be the $(r \times n)$ -matrix whose rows are the vectors c^i , $i = 1, \dots, r$.

We consider the following multi-objective linear program (MOLP):

$$\begin{cases} \text{Max } Z_i(x) = c^i x & i = 1, \dots, r \\ x \in S \end{cases} \tag{1}$$

A solution $x \in S$ is known as efficient for MOLP problem, if there does not exist another solution $y \in S$ such that $Cy \geq Cx$ and $Cy \neq Cx$. The image of an efficient solution in the criteria space is called a non-dominated solution or Pareto optimal solution. Let $S_E \subset S$ be the set of all efficient vertices of MOLP problem. The number of solutions in S_E being generally very large, deciding to find all of them becomes a very heavy task and requires an exponential computing time. In addition, the choice of the best solution can be very difficult according to the decision maker's preference with a long efficient solutions list. Therefore, it ought to be more computationally practical to generate the preferred efficient solution that maximizes a

decision maker’s utility, corresponding to the following mathematical programming problem to solve in this work:

$$(P) \begin{cases} \text{Max } f(x) \\ x \in S_E \end{cases} \tag{2}$$

where $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$ is a continuous linear function over S and $f(x) = \sum_{j=1}^n f_j x_j$.

f is not necessarily a combination of the MOLP problem’s criteria and can be any linear function. The resolution of the program (P) is based on the relaxed linear program (PR) defined by:

$$(PR) \begin{cases} \text{Max } f(x) \\ x \in S \end{cases} \tag{3}$$

Applying the simplex method, let x_l^* be a basic feasible solution of (PR) obtained at a step l , $l \geq 0$, of the solving procedure dedicated to the program (P) . Let’s note:

1. B_l and N_l : the sets of indices of basic variables and non-basic variables respectively of x_l^* ,
2. \hat{c}_j^i and \hat{f}_j : the reduced costs associated with MOLP problem’s criteria (1), as well as those associated to f in program (2) after the pivoting process,
3. H_l : the set of indices that indicate possible improving directions for the criteria c^i , $i = 1, \dots, r$, to search efficient extreme points in a structured search tree.

$$H_l = (H_l^1 \cap H_l^3) \cup (H_l^2 \cap H_l^4) \tag{4}$$

where: $H_l^1 = \{j \in N_l | \exists i \in \{1, \dots, r\}; \hat{c}_j^i > 0\}$, $H_l^2 = \{j \in N_l | \hat{c}_j^i = 0, \forall i \in \{1, \dots, r\}\}$,
 $H_l^3 = \{j \in N_l | \hat{f}_j \leq 0\}$ and $H_l^4 = \{j \in N_l | \hat{f}_j < 0\}$.

4. $(T)_l$: the following linear program :

$$(T)_l \begin{cases} \text{Max } w = \sum_{j=1}^r e_j \\ Cx - e = Cx_l^* \\ Ax \leq b \\ x \geq 0, e_j \geq 0, j = 1 \dots, r \\ e = (e_j)_{j=1, \dots, r} \end{cases} \tag{5}$$

to test the efficiency of x_l^* like described in Ecker and Kouada (1975). We shall derive below from the program $(T)_l$ a simple procedure, called the BCO-efficiency test, so that we can test whether a basic feasible solution is efficient or not.

In the following, we present the principle of the method.

3 Method description

The main subject of our work is to propose an exact method for solving the global optimization program (P) . To do this, we focus on the optimization of the function f over the feasible set corresponding to the vertices of S , the relaxed domain of S_E , by solving the linear program (PR) using simplex method, without having to revisit those already visited. Indeed, the simplex table is increased by the r rows of the matrix of criteria C , $Z_i(x) = c^i x$ for $i = 1, \dots, r$, to be reevaluated and updated simultaneously with the function f .

Thereby, to solve program (P) , our algorithm is developed to rank the basic feasible solutions of the (PR) program. The algorithm is structured in a search tree in which at each node l , $l \geq 0$, a basic feasible solution x_l^* of program (PR) is generated. If this k -th best ($k \geq 0$) basic feasible solution of (PR) is the first one to satisfy the efficiency test along the branch ending at node l , then, the corresponding node is fathomed and the best value encountered f^* of f is updated as well as the optimal solution x^* which is replaced by x_l^* .

Else, the set H_l is determined to create as many child nodes as there are elements in H_l and each of them will corresponds to another vertex of the domain S , obtained from x_l^* by pivoting with respect to a corresponding direction $j \in H_l$. This process is repeated until all the created nodes are fathomed.

A slave program is launched to test the efficiency of each visited feasible basic solution.

A node l of the search tree is fathomed if one of the three conditions is satisfied:

1. The corresponding feasible basic solution x_l^* is efficient or already visited,
2. The set H_l is empty,
3. $f(x_l^*) < f^*$, where f^* is the best value of f previously encountered.

The algorithm is summarized in the following steps:

3.1 Algorithm of the proposed method

Inputs : A, b, C and f .

Outputs : x^* : optimal solution of (P) and $f^* = f(x^*)$.

Step 0: (Initial step)

$l := 0$; (**search tree root**); $f^* = -\infty$; $VN := \emptyset$; (**set of generated vertices of S**).

0.1. Solve (PR) at node 0. If it is not feasible, then terminate, (P) is unfeasible.

0.2. Otherwise, let x_0^* be an obtained optimal solution for (PR) , $VN := \{x_0^*\}$.

0.2.1 If the solution x_0^* is efficient, $x^* := x_0^*$ and $f^* := f(x_0^*)$. Terminate.

0.2.2. Else:

- Determine the sets N_0 and H_0 .
- For each $j \in H_0$, a node $l, l = 1, \dots, |H_0|$, will be created in the search tree.
- Go to Step 1.

