Optimization and Engineering (2019) 20:89–117
https://doi.org/10.1007/s11081-018-9399-0

RESEARCH ARTICLE

GoNDEF: an exact method to generate all non-dominated points of multi-objective mixed-integer linear programs

Seyyed Amir Babak Rasmi[1] · Metin Türkay[1]

Received: 1 March 2018 / Revised: 30 July 2018 / Accepted: 30 July 2018 /
Published online: 7 August 2018

Abstract

Most real-world problems involve multiple conflicting criteria. These problems are called multi-criteria/multi-objective optimization problems (MOOP). The main task in solving MOOPs is to find the non-dominated (ND) points in the objective space or efficient solutions in the decision space. A ND point is a point in the objective space with objective function values that cannot be improved without worsening another objective function. In this paper, we present a new method that generates the set of ND points for a multi-objective mixed-integer linear program (MOMILP). The Generator of ND and Efficient Frontier (GoNDEF) for MOMILPs finds that the ND points represented as points, line segments, and facets consist of every type of ND point. First, the GoNDEF sets integer variables to the values that result in ND points. Fixing integer variables to specific values results in a multi-objective linear program (MOLP). This MOLP has its own set of ND points. A subset of this set establishes a subset of the ND points set of the MOMILP. In this paper, we present an extensive theoretical analysis of the GoNDEF and illustrate its effectiveness on a set of instance problems.

Keywords Multi-objective optimization · Mixed-integer linear programming · Non-dominated point · Exact method

1 Introduction

Many decision making processes deal with multiple decision criteria. Problems that inherently contain more than one decision criterion are found in a wide range of decision problems in engineering, business, health care, medicine, and chemistry.

✉ Metin Türkay
mturkay@ku.edu.tr

Seyyed Amir Babak Rasmi
srasmi14@ku.edu.tr

[1] Department of Industrial Engineering, Koç University, Istanbul 34450, Turkey

Such problems include genetic network stability, facility location, forward/reverse logistics, urban transportation, and portfolio optimization. Moreover, even single objective problems may be converted to multiple-objective problems to explore the trade-offs between different issues. In the literature of MOOPs, a variety of solution methods were applied to real-world problems such as resource management problems (Can and Erol 2014; Vadenbo et al. 2014), (sustainable) location/transportation problems (Abounacer et al. 2014; Anvari and Turkay 2017; Pascual-González et al. 2016; Charkhgard et al. 2018), disaster planning (Najafi et al. 2013), aerodynamic shape optimization (Nadarajah and Tatossian 2010), vehicle design (Gobbi 2013), and RNA structure prediction (Saule and Giegerich 2015).

There are different solution methods for MOOPs that can be categorized into two broad groups: interactive and non-interactive methods. Interactive methods generate the solutions that are more important for decision makers based on their preferences. For example, Alves and Clímaco (2007) and Miettinen et al. (2016) study interactive methods for MOOPs. Non-interactive methods assume that there is no preference between ND points. On the other hand, exact methods are designed to find either all or a subset of ND points or the efficient solutions set when non-interactive approaches are used. Moreover, evolutionary algorithms find approximate solutions to cover a large subset of the ND points set (e.g., Zitzler 1999; Deb 2001; and Deb et al. 2002).

MOOPs can be categorized into different classes depending on the type of variables and the form of the objective functions and/or constraints. In one aspect, variables can have continuous or discrete values to form a feasible region in the decision space. When there are k number of objective functions, the image of a n-dimensional feasible region onto the objective space is a k-dimensional polyhedron (the dimension of this polyhedron may be less than k) if all variables are continuous; if all variables are discrete (integer or binary), this image becomes a set of single points. Moreover, if both continuous and discrete variables exist, such as mixed-integer programming problems, this image becomes a finite number of most k-dimensional polyhedra in the objective space (we discuss each polyhedron of this case in Sect. 2). The objective functions and constraints of a problem can also be linear or nonlinear that alter the characteristics of the polyhedra. With these differences in the MOOP characteristics, the problem can be categorized into multi-objective (non)linear problem (MO(N)LP), multi-objective integer (non)linear problem (MOI(N)LP), and multi-objective mixed-integer (non)linear problem (MOMI(N)LP).

The concepts of the conventional simplex method with a single objective function to find the ND extreme points of MOLPs are used by Evans and Steuer (1973) and Yu and Zeleny (1975). They provide a thorough theoretical analysis of MOLPs that is built on by other studies (Rudloff et al. 2017; Schechter 2005). Many researchers used similar concepts between 1970 and 1990 to design different algorithms for solving MOLPs (Steuer 1994). Moreover, Ehrgott et al. (2007) present an effective algorithm to find the entire ND points of MOLPs. In general, a number of studies in MOLPs are provided [see Wiecek et al. (2016) for more discussion].

MOI(L)Ps are more complicated than MOLPs. These problems have attracted a great deal of attention to generate a subset, an approximation, or all ND points using (non)interactive methods. Decomposition of 0–1 MOLP into a series of linear/integer programming sub-problems similar to Benders decomposition has been studied (Jahanshahloo et al. 2005; Tohidi and Razavyan 2012). The scalarization techniques are the methods that convert a MOOP to a number of single objective problems with some constraints. These single objective problems must be solved using a systematic approach to generate a large number of ND points. Scalarization techniques are the most preferred techniques to find all or a subset of the ND points (Ehrgott 2006). Weighted-sum scalarization methods (Jorge 2009; Lokman and Köksalan 2013; Sylva and Crema 2004) and the characteristics of the ϵ-constraint method (Özlen and Azizoglu 2009; Özlen et al. 2014; Mavrotas and Florios 2013) for solving MOIPs are used. Boland et al. (2017b, 2016, 2014) present algorithms to generate the ND points set of tri-objective integer linear programs (TOILP). Moreover, finding the Nadir point and optimizing a function over the set of efficient solutions are studied for MOILPs (Boland et al. 2017a). In addition, the ND points set of a MOIP includes too many ND points for large-scale MOIPs; thus, it is more practical to find a subset of this set considering the preferences of decision makers (Lokman and Köksalan 2014).

Exact solution methods for MOOPs with both continuous and integer variables were occasionally addressed in the literature. MOMILPs are very practical for real-world problems since continuous variables often represent operational decisions, and binary/integer variables show strategic/managerial decisions in mathematical programming problems. The class of bi-objective MILPs (BOMILP) is a subclass of MOMILPs with only two objective functions. Belotti et al. (2013), Vincent et al. (2013), Stidsen et al. (2014), Boland et al. (2015), Soylu and Yildiz (2016), and Fattahi and Turkay (2018) propose exact algorithms to generate all ND points of BOMILPs. These algorithms use different methods to find integer solutions that generate ND points. Moreover, since the ND points set of a BOMILP consists of points and line segments, these studies use different techniques to find the ND line segments.

Regarding general MOMILPs with more than two objective functions, a branch-and-bound approach for solving these problems is provided in order to compare the solutions to a reference point determined by decision makers. This approach employs scalarization methods to find a ND point which is close to the reference point in the objective space (Alves and Climaco 2000). An algorithm to solve 0–1 MOMILP for small and medium size problems is presented by Mavrotas and Diakoulaki (2005). The main approach is based on evaluating all possible combinations of binary variables, then finding the ND extreme points of the remaining MOLP, and removing the dominated points by previously-found ND points. These studies do not address the entire ND points of MOMILPs in the form of facets. One important type of ND points is the extreme supported ND (ESN) point. ESN points are the extreme points of the convex hull of all ND points. The set of these ND points is very important and interesting for decision makers. Özpeynirci Ö and Köksalan (2010) and Przybylski et al. (2010) present algorithms to generate all ESN points. Alves and Costa (2016) propose a method to find all ESN points of

tri-objective mixed-integer linear programs (TOMILP). Note that the presence of unsupported ND points and non-extreme supported ND points make generating the entire ND points of a MOMILP difficult (Boland et al. 2015).

To the best of our knowledge, the existing exact algorithms do not address all ND points of MOMILPs in the form of k'-dimensional facets ($0 \leq k' \leq k-1$ and $k \geq 3$). Moreover, although these algorithms find all ND points theoretically, they do not find all entries of the ND points set in practice. In this paper, we present an innovative method to find the ND points of a general MOMILP in the form of facets.

In this paper, we analyze MOMILPs and their ND points in Sect. 2. In this section, an illustrative example is provided for showing the outputs of the GoNDEF. Our groundbreaking method, the GoNDEF, is presented in four steps in Sect. 3. In the description of each step, we also provide theoretical statements to support the accuracy of our method. Then, in Sect. 4, we examine the GoNDEF on a set of instance problems. Finally, we summarize the contributions of the GoNDEF to multi-objective optimization in Sect. 5, where we outline our conclusions.

2 Problem definition

A general MOMILP is given in (1).

$$\begin{aligned} \max \quad & z(x,y) = C_C x + C_Z y \\ \text{s.t.} \quad & A_C x + A_Z y \leq b,\ x \in \mathbb{R}^n, y \in \mathbb{Z}^q, \end{aligned} \tag{1}$$

where $z(x,y) = (z_1(x,y), \ldots, z_k(x,y))$, x and y are n and q-vectors of continuous and integer variables, respectively. C_C and C_Z are $k \times n$ and $k \times q$ matrices, respectively. A_C and A_Z are $m \times n$ and $m \times q$ matrices, respectively, and b is a m-vector. If we set y to a specific vector of integer values (e.g., $\bar{y}$), MOMILP given in (1) is converted to a MOLP. Let sub-MOLP($\bar{y}$) be the MOLP found after fixing integer variables of a MOMILP to $\bar{y}$ as follows:

$$\begin{aligned} &\text{sub-MOLP}(\bar{y}): \\ &\quad \max \quad z(x,\bar{y}) = C_C x + C_Z \bar{y} \\ &\quad \text{s.t.} \quad A_C x \leq b - A_Z \bar{y}, x \in \mathbb{R}^n. \end{aligned} \tag{2}$$

Note that $C_Z\bar{y}$ is a constant k-vector that does not change the efficient solutions of the sub-MOLP given in (2). To simplify our notation, we denote the feasible region of (2) by $S(\bar{y}) := \{x \in \mathbb{R}^n \mid A_C x \leq b - A_Z \bar{y}\}$. Therefore, the feasible region of (1) can be shown as $\{x \in S(y), y \in \mathbb{Z}^q\}$. Moreover, assume that $S(y)$ for an arbitrary feasible y is a closed convex set and there are a finite number of integer solutions which result in nonempty $S(y)$. For the simplicity, we call each entry of the set of feasible integer solutions, the integer solution.