Step 1: (General step)

As long as there are nodes in the search tree not fathomed, do:

1.1. Choose a node l from the search tree regarding depth first strategy. Suppose that the non-basic variable of index $j, j \in H_k$, is considered at this node, where node k denotes the parent of the node $l, l \geq k$.

1.2. Consider the simplex tableau at node k and pivot according to the non-basic variable of index $j \in H_k$.

1.3. Let x_l^* be the obtained basic feasible solution of program (PR):

1.3.1. If $x_l^* \in VN$, the node l is fathomed.

1.3.2. Else, $VN := VN \cup \{x_l^*\}$.

1.3.2.1. If $f^* \geq f(x_l^*)$, then the node l is fathomed. Go to Step 1.

1.3.2.2. Otherwise, applied the efficiency test for the solution x_l^* :

1.3.2.2.1. If it is efficient, then $x^* := x_l^*$ and $f^* := f(x^*)$. The node l is fathomed, go to Step 1.

1.3.2.2.2. Else, determine the sets N_l and H_l .

1.4. For each $j \in H_l$, a node $p, p = q + 1, \dots, q + |H_l|$, will be created in the search tree, where q is the number of nodes already created in the search tree.

1.5. Go to Step 1.

In what follows, we highlight the main theoretical results which ensure the convergence of our algorithm.

4 Theoretical results

In this section, the justifications of the steps described in the above algorithm are established. Let x_l^* be the basic feasible solution of program (PR) obtained at node l of the search tree.

Theorem 1 *Let $x, x \neq x_l^*$, an efficient feasible basic solution of which x_l^* is the parent in the search tree. Then x is obtained according to a pivot column $j \in H_l^1 \cup H_l^2$.*

Proof Suppose for any coordinate x_j of $x, j \in N_l$, we have : $j \notin H_l^1 \cup H_l^2$. Then $j \notin H_l^1$ and $j \notin H_l^2$ hence, for all j in $N_l, \hat{c}_j^i \leq 0, \forall i \in \{1, \dots, r\}$ with at least one strict inequality. Then we have: $\forall i \in \{1, \dots, r\}, Z^i(x) = Z^i(x_l^*) + \sum_{j \in N_l} \hat{c}_j^i x_j \Rightarrow, \forall i \in \{1, \dots, r\}, Z^i(x) < Z^i(x_l^*)$ because $\sum_{j \in N_l} \hat{c}_j^i x_j < 0$.

Thus, the vector $Z(x)$ is dominated by the vector $Z(x_l^*)$: which means that x is not efficient. □

Remark 1 As $H_l = (H_l^1 \cap H_l^3) \cup (H_l^2 \cap H_l^4)$, then $H_l \subseteq (H_l^1 \cup H_l^2)$. According to theorem 1, only the moves in the directions $j \in H_l$ can ensure to find efficient solutions of the MOLP problem.

Theorem 2 *Let $x, x \neq x_l^*$, an optimal solution for program (P) of which x_l^* is the parent solution in the search tree. Then, x is obtained according to a pivot column $j \in H_l$.*

Proof Suppose that for any coordinate x_j of x , $j \in N_l$, $j \notin H_l$. Then $j \notin (H_l^1 \cap H_l^3)$ and $j \notin (H_l^2 \cap H_l^4)$.

So we have the following cases:

1st case:

$$\left\{ \begin{array}{l} j \in H_l^1 \text{ and } j \notin H_l^3 \\ j \in H_l^2 \text{ and } j \notin H_l^4 \end{array} \right\} \text{ or } \left\{ \begin{array}{l} j \in H_l^1 \text{ and } j \notin H_l^3 \\ j \notin H_l^2 \text{ and } j \notin H_l^4 \end{array} \right\} \text{ or } \left\{ \begin{array}{l} j \notin H_l^1 \text{ and } j \notin H_l^3 \\ j \in H_l^2 \text{ and } j \notin H_l^4 \end{array} \right\} \quad (6)$$

As $j \notin (H_l^3 \cup H_l^4)$, then $\hat{f}_j > 0$ and therefore, the solution x is not optimal.

2nd case:

$$\left\{ \begin{array}{l} j \in H_l^1 \text{ and } j \notin H_l^3 \\ j \notin H_l^2 \text{ and } j \in H_l^4 \end{array} \right\} \quad (7)$$

We obtain a contradiction in this case, because $j \notin H_l^3$ therefore $\hat{f}_j < 0$ and $j \in H_l^4$ thus $\hat{f}_j > 0$.

3rd case:

$$\left\{ \begin{array}{l} j \notin H_l^1 \text{ and } j \in H_l^3 \\ j \in H_l^2 \text{ and } j \notin H_l^4 \end{array} \right\} \quad (8)$$

This corresponds to $\hat{f}_j = 0$ and $\hat{c}_j^i = 0$ for all $i \in \{1, \dots, r\}$. Therefore the solution x is alternative to the solution x_l^* relative to the function f and at the same time to the criteria Z^i $i = 1, \dots, r$. Thus it has the same status as x_l^* which is not efficient, otherwise we would have probed the node l .