In general, the objective functions of a MOOP are conflicting; hence, the concept of optimality is replaced by the Pareto optimality where one aims to generate the ND points set. If a ND point in the objective space results in the vector of objective

function values $\hat{z}$, then there is no (x, y) such that $\{x \in S(y), y \in \mathbb{Z}^q, z_i(x, y) \geq \hat{z}_i, i = 1, \ldots, k\}$ and at least for one i in $\{1, \ldots, k\}$, $z_i(x, y) > \hat{z}_i$. Consequently, if $z(\hat{x}, \hat{y}) = \hat{z}$ for a feasible $(\hat{x}, \hat{y})$, then $(\hat{x}, \hat{y})$ is an efficient solution.

Let the image of $S(\bar{y})$ onto the objective space be a k-dimensional closed convex polyhedron. Then, the boundary of this polyhedron is a number of connected $(k-1)$-dimensional facets. Moreover, the ND points set of the problem given in (2) is a subset of these facets. Note that each ND facet is the convex hull of a number of ND extreme points. Yu and Zeleny (1975) propose a multi-criteria simplex method to solve MOLPs. This method provides all ND extreme points, all ND facets, and identifies adjacent ND extreme points. Let $NDEP_{\bar{y}}$ and $NDFC_{\bar{y}}$ be the set of all ND extreme points and all ND facets, respectively. Next, we define an efficient integer solution and a ND facet for MOMILPs.

Definition 1 If $\bar{x} \in S(\bar{y})$ exists such that $(\bar{x}, \bar{y})$ is an efficient solution of (1), then $\bar{y}$—as the integer part of this efficient solution—is an efficient integer solution.

Regarding Definition 1, the existence of at least one efficient solution such as $(\bar{x}, \bar{y})$ suffices to call $\bar{y}$, an efficient integer solution. Note that there may exist infinite number of efficient solutions correspond to $\bar{y}$ as an efficient integer solution.

Definition 2 Let F be the convex hull of some ND extreme points of (2) and a ND facet for the sub-MOLP. If $\bar{z} \in F$ exists such that $\bar{z}$ is ND for the MOMILP given in (1), then F is a ND facet for the MOMILP. In this case, facet F is either completely or partially ND.

We provide an instance of MOMILP with three objective functions (TOMILP where $k = 3$). Figure 1 shows this instance in the objective space from two different perspectives (see Appendix 1 for the formulation). We show the existing polyhedra by green, red, and blue colors. Note that fixing the vector of integer variables to each

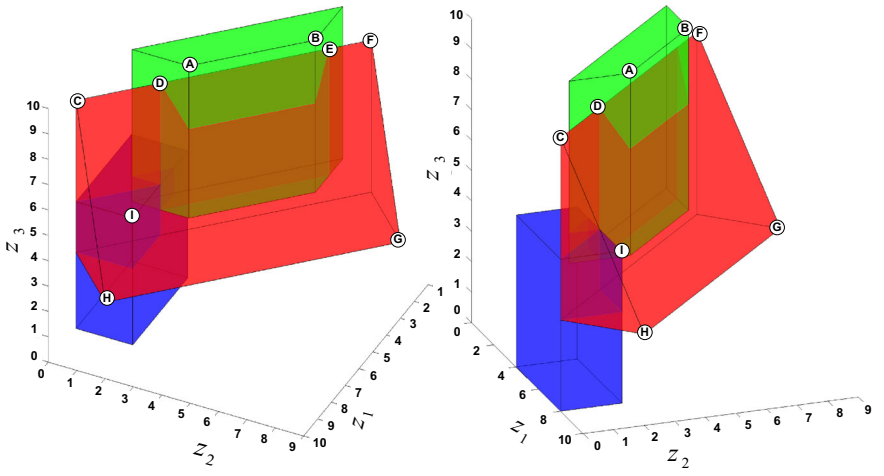

Fig. 1 The illustration of the example in Appendix 1 in the objective space

integer solution gives a polyhedron. We denote these integer solutions by $y_{GR} := (1,0,0)$, $y_{RE} := (0,1,0)$, and $y_{BL} := (0,0,1)$. Then, the images of $S(y_{GR})$, $S(y_{RE})$, and $S(y_{BL})$ onto the objective space correspond to the green, red, and blue polyhedra, respectively. Moreover, note that each polyhedron also corresponds to a sub-MOLP because the integer variables are fixed to (1, 0, 0), (0, 1, 0), or (0, 0, 1).

In this paper, if z^1 and z^2 are two different points in the objective space, then $[z^1, z^2]$ denotes the line segment in the objective space between z^1 and z^2. We also use "[]" and "()" to show closed and open intervals, respectively. For example, $[\text{Ⓒ}, \text{Ⓕ})$ is a line segment that includes point Ⓒ but not point Ⓕ.

In this illustrative example, $[\text{Ⓐ}, \text{Ⓑ}]$ is the ND points set of the green polyhedron. Then, we determine $NDEP_{y_{GR}} = \{\text{Ⓐ}, \text{Ⓑ}\}$, $NDFC_{y_{GR}} = \{ConvexHull\{\text{Ⓐ}, \text{Ⓑ}\}\}$ and identify that Ⓐ and Ⓑ are adjacent (Ⓐ $= (6,3,10)$ and Ⓑ $= (3,6,10)$). This line segment also provides a subset of the ND points set of the MOMILP. The ND points set of sub-MOLP(y_{RE}) is the convex hull of points Ⓒ, Ⓕ, Ⓖ, and Ⓗ where Ⓒ $= (8,0,9)$, Ⓕ $= (1,7,9)$, Ⓖ $= (3,9,3)$, and Ⓗ $= (10,2,3)$. Note that $NDEP_{y_{RE}} = \{\text{Ⓒ}, \text{Ⓕ}, \text{Ⓖ}, \text{Ⓗ}\}$ and $NDFC_{y_{RE}} = \{ConvexHull\{\text{Ⓒ}, \text{Ⓕ}, \text{Ⓖ}, \text{Ⓗ}\}\}$. These four points form a ND facet for the red polyhedron (ignoring the green and blue polyhedra); however, it is a partially ND facet for the MOMILP. This facet is not completely ND since a subset of its points is dominated by $[\text{Ⓐ}, \text{Ⓑ}]$ (see Definition 2). We find $NDEP_{y_{BL}} = NDFC_{y_{BL}} = \{\text{Ⓘ}\}$ (Ⓘ $= (8,2,5)$) by solving sub-MOLP(y_{BL}). Note that the ND facet of sub-MOLP(y_{BL}) is a 0-dimensional facet. Moreover, point Ⓘ is not ND for the MOMILP since it is dominated by some points that belong to the red polyhedron (e.g., (8.02, 2.54, 5.16) which is in $ConvexHull\{\text{Ⓒ}, \text{Ⓕ}, \text{Ⓖ}, \text{Ⓗ}\}$ and dominates Ⓘ). In this case, there are no ND points in the blue polyhedron and hence $y_{GR} = (1,0,0)$ and $y_{RE} = (0,1,0)$ are the efficient integer solutions.

Finding extreme points is an important task in optimization problems and the edges are shown as the convex hull of adjacent extreme points. In Fig. 1, facet Ⓒ − Ⓕ − Ⓖ − Ⓗ is formed by four edges (line segments) which are the boundaries of the facet. Specifying the ND segments of these edges (e.g., $[\text{Ⓒ}, \text{Ⓓ})$,[1] $[\text{Ⓕ}, \text{Ⓖ}]$, and $[\text{Ⓕ}, \text{Ⓔ})$ where Ⓓ $= (6,2,9)$ and Ⓔ $= (2,6,9)$) provides a better and more clear presentation of the partially ND facets since the dominance of the boundaries are identified.

In the next section, we present our method, the GoNDEF, which generates the ND facets of a MOMILP (Ⓒ − Ⓕ − Ⓖ − Ⓗ and $[\text{Ⓐ}, \text{Ⓑ}]$ in the instance associated with Fig. 1) and the ND segments of the edges between pairs of adjacent ND extreme points (e.g., $[\text{Ⓒ}, \text{Ⓓ})$ and $(\text{Ⓔ}, \text{Ⓕ}]$ which are the ND segments of the edge $[\text{Ⓒ}, \text{Ⓕ}]$).

Regarding the partially ND facets, assume that F is a partially ND facet such as Ⓒ − Ⓕ − Ⓖ − Ⓗ. Let $z^1, \ldots, z^{n_F}$ be the corner points of F and ND extreme points. Hence, all points of F are in the convex hull of $\{z^1, \ldots, z^{n_F}\}$. Our method identifies this facet as a ND facet since it includes at least one ND point and our method does not identify which parts of F is dominated. The GoNDEF identifies the

[1] Note that point Ⓓ is not included in the set of ND points since Ⓓ is dominated by Ⓐ. Point Ⓓ is a weakly ND point. Then, we do not address this type of points.

dominated parts of F if only they are shown as the segments of the edges between adjacent ND extreme points. Regarding this issue, assume that we present F to one as a ND facet and he is interested in point $\bar{z} := \sum_{j=1}^{n_F} \lambda_j z^j$ where $\sum_{j=1}^{n_F} \lambda_j = 1$ and $\lambda_j > 0, \forall j = 1, \ldots, n_F$. Then, he can check the dominance of $\bar{z}$ by solving the MILP given in (3) and discussed in Sect. 3.1.