4th case:

$$\left\{ \begin{array}{l} j \notin H_l^1 \text{ and } j \in H_l^3 \\ j \notin H_l^2 \text{ and } j \in H_l^4 \end{array} \right\} \text{ or } \left\{ \begin{array}{l} j \notin H_l^1 \text{ and } j \in H_l^3 \\ j \notin H_l^2 \text{ and } j \notin H_l^4 \end{array} \right\} \text{ or } \left\{ \begin{array}{l} j \notin H_l^1 \text{ and } j \notin H_l^3 \\ j \notin H_l^2 \text{ and } j \notin H_l^4 \end{array} \right\} \quad (9)$$

Since $j \notin (H_l^1 \cup H_l^2)$, then the solution x is not efficient according to theorem 1. Thus, theorem 2 is proved by contraposition. □

Proposition 1 *The vertices of set S generated at the nodes of a branch of the search tree, are all different.*

Proof Remind that $H_l = (H_l^1 \cap H_l^3) \cup (H_l^2 \cap H_l^4)$ with $H_l^3 = \{j \in N_l, \hat{f}_j \leq 0\}$ and $H_l^4 = \{j \in N_l, \hat{f}_j < 0\}$. Thus, along a branch of the search tree, when one pivots according to the simplex method from x_l^* towards another vertex x according to a

pivot column $j \in H_l$, i.e. $j \in H_l^3$ or $j \in H_l^4$, the function f does not increase. In addition, the instruction 1.3.1. of the general step of the algorithm prohibits processing of a vertex already visited. \square

Lemma 1 *If at a node l of the search tree, we have $f^* \geq f(x_l^*)$ then, there is no solution $x \in S$, when we move in the directions of H_l , such that $f(x) > f^*$.*

Proof From x_l^* , if we move in the directions of H_l , the function f decreases. Thus, $f(x) \leq f(x_l^*)$ for any solution x obtained by moving in the directions of H_l . \square

Theorem 3 *The program $(T)_l$ defined by Eq. 5 is equivalent to the following program:*

$$(R)_l \begin{cases} \text{Max } w = 1_{1 \times r} (C_{B_l} B_l^{-1} A - C) \cdot x + 1_{1 \times r} (C_{B_l} B_l^{-1}) \cdot y \\ B_l^{-1} A \cdot x + B_l^{-1} \cdot y = B_l^{-1} b \\ (C_{B_l} B_l^{-1} A - C) \cdot x + C_{B_l} B_l^{-1} \cdot y + I_{r \times r} \cdot e = 0_{r \times 1} \\ x \geq 0, \quad y \geq 0, \quad e \geq 0 \end{cases} \tag{10}$$

Proof Recall that program $(T)_l$ is :

$$(T)_l \begin{cases} \text{Max } w = \sum_{j=1}^r e_j \\ Cx - e = Cx_l^* \\ Ax \leq b \\ x \geq 0, \quad e_j \geq 0, \quad j = 1 \dots, r \\ e = (e_j)_{j=1, \dots, r} \end{cases}$$

where $1_{r \times r} = (1, 1, \dots, 1)$. Its standard form is then written as follows:

$$\begin{cases} A_{m \times n} \cdot x_{n \times 1} + I_{m \times m} \cdot y_{m \times 1} + 0_{m \times r} \cdot e_{r \times 1} = b_{m \times 1} \dots \dots \dots (7) \\ C_{r \times n} \cdot x_{n \times 1} + 0_{r \times m} \cdot y_{m \times 1} - I_{r \times r} \cdot e_{r \times 1} = Cx_l^* \dots \dots \dots (8) \\ 0_{1 \times n} \cdot x_{n \times 1} + 0_{1 \times m} \cdot y_{m \times 1} - 1_{1 \times r} \cdot e_{r \times 1} - w = 0 \dots \dots \dots (9) \end{cases}$$

Using the current basis B_l at the basic feasible solution x_l^* , $C = [C_{B_l}, C_{N_l}]$, we can write:

$$\begin{cases} B_l^{-1} A \cdot x + B_l^{-1} \cdot y + 0_{m \times r} \cdot e = B_l^{-1} b \dots \dots \dots (10) \\ (C_{B_l} B_l^{-1} A - C) \cdot x + C_{B_l} B_l^{-1} \cdot y + I_{r \times r} \cdot e = 0_{r \times 1} \dots \dots \dots (11) \\ 1_{1 \times r} (C_{B_l} B_l^{-1} A - C) \cdot x + 1_{1 \times r} (C_{B_l} B_l^{-1}) \cdot y + 0_{1 \times r} \cdot e - w = 0 \dots \dots (12) \end{cases}$$

where the Eqs. (10), (11) and (12) are obtained as follows:

(10) = $B_l^{-1} \cdot (7)$; (11) = $C_{B_l} \cdot B_l^{-1} (7) - (8)$ since $C_{B_l} \cdot B_l^{-1} \cdot b_{m \times 1} = Cx_l^*$ and (12) = $1_{1 \times r} \cdot [C_{B_l} \cdot B_l^{-1} \cdot (7) - (8)] + (9)$.

The corresponding linear program is the follows:

$$(R)_l \begin{cases} \text{Max } w = 1_{1 \times r}(C_{B_l}B_l^{-1}A - C).x + 1_{1 \times r}(C_{B_l}B_l^{-1}).y \\ B_l^{-1}A.x + B_l^{-1}.y = B_l^{-1}b \\ C_{B_l}B_l^{-1}A.x + C_{B_l}B_l^{-1}.y + I_{r \times r}.e = Cx \\ x \geq 0, \quad y \geq 0, \quad e \geq 0 \end{cases}$$

Hence, the obtained program provide a feasible simplex tableau for program $(T)_l$ with the basic feasible solution $(x, e) = (x_l^*, 0)$ when $y = 0$. □

In what follows, we propose another particular version of the efficiency-test which exploits the data of the simplex tableau in x_l^* . First of all, solving program $(O)_0$ is a check for efficiency of an optimal solution x_0^* for the program (PR) . To prove the efficiency of the solution x_0^* at step 0.2.1 of the algorithm, we prove the following theorem:

Theorem 4 *The program $(R)_l$ defined above and the following program $(O)_l$ have the same admissible regions:*

$$(O)_l \begin{cases} \text{Max } g(x) = -1_{1 \times m}B_l^{-1}b + (-f_{1 \times n} + 1_{1 \times m}B_l^{-1}A).x + 1_{1 \times m}B_l^{-1}.y \\ B_l^{-1}A.x + B_l^{-1}.y = B_l^{-1}b \\ (C_{B_l}B_l^{-1}A - C).x + C_{B_l}B_l^{-1}.y + Z(x) = C_{B_l}B_l^{-1}b \\ x \geq 0, \quad y \geq 0, \end{cases}$$

where $Z(x) = (Z_i(x))_{i=1, \dots, r}$

Proof As described in the principle of the proposed method, the criteria matrix C evolves in the same way according to the pivoting process in the simplex tableau when solving the program (PR) . Therefore, the initial program to solve is as follows:

$$\begin{cases} \text{Max } f(x) = f_{1 \times n}x + 0_{1 \times m}y \\ A_{m \times n}x + I_{m \times m}y = b_{m \times 1} \\ C_{r \times n}x + 0_{r \times m}y - Z(x) = 0_{r \times 1} \\ x \geq 0, \quad y \geq 0 \end{cases}$$

Its standard form is given by:

$$\begin{cases} A_{m \times n}.x + I_{m \times m}.y = b_{m \times 1} \dots \dots \dots (13) \\ C_{r \times n}.x + 0_{r \times m}.y - Z(x) = 0_{r \times 1} \dots (14) \\ f_{1 \times n}.x + 0_{1 \times m}.y - f(x) = 0 \dots \dots (15) \end{cases}$$

At any node l of the search tree, by considering the basis B_l associated with the basic feasible solution x_l^* , we obtain the following linear system :

$$\begin{cases} B_l^{-1}Ax + B_l^{-1}.y = B_l^{-1}b \dots \dots \dots (16) \\ (C_{B_l}B_l^{-1}A - C).x + C_{B_l}B_l^{-1}.y + Z(x) = C_{B_l}B_l^{-1}b \dots \dots \dots (17) \\ (1_{1 \times m}B_l^{-1}A - f_{1 \times n}).x + 1_{1 \times m}B_l^{-1}.y + f(x) = 1_{1 \times m}B_l^{-1}b \dots \dots \dots (18) \end{cases}$$

where the Eqs. (16), (17) and (18) are obtained as follows:

$$(16) = B_l^{-1}.(13); (17) = C_{B_l}.(16) - (14) \text{ and } (18) = 1_{1 \times m}(16) - (15).$$

Thus, we obtain the standard form of the following linear program:

$$(P)_l \begin{cases} \text{Max } f(x) = 1_{1 \times m}B_l^{-1}b + (f_{1 \times n} - 1_{1 \times m}B_l^{-1}A).x - 1_{1 \times m}B_l^{-1}.y \\ B_l^{-1}Ax + B_l^{-1}.y = B_l^{-1}b \\ (C_{B_l}B_l^{-1}A - C).x + C_{B_l}B_l^{-1}.y + Z(x) = C_{B_l}B_l^{-1}b \\ x \geq 0, \quad y \geq 0, \end{cases}$$

Note that the programs $(P)_l$ and $(O)_l$ have the same feasible solutions set and $g(x) = -f(x)$. Thus, we aim to find an optimal solution of the function $g = -f$ over the corresponding feasible solution set. Furthermore, at node l , $Z(x_l^*) = C_{B_l}B_l^{-1}b$ and to test the efficiency of x_l^* , we have to search a solution x in the domain S , such that $Z(x) \geq C_{B_l}B_l^{-1}b$. Hence, the solution x satisfies the inequality $Z(x) - C_{B_l}B_l^{-1}b \geq 0$ and consequently, $C_{B_l}B_l^{-1}Ax + C_{B_l}B_l^{-1}y \leq Cx$ is obtained from the Eq. (17).

Thus, we conclude that the feasible solutions set of the program $(O)_l$ is the same as that of the program $(R)_l$ defined in Theorem 3. □

Theorem 5 x_0^* is efficient if and only if x_0^* is the unique optimal solution for program $(O)_0$.

Proof \Leftarrow Recall that the admissible domain of $(O)_l$ is not other than $D(x_l^*) = \{x \in S | Cx \geq Cx_l^*\}$ being the optimal solution of the program (PR) (step 0.1 of the algorithm), it maximizes the function f over the domain S , and therefore over the domain $D(x_0^*)$. On the other hand, x_0^* being the unique optimal solution of the program $(O)_0$, it maximizes $(-f(x))$ over $D(x_0^*)$. So, x_0^* minimizes $f(x)$ over $D(x_0^*)$. Thus, both the maximum and the minimum of the function f over the domain $D(x_0^*)$ are achieved at x_0^* , it is therefore the unique feasible solution in the domain $D(x_0^*)$ so that $Cx = Cx_0^*$. For this purpose, the optimal objective value of the program $(T)_0$ is 0 and therefore, x_0^* is efficient.

\Rightarrow Proof by contradiction. Suppose that x_0^* is not an optimal solution of $(O)_0$, then there exists a solution y in $D(x_0^*)$, y different from x_0^* , such that $-f(x_0^*) < -f(y)$. Hence, there exists a criterion i in $\{1, \dots, r\}$, such that $c^i y - c^i x_0^* = e_i$, with $e_i = f(x_0^*) - f(y)$, $e_i > 0$ since $f(x_0^*) > f(y)$. Then, (y, e) is a feasible solution for program $(T)_0$, $e = (e_j)$, $j = 1, \dots, r$, with at least one criterion i such that $e_i = f(x_0^*) - f(y)$ and $\sum_{j=1}^r e_j > 0$. Hence, solution x_0^* is not efficient.