3 GoNDEF: an innovative method for solving MOMILPs

We present a new method that solves a general MOMILP with a unique algorithmic approach. The GoNDEF iteratively finds an integer solution. Then, the method solves the sub-MOLP associated with this integer solution if it is an efficient integer solution (see Definition 1). Note that solving each sub-MOLP results in its ND points in the form of most $(k-1)$-dimensional facets (set *NDFC*) and the ND extreme points (set *NDEP*). We use straightforward operations and excluding constraints and solve single objective LPs/MILPs to generate the ND points set.

Note that we focus on the objective function space and the ND points. Since the number of objective functions is less than the number of variables in general and the objective space is more preferable for decision makers, working in the objective space is more interesting for researchers. Algorithms that operate in the space of decision variables such as Armand and Malivert (1991), Armand (1993) and Sayin (1996) generate efficient solutions, and if there is more than one efficient solution associated with a ND point, they are generated. The computational effort in the algorithms which work in the objective space, however, are less.

The GoNDEF method consists of four main steps. Let *EIS* be the set of explored efficient integer solutions. We set $EIS := \emptyset$ at the start of the algorithm.

Step 1 Find an efficient integer solution $\bar{y}$ such that $\bar{y} \notin EIS$. Then, $EIS := EIS \cup \{\bar{y}\}$. Terminate the algorithm if such an efficient integer solution does not exist. Step 1

Step 2 Solve sub-MOLP($\bar{y}$), which gives $NDEP_{\bar{y}}$ and $NDFC_{\bar{y}}$. Step 2

Step 3 Identify the ND segments of each edge between pairs of adjacent ND extreme points in $NDEP_{\bar{y}}$. Step 3

Step 4 Identify the ND facets of the MOMILP in $NDFC_{\bar{y}}$ by filtering completely dominated facets out. Go to Step 1. Step 4

Assume that we find an efficient integer solution in Step 1 for a MOMILP. Hence, some ND points of the sub-MOLP associated with this efficient integer solution are ND for the MOMILP. The ND frontier of this sub-MOLP is generated in Step 2. In this frontier, there are edges between pairs of adjacent ND extreme points and facets. Note that solving sub-MLOPs in Step 2 of the GoNDEF could become very time-consuming. Hence, Step 1 generates integer solutions that are efficient in order to save computational effort. In other words, we do not solve the sub-MOLPs of integer solutions which do not contribute to the ND points set. In Step 3, we identify which segments of these edges are dominated and which segments are ND for the MOMILP. Regarding Step 4, set *NDFC* contains a number

of facets that are ND for the sub-MOLP. Note that a facet in *NDFC* is completely dominated, partially ND, or completely ND for the MOMILP. Then, in Step 4, we filter the completely dominated facets out and show the ND facets of the MOMILP.

3.1 Step 1: finding the efficient integer solutions

In Step 1, we aim to find integer solutions that are efficient in order to avoid solving the sub-MOLPs that do not contribute to the ND points set. The following MILP problem is a reformulation of two problems provided by Steuer and Choo (1983)[2] and Mavrotas and Diakoulaki (2005) to check the dominance of a point.

$$\begin{aligned} \text{DZ} \quad & (\hat{z}, Y_{exc}, ExC): \\ & \max \quad \sum_{i=1}^{k} \epsilon_i \\ & \text{s.t.} \quad z_i(x,y) - \epsilon_i \geq \hat{z}_i, \ i = 1,\ldots,k, \\ & \qquad x \in S(y), y \in \mathbb{Z}^q \backslash Y_{exc}, ExC, \\ & \qquad \epsilon_i \geq 0, \ i = 1,\ldots,k, \end{aligned} \tag{3}$$

where $\hat{z} \in \mathbb{R}^k$ is a point in the objective space. *ExC* is the set of constraints to exclude the dominated cone of some points from the feasible region. Let $exd(\bar{z})$ be the set of constraints that exclude the dominated cone of $\bar{z} \in \mathbb{R}^k$ in the objective space. Then, $exd(\bar{z}) := \{\bar{z}_i + \delta \leq z_i(x,y) + Mt_i, \sum_{i=1}^{k} t_i \leq k-1, t_i \in \{0,1\}, i = 1,\ldots,k\}$ excludes the dominated cone of $\bar{z}$ where M is a sufficiently large positive number and δ is a sufficiently small positive number. Note that if $\bar{z}$ is a ND point and we add $exd(\bar{z})$ to (3), then $\delta = 0$ may result in a point that is weakly ND. So, we can establish the constraints in *ExC* using *exd*(.)'s. Moreover, Y_{exc} is the set of excluded integer solutions using no-good constraints (Hooker 1994, 2011; Soylu and Yildiz 2016). Assume that (3) is feasible, then (x^D, y^D) denotes the solution that results in optimal objective function value and $z^D := z(x^D, y^D)$. Note that $\text{DZ}(\hat{z}, Y_{exc}, ExC)$ results in $y^D \notin Y_{exc}$. Moreover, z^D dominates $\hat{z}$ and is not in the dominated cone of some points that their dominated cones are excluded.