Suppose again that x_0^* is an optimal solution of $(O)_0$, but not unique. Then there exists a solution y in $D(x_0^*)$, y different from x_0^* , such that $f(x_0^*) = f(y)$. Hence, x_0^* is not the unique optimal solution for program (PR) and y is a basic feasible solution in S . This is a situation where the function f is proportional to both a criterion and a constraint of the admissible region S . There is then a criterion i such that $c^i y = c^i x_0^*$,

and a criterion j , $j \neq i$, such that $c^j y > c^j x_0^*$. By setting $e_j = c^j y - c^j x_0^*$, $e_j > 0$ and $\sum_{k=1}^r e_k > 0$, which proves that x_0^* is not efficient. \square

We note $(-O)_l$ the program $(O)_l$ with objective function f to maximize instead of g . Hence, programs $(-O)_l$ and $(O)_l$ have the same admissible region $D(x_l^*)$. Moreover, to ensure the validity of the BCO-efficiency test, program $(-O)_l$ must also be solved taking into account that x_l^* , $l \geq 1$, is not the optimal solution of the program (PR) .

Corollary 1 x_l^* , $l \geq 1$ is efficient if and only if x_l^* is the unique optimal solution for both programs $(O)_l$ and $(-O)_l$.

Proof If x_l^* is not the same optimal solution for the two programs $(O)_l$ and $(-O)_l$, or x_l^* is not the unique optimal solution for one of the two programs, then the domain $D(x_l^*)$ admits a feasible solution y other than x_l^* . Hence, solution x_l^* is not efficient (see Proof of Theorem 5). Conversely, if x_l^* is the unique optimal solution for each of the two programs, this means that the domain $D(x_l^*)$ is reduced to the only solution x_l^* , which means that x_l^* is efficient. \square

Remark 2 In fact, as soon as one of the two programs $(O)_l$ or $(-O)_l$ encounters a solution u different from x^*l , the computation stops indicating that x^*l is not efficient. Moreover, if x^*l is not efficient, no solution obtained from $(O)l$ or $(-O)_l$ can be used because it is not necessarily a basic feasible solution of the admissible region S .

4.1 The BCO-efficiency test algorithm

1. Let $l = 0$ and x_0^* the optimal solution for program (PR) , solve the $(O)_0$ program.
 - If x_0^* is the optimal solution, then x_0^* is efficient (Theorem 5).
 - Else
 - If x_0^* is not optimal for program (PR) , then x_0^* is not efficient,
 - Else (x_0^* is not unique), then x_0^* is not efficient.
2. For $l \geq 1$, let x_l^* be the obtained basic feasible solution for program (PR) , solve the $(O)_l$ program,
 - 2.1. If x_l^* is not an optimal solution, then x_l^* is not efficient,
 - 2.2. If x_l^* is not the unique optimal solution, then x_l^* is not efficient,
 - 2.3. If x_l^* is the unique optimal solution, then solve the $(-O)_l$ program,
 - 2.3.1. If x_l^* is not an optimal solution, then x_l^* is not efficient,
 - 2.3.2. If x_l^* is not the unique optimal solution, then x_l^* is not efficient,
 - 2.3.3. If x_l^* is the unique optimal solution, then x_l^* is efficient (Corollary 1).

Theorem 6 *The algorithm terminates in a finite number of iterations and returns an optimal solution of the program (P) .*

Proof The set S of feasible solutions being compact, it contains a finite number of vertices. According to proposition 1, the algorithm ends in a finite number of iterations. The convergence of the algorithm towards an optimal solution of the program (P) is ensured by lemma 1, corollary 1 and theorems 1, 2 and 5. \square

5 Illustrative example

Let us consider the following problem of optimizing a linear function over an efficient set defined as:

$$(P) \begin{cases} \text{Max } f(x) = -x_1 + 2x_3 \\ x \in S_E \end{cases}$$

where S_E is the efficient vertex set of the following MOLP problem:

$$\begin{cases} \text{Max } Z_1(x) = x_1 - x_3 \\ \text{Max } Z_2(x) = x_2 + x_3 \\ \text{s.t. : } 2x_1 + x_2 \leq 16 \\ \quad 8x_1 + 5x_2 \leq 66 \\ \quad 2x_1 + 3x_2 \leq 27 \\ \quad x_2 \leq 7 \\ \quad x_3 \leq 2 \\ \quad x_i \geq 0 ; i = 1, 2, 3 \end{cases}$$

The relaxed linear program (PR) of (P) is then written as follows:

$$(PR) \begin{cases} \text{Max } f(x) = -x_1 + 2x_3 \\ \text{s.t. : } 2x_1 + x_2 \leq 16 \\ \quad 8x_1 + 5x_2 \leq 66 \\ \quad 2x_1 + 3x_2 \leq 27 \\ \quad x_2 \leq 7 \\ \quad x_3 \leq 2 \\ \quad x_i \geq 0 ; i = 1, 2, 3 \end{cases}$$

Table 1 Initial step

	x_1	x_2	x_3	b
x_4	2	1	0	16
x_5	8	5	0	66
x_6	2	3	0	27
x_7	0	1	0	7
x_3	0	0	1	2
$-f$	-1	0	-2	-4
$-Z_1$	1	0	1	2
$-Z_2$	0	0	-1	-2

Table 2 Efficiency test 1

	x_1	x_2	x_3	b
x_4	2	1	0	16
x_5	8	5	0	66
x_6	2	3	0	27
x_7	0	1	0	7
x_8	-1/2	0	1	2
y_1	-1	0	1	0
y_2	0	0	-1	0
$-g$	1	0	-2	0

Table 3 Efficiency test 2

	x_4	x_2	x_3	b
x_1	1/2	1	0	8
x_5	-4	5	0	2
x_6	-1	3	0	11
x_7	0	1	0	7
x_8	1	0	1	6
y_1	1/2	0	1	8
y_2	0	0	-1	0
$-g$	-1/2	0	-2	-8