In the GoNDEF, the problem given in (3) is used in two ways based on the value of entries of vector $\hat{z}$.

DZ1 Assume that both Y_{exc} and *ExC* are empty sets, and $\hat{x} \in S(\hat{y}), \hat{y} \in \mathbb{Z}^q$ exist such that $\hat{z} = z(\hat{x}, \hat{y})$. Then, if the optimal objective function value is zero or DZ is infeasible, then $\hat{z}$ is a ND point. Otherwise, $\hat{z}$ is dominated by z^D.

Remark 1 Assume that $\hat{z}$ is ND for the sub-MOLP($\hat{y}$). Then, $\text{DZ}(\hat{z}, \hat{y}, \emptyset)$ can be used for checking the dominance of $\hat{z}$. If the optimal objective function value is zero or DZ is infeasible, then $\hat{z}$ is a ND point.

[2] This study provides a similar formulation to minimize Tchebycheff distance between the points in the objective space and the Ideal point.

DZ2 Assume that $\hat{z}$ is an arbitrary point in $\mathbb{R}^k$ and the feasible region of (3) is not empty. Moreover, assume that $Y_{exc} = ExC = \emptyset$. Then, z^D is ND.

Proof The optimal objective function value is $\sum_{i=1}^{k}(z_i^D - \hat{z}_i)$. Assume to the contrary that z^D is dominated. Then, a ND point such as $z^{ND} := z(x^{ND}, y^{ND})$ exists such that $z_i^{ND} \geq z_i^D$, $i = 1, \ldots, k$, and at least for one i, $z_i^{ND} > z_i^D$. Then, $\sum_{i=1}^{k}(z_i^D - \hat{z}_i) < \sum_{i=1}^{k}(z_i^{ND} - \hat{z}_i)$ which contradicts that the optimal objective function value is $\sum_{i=1}^{k}(z_i^D - \hat{z}_i)$. □

Let z^S be an approximation of the Nadir point (ANP) of (1) such that z^S is dominated by the Nadir point. For example, $z_i^S := min\{z_i(x, y) | x \in S(y), y \in \mathbb{Z}^q\}$, for $i = 1, \ldots, k$. Then, all ND points of (1) are included in the feasible region of $\text{DZ}(z^S, \emptyset, \emptyset)$.

In the next two subsections, we provide a method to check the efficiency of an integer solution and find all efficient integer solutions.

3.1.1 Checking the efficiency of an integer solution

Let y^* be an integer solution and we aim to find a ND point in the image of $S(y^*)$ onto the objective space. If such a ND point exists, then y^* is an efficient integer solution. Otherwise, y^* is not an efficient integer solution. In Algorithm 1, we develop a method to check the efficiency of y^*. If we set $Y_{exc} := \mathbb{Z}^q \backslash \{y^*\}$, then $\text{DZ}(z^S, Y_{exc}, ExC)$ in the second line of Algorithm 1 results in z^D that is ND for sub-MOLP(y^*) (for discussion, see DZ2).[3] If, regarding DZ1, z^D is ND for the MOMILP, then y^D is an efficient integer solution. Otherwise (line 8 of Algorithm 1), we find a point such as z^{D1} that dominates z^D (line 9). Then, we add the constraints that exclude the dominated cone of z^{D1} to ExC and return to the second line of Algorithm 1. Note that if we do not update ExC, the same dominated point will be found in the next iteration.

Algorithm 1 Check the efficiency of y^*

1: $ExC := \emptyset$
2: $\text{DZ}(z^S, \mathbb{Z}^q \backslash \{y^*\}, ExC) \rightarrow z^D := z(x^D, y^*)$
3: **if** DZ is infeasible **then**
4: y^* is not an efficient integer solution.
5: **else**
6: **if** z^D is ND **then**
7: y^* is an efficient integer solution.
8: **else**
9: $\text{DZ}(z^D, \{y^*\}, \emptyset) \rightarrow z^{D1} := z(x^{D1}, y^{D1})$ (for discussion, see DZ1 and Remark 1)
10: $ExC := ExC \cup exd(z^{D1})$
11: **go to line** 2

In each iteration of Algorithm 1, we add new constraints to $\text{DZ}(z^S, \mathbb{Z}^q \backslash \{y^*\}, ExC)$ that tighten the region $\{x \in S(y^*)\}$. These new constraints exclude the dominated regions in $\{z(x, y^*) | x \in S(y^*)\}$. We iteratively continue this

[3] Note that $y \in \mathbb{Z}^q \backslash \{\mathbb{Z}^q \backslash \{y^*\}\}$ is equivalent to fixing y to y^*.

procedure to find a ND point or see the infeasibility of DZ in the second line. Then, Algorithm 1 stops in two cases:

Alg1-1 $\text{DZ}(z^S, \mathbb{Z}^q \backslash \{y^*\}, ExC)$ becomes infeasible due to ExC which removes all solutions in $S(y^*)$ (line 4). The infeasibility of DZ shows that all solutions in the region $\{(x, y^*) | x \in S(y^*)\}$ are inefficient.