Table 4 Solution x_1^* of (PR)

	x_2	x_4	x_8	b
x_1	1/2	1/2	0	8
x_5	1	-4	0	2
x_6	2	-1	0	11
x_7	1	0	0	7
x_3	0	0	1	2
$-f$	1/2	1/2	-2	4
$-Z_1$	-1/2	-1/2	1	-6
$-Z_2$	1	0	-1	-2

Initial step: $l = 0$; $f^* = -\infty$; After solving the program (PR), the optimal solution thus obtained is $x_0^* = (0, 0, 2)$ as shown in the following simplex tableau:

From Table 1 $VN := \{x_0^*\}$ and the BCO-efficiency test is applied to x_0^* (see Tables 2 and 3). To do this, $f(x)$ is replaced by $g(x) = -f(x)$ in the last tableau, the criteria Z_1 and Z_2 are added as constraints and the obtained tableau will be optimized:

The obtained optimal solution is $u = (8, 0, 0)$, $u \neq x_0^*$, then x_0^* is not efficient. $f^* = -8$. The sets $N_0 = \{1, 2, 9\}$ and $H_0 = \{1, 2, 9\}$ correspond to x_0^* .

Table 5 Solution x_2^* of (PR)

	x_2	x_3	x_4	b
x_1	1/2	0	1/2	8
x_5	1	0	-4	2
x_6	2	0	-1	11
x_7	1	0	0	7
x_8	0	1	0	2
$-f$	1/2	2	1/2	8
$-Z_1$	-1/2	-1	-1/2	-8
$-Z_2$	1	-1	0	0

Step 1

The separation starts with the creation of the node 1 corresponding to the first direction of H_0 , $j = 1$. By pivoting in this direction, we find the optimal solution $x_1^* = (8, 0, 2)$ given by this tableau (Table 4):

$VN := VN \cup \{x_1^*\}$. The BCO-efficiency test shows that the solution x_1^* is not efficient. $N_1 = \{2, 4, 9\}$ and $H_1 = \{9\}$. Using deep first strategy, we create node 2, corresponding to the single direction of H_1 , $j = 9$. By pivoting in this direction, the solution $x_2^* = (8, 0, 0)$ given in Table 5 is obtained:

$VN = VN \cup x_2^*$. x_2^* is efficient by applying the BCO-efficiency test. Therefore $x^* := x_2^*$ and $f^* = -8$. This current node is fathomed. From node 1, we create node

Table 6 Solution x_3^* of (PR)

	x_1	x_7	x_8	b
x_4	2	-1	0	9
x_5	8	-5	0	31
x_6	2	-3	0	6
x_2	0	1	0	7
x_3	0	0	1	2
$-f$	-1	0	-2	-4
$-Z_1$	1	0	1	2
$-Z_2$	0	-1	-1	-9

Table 7 Solution x_4^* of (PR)

	x_6	x_7	x_8	b
x_4	-1	2	0	3
x_5	-4	7	0	7
x_1	1/2	-3/2	0	3
x_2	0	1	0	7
x_3	0	0	1	2
$-f$	1/2	-3/2	-2	-1
$-Z_1$	-1/2	3/2	1	-1
$-Z_2$	0	-3/2	-1	-9

Table 8 Solution x_5^* of (PR)

	x_1	x_7	x_8	b
x_4	2	1	0	16
x_5	8	5	0	66
x_1	2	3	0	27
x_2	0	1	0	7
x_3	0	0	1	2
$-f$	-1	0	-2	-4
$-Z_1$	1	0	1	2
$-Z_2$	0	0	-1	-2

Table 9 Solution x_6^* of (PR)

	x_1	x_3	x_7	b
x_4	2	0	-1	9
x_5	8	0	-5	31
x_6	2	0	-3	6
x_2	0	0	1	7
x_8	0	1	0	2
$-f$	-1	2	0	0
$-Z_1$	1	-1	0	0
$-Z_2$	0	-1	-1	-7

3 corresponding to the second direction of H_0 ; $j = 2$. By pivoting in this direction, the solution $x_3^* = (0, 7, 2)$ is obtained in Table 6:

$VN = VN \cup \{x_3^*\}$. x_3^* is not efficient by applying the BCO-efficiency test. $N_3 = \{1, 8, 9\}$ and $H_3 = \{1, 8, 9\}$, then we create a new node 4 corresponding to the direction of H_3 ; $j = 1$ and we execute a pivoting in this direction to obtain the solution $x_4^* = (3, 7, 2)$ in Table 7:

$VN = VN \cup \{x_4^*\}$. By applying the BCO-efficiency test, we prove that solution x_4^* is efficient. Hence, $x^* := x_4^*$ and $f^* = 1$. Node 4 is fathomed. We create a new node 5 corresponding to the second direction of H_3 ; $j = 8$, and we obtain the solution $x_5^* = (0, 0, 2)$, in Table 8:

Since this solution is already visited, then $VN := VN$ and this node is fathomed. Node 6 corresponding to the third direction of H_3 ; $j = 9$ is created and the new solution $x_6^* = (0, 7, 0)$ is generated in Table 9:

$VN = VN \cup \{x_6^*\}$. This node is fathomed since $f(x_6^*) < f^*$. Node 7 corresponding to the final direction of H_0 ; $j = 9$ is then created and the solution $x_7^* = (0, 0, 0)$ is obtained in Table 10:

$VN = VN \cup \{x_7^*\}$ but $f(x_7^*) < f^*$. This node is fathomed.

The algorithm terminates since all created nodes are fathomed and the optimal solution for (P) is $x^* = (3, 7, 2)$ and $f^* = 1$.

To summarize the proposed method throughout this example, we present a search tree representing states of the nodes during the process in Fig. 1.

Table 10 Solution x_7^* of (PR)

	x_1	x_2	x_3	b
x_4	2	1	0	16
x_5	8	5	0	66
x_6	2	3	0	27
x_7	0	1	0	7
x_8	0	0	1	2
$-f$	-1	0	-2	0
$-Z_1$	1	0	-1	0
$-Z_2$	0	1	1	0

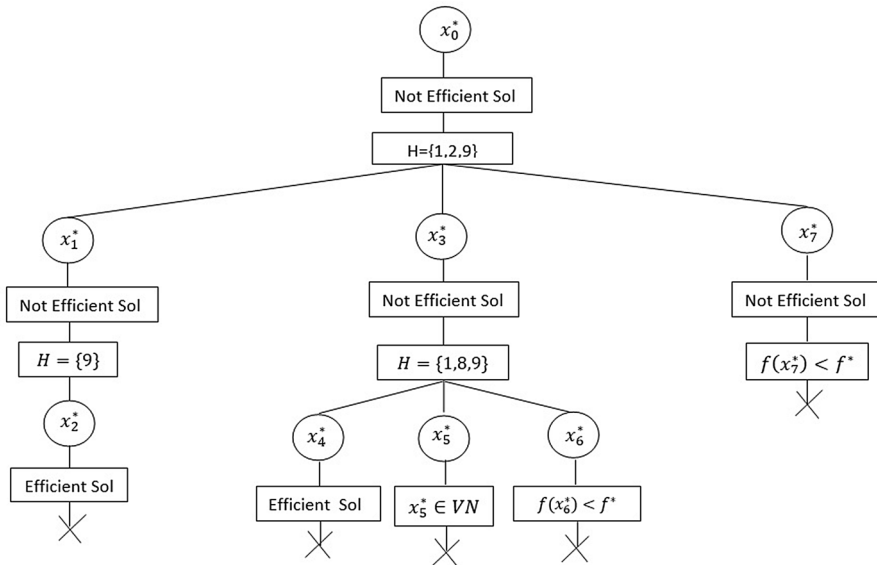


Fig. 1 Search tree

6 Computational results and comparative study

The experimental results obtained during the implementation of our algorithm are summarized in this section. We carried out the numerical implementation using MATLAB R2015 software, on an Intel R Core™ i7-5500U, 2.40Ghz and 8GB Ram processor computer. All the procedures of the BCO-method are programmed and none of the optimization packages are used. The algorithm is compared with the Primal method described in reference Liu and Ehrgott (2018) on instances being on the site: <https://doi.org/10.17635/lancaster/researchdata/224>. For each instance, r is the number of criteria, n is the number of variables and m is the number of constraints. Three instances are given for the same triplet (r, n, m) , all used data are real numbers given with 10^{-16} accuracy and distributed over intervals $[a, b]$, with

minimum value $a < 0$ and maximum value $b > 0$. The constraints matrix values A belongs to the interval $[-1, 0]$, those of the right-hand side vector b vary according to the size of the instance, the larger it is and the more b increases. For example for instance (4, 100, 100), b belongs to $[-27, -20]$. The criteria coefficients vary in $[-1, 1]$. The results of the experimental compilation are given in the following Table 1. We reported the percentage of negative coefficients on average, the intervals of coefficients for the linear function f represented by the max and min values and the mean of CPU time for each method.

Obviously, the computation time increases rapidly as the size of the instances increase. The largest size of the instances tested is 6 objectives, 500 variables and 500 constraints. Even though the authors of the Primal method described in Liu and Ehrgott (2018) praised it because described in the criteria space, it is supposed to support big instances better than in decision space, we observe that the CPU time increases strongly depending on the number of criteria. In this case, the number of criteria has a negative impact on the CPU time since it makes a huge jump of less than one second for less than 5 criteria, at one hour and more for the instances with

Table 11 Results

r	$n = m$	Proportion of $f < 0$ (%)	Interval	CPU (seconds)	
				The BCO-method	Primal method
2	5	60.00	$[-1, 1]$	0.0560	0.0067
	10	40.00	$[-0.9, 1.5]$	0.0022	0.0117
	50	52.00	$[-1, 1]$	0.8098	0.0083
	100	48.00	$[-1.3, 1.3]$	8.6192	0.0309
	500	52.00	$[-1.6, 1.6]$	0.1530	0.0455
3	5	40.00	$[-1.9, 0.6]$	0.0006	0.0214
	10	60.00	$[-1.1, 1.1]$	0.0016	0.0204
	50	48.00	$[-1.2, 1.1]$	2.1227	0.0611
	100	45.00	$[-1.8, 1.5]$	10.6008	0.5441
	500	48.40	$[-2.5, 2.2]$	0.0155	0.0466
4	5	46.60	$[-0.9, 0.9]$	0.0007	0.0874
	10	53.30	$[-2.1, 1.6]$	0.0503	0.3035
	50	56.00	$[-1.4, 1.5]$	0.7697	0.7772
	100	46.00	$[-2.1, 2.2]$	4.1703	1.0124
	500	51.40	$[-2.1, 1.3]$	1380	30.9081
5	5	60.00	$[-2.2, 1.5]$	0.0010	1.19
	10	50.00	$[-1.4, 2]$	0.0174	3.681
	50	52.66	$[-1.8, 1.7]$	0.0392	>3004.86
	100	52.00	$[-2, 2]$	0.1452	>10800.00
6	5	70.00	$[-0.0001, 0.00008]$	0.0011	>5400.00
	10	30.00	$[-0.0001, 0.0001]$	0.0027	>3600.00
	50	44.00	$[-0.00027, 0.0002]$	0.0399	>3600.00
	100	51.50	$[-0.00018, 0.00021]$	0.1408	>7200.00
	500	50.40	$[-0.00026, 0.00025]$	3.7644	>3600.00