Alg1-2 We find a ND point such as $z^D := z(x^D, y^*)$ where $x^D \in S(y^*)$ (line 7).

3.1.2 Finding all efficient integer solutions

In this section, we present a method to generate all efficient integer solutions in Step 1. This method is shown in Algorithm 2 and at some of the iterations of this algorithm, Algorithm 1 is called.

In Algorithm 2, we solve $\text{DZ}(z^S, Y_{exc}, ExC)$ iteratively. In each iteration, the solution of the problem given in (3) leads to a potentially ND point (for discussion, see DZ2). Note that due to $Y_{exc} := \emptyset$, at the first iteration, z^D is ND. However, in the next iterations, we add some constraints to DZ by updating Y_{exc} (line 7 of Algorithm 2) and ExC (lines 10 and 13). Then, we cannot guarantee that z^D is ND. z^D may be dominated by some points in $\{z(x, y) | x \in S(y), y \in Y_{exc}\}$ which have been already excluded. Our method solves $\text{DZ}(z^D, \{y^D\}, \emptyset)$ to check the dominance of z^D (line 6). If z^D is ND, then y^D is an efficient integer solution. Note that if z^D is dominated by a ND point such as z^{D1}, then we cannot determine the efficiency of y^D. Hence, we use Algorithm 1 in line 12 to check if y^D is an efficient integer solution. If $x \in S(y^D)$ exists such that $z(x, y^D)$ is ND, then y^D is an efficient integer solution. Hence, we define $z^D := z(x, y^D)$ and exclude the dominated cone of z^D for the next iteration (line 13). We continue this process until $\text{DZ}(z^S, Y_{exc}, ExC)$ becomes infeasible due to ExC and/or Y_{exc}.

Algorithm 2 Generate all efficient integer solutions

```
1:  ExC := ∅, Y_exc := ∅
2:  DZ(z^S, Y_exc, ExC) → z^D := z(x^D, y^D)
3:  if DZ is infeasible then
4:      Stop.
5:  else
6:      Check the dominance of z^D by solving DZ(z^D, {y^D}, ∅)
7:      Y_exc := Y_exc ∪ {y^D}
8:      if z^D is ND then
9:          y^D is an efficient integer solution.
10:         ExC := ExC ∪ exd(z^D)
11:     else
12:         Check the efficiency of y^D by applying Algorithm 1
13:         if y^D is efficient integer solution then ExC := ExC ∪ exd(z^D)*
14:         go to line 2
    * z^D is found by Algorithm 1.
```

Without loss of generality, for the simplicity of presentation, we illustrate this part of our method on a maximization BOMILP example provided in Fig. 2. There are six polyhedra corresponding to integer solutions in Fig. 2. We set the ANP point to (0, 1) and show by the blue point. At the first iteration, we solve $\text{DZ}(z^S, \emptyset, \emptyset)$.