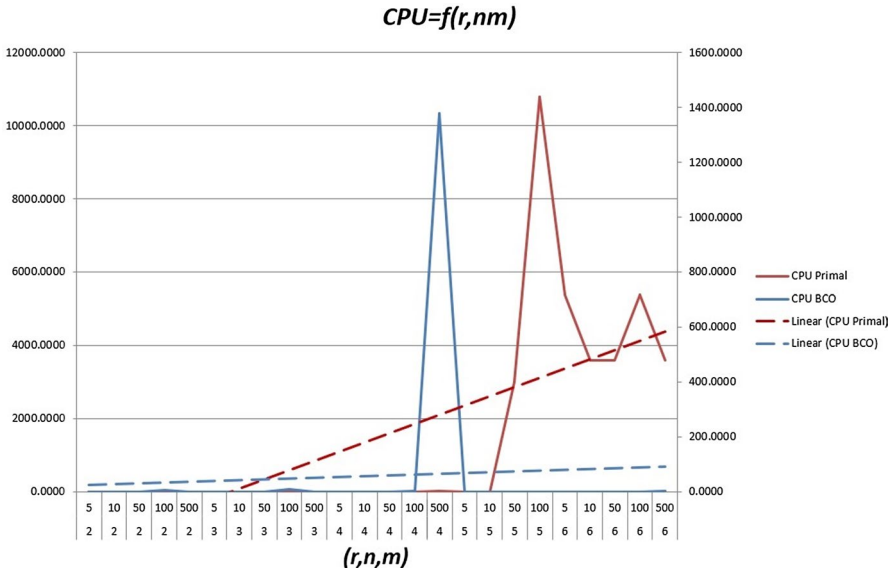


Fig. 2 Graph results

five and six criteria. Given that the BCO-method solves the problem directly in the decision space, it seems to be influenced just by the described instances particularity unlike what we might expect. Regarding the obtained results comparison, we can see that the CPU time of the Primal method Liu and Ehrgott (2018) exceeds the CPUBCO-method on only six (06) instances of the instances presented in Table 11. However, the BCO-method is better on the remaining instances.

We can see from Fig. 2 that the linear trend curve (blue dashed line) of CPU BCO-method is almost horizontal when we except the instance (4,500,500) which represents a singular point with negative coefficients proportion equal to 51.33%. However, the linear trend curve (red dashed line) for the CPU Primal rapidly grows with the criteria number.

Tacking into account that the Primal method Liu and Ehrgott (2018) is better than several state of the art algorithms from the literature Benson (2011), Fülöp and Muu (2000), Thoai (2000), like observed by the authors, we can confirm the competitive statute to our proposed method.

7 Conclusion

In this study, we have proposed a branch and bound based method to search for an efficient extreme point which optimizes a linear function over the set of efficient extreme points of a convex polyhedron. The separation principle is based on the improving directions of the criteria at each feasible extreme point. Moreover, several tests are identified to fathom nodes whose corresponding domains not

contain efficient or optimal extreme points, and thus make it possible to prune entire branches from the search tree. The adapted BCO-efficiency test at a feasible extreme point greatly simplified the computation and, therefore, contributed to accelerating the convergence of the method. The comparative study carried out shows that the BCO-method is very competitive and that the considered number of criteria does not constitute a handicap, unlike many methods cited in the literature. On the other hand, the proposed method can be generalized to other specific functions considered in the program (P), because of its manoeuvrability.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

- Benson HP (1991) An all-linear programming relaxation algorithm for optimizing over the efficient set. *J Global Opt* 1(1):83–104
- Benson HP (1993) A bisection-extreme point search algorithm for optimizing over the efficient set in the linear dependence case. *J Global Optim* 3(1):95–111
- Benson HP (2011) An outcome space algorithm for optimization over the weakly efficient set of a multiple objective nonlinear programming problem. *J Global Optim* 52(3):553–574
- Boland N, Charkghard H, Savelsbergh M (2017) A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *Eur J Oper Res* 260(3):904–919
- Bolintineanu S (1993) Minimization of a quasi-concave function over an efficient set. *Math Program* 61(1–3):89–110
- Ecker JG, Kouada IA (1975) Finding efficient points for linear multiple objective programs. *Math Program* 8(1):375–377
- Ecker JG, Song JH (1994) Optimizing a linear function over an efficient set. *J Optim Theory Appl* 83(3):541–563
- Fülöp J, Muu LD (2000) Branch-and-bound variant of an outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. *J Optim Theory Appl* 105(1):37–54
- Hamel AH, Löhne A, Rudloff B (2013) Benson type algorithms for linear vector optimization and applications. *J Global Optim* 59(4):811–836
- Isermann H (1974) Technical note—proper efficiency and the linear vector maximum problem. *Oper Res* 22(1):189–191
- Jorge JM (2009) An algorithm for optimizing a linear function over an integer efficient set. *Eur J Oper Res* 195(1):98–103
- Liu Z, Ehrgott M (2018) Primal and dual algorithms for optimization over the efficient set. *Optimization* 67(10):1661–1686
- Löhne A, Rudloff B, Ulus F (2014) Primal and dual approximation algorithms for convex vector optimization problems. *J Global Optim* 60(4):713–736
- Philip J (1972) Algorithms for the vector maximization problem. *Math Program* 2(1):207–229
- Piercy CA, Steuer RE (2019) Reducing wall-clock time for the computation of all efficient extreme points in multiple objective linear programming. *Eur J Oper Res* 277(2):653–666
- Thoi NV (2000) Conical algorithm in global optimization for optimizing over efficient sets. *J Global Optim* 18(4):321–336
- Yamamoto Y (2002) Optimization over the efficient set: overview. *J Global Optim* 22:285–317

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.