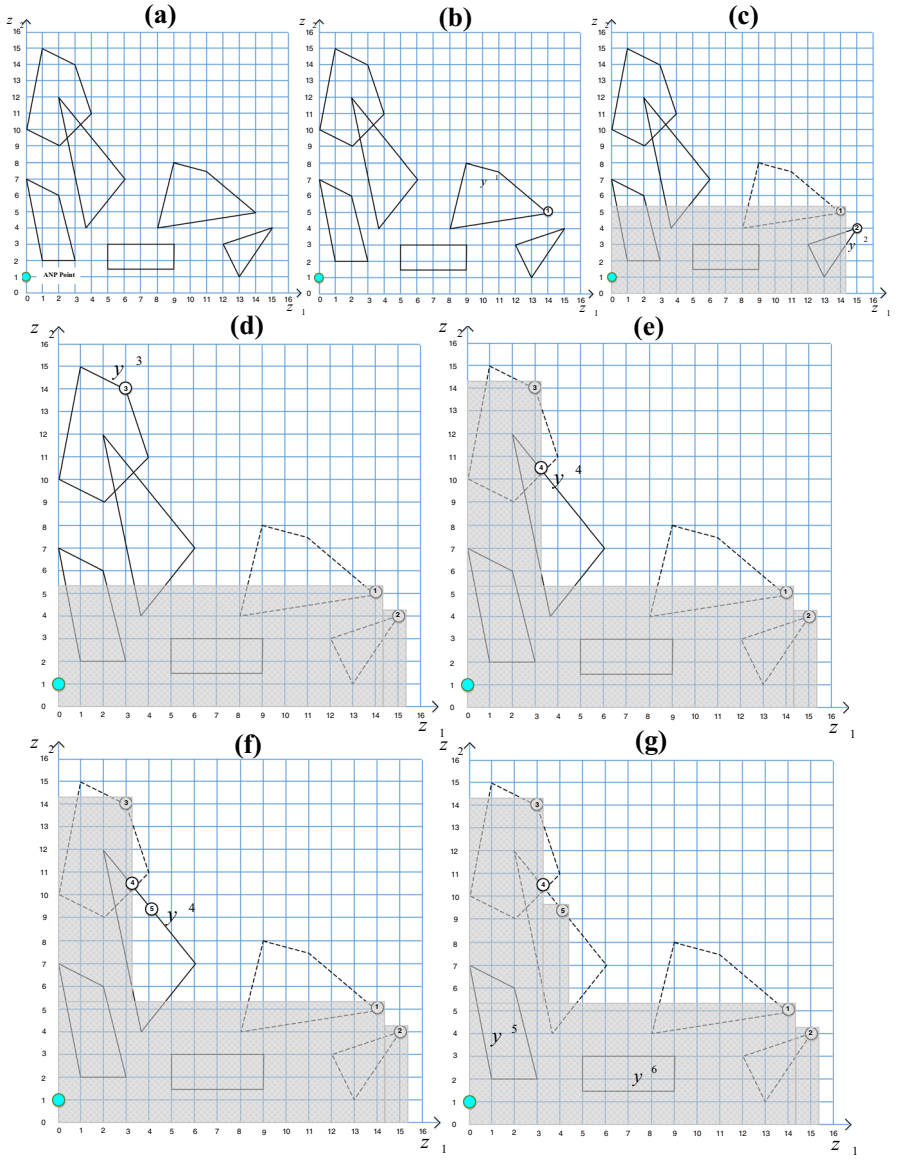

Fig. 2 An illustrative example for finding all efficient integer solutions in Step 1 (hatched regions are excluded due to *ExC* and dashed-lines denote the polyhedra excluded due to Y_{exc})

Optimal objective function value is 19 (14 + 5) and found at ① (Fig. 2b). Let ① be associated with integer solution y^1. ① is ND and hence, y^1 is an efficient integer solution. At the next iteration (Fig. 2c), $ExC := \{exd(①)\}$ and $Y_{exc} := \{y^1\}$. Solution of $\mathrm{DZ}(z^S, \{y^1\}, exd(①))$ happens at ② which is ND. Then, we update *ExC* and Y_{exc}. At the next iteration (Fig. 2d), we solve $\mathrm{DZ}(z^S, \{y^1, y^2\}, \{exd(①), exd(②)\})$. The result, ③, is ND. Hence, y^3 is an efficient

integer solution. We update Y_{exc} and ExC. At the next iteration (Fig. 2e), solving $\mathrm{DZ}(z^S, Y_{exc}, ExC)$ results in ④. ④ is dominated by some points in the polyhedron associated with y^3. Then, we check the efficiency of y^4 by using Algorithm 1 and find ⑤ that is ND (Fig. 2f). Then, we set $Y_{exc} := \{y^1, y^2, y^3, y^4\}$ and $ExC := \{exd(①), exd(②), exd(③), exd(⑤)\}$. At the next iteration (Fig. 2g), DZ is infeasible and hence there are no more efficient integer solutions.

Note that Algorithm 2 iterates four times and there are six integer solutions. Since we exclude some regions by ExC, the algorithm does not enumerate all integer solutions and performs effectively. For example, Fig. 2g shows that y^5 and y^6 are not in the set of excluded integer solutions. They are excluded due to ExC. In Proposition 1, we show that the provided process in Algorithm 2 generates all efficient integer solutions.

Proposition 1 *Let $\bar{y}$ be an arbitrary efficient integer solution for an instance of* (1)*, then Algorithm 2 identifies $\bar{y}$ as an efficient integer solution at one iteration.*

Proof Let $\bar{x} \in S(\bar{y})$ such that $\bar{z} := z(\bar{x}, \bar{y})$ is ND. If Algorithm 2 does not find a point such as $\bar{z}$, there are two possibilities:

1—$\bar{z}$ is excluded because of the constraints in ExC. ExC excludes the dominated cone of some ND points. Moreover, $\bar{z}$ is ND and is not in the dominated cone of other points. Then, the constraints of the set ExC do not exclude $\bar{z}$. Note that we assume that the value of δ in constraints exd is sufficiently small. Then, a ND point such as $\bar{z}$ is not excluded in the objective space due to a large value of δ.

2—$\bar{z}$ is excluded because $\bar{y} \in Y_{exc}$. We start with $Y_{exc} := \emptyset$. At each iteration, we add the integer solution of solving $\mathrm{DZ}(z^S, Y_{exc}, ExC)$ to Y_{exc}. Hence if $\bar{y} \in Y_{exc}$, then at an iteration where $\bar{y} \notin Y_{exc}$, $z(\hat{x}, \bar{y})$ such that $\hat{x} \in S(\bar{y})$ have been found by solving $\mathrm{DZ}(z^S, Y_{exc}, ExC)$. At that iteration, if $z(\hat{x}, \bar{y})$ is ND, then $\bar{y}$ have been found as an efficient integer solution. If $z(\hat{x}, \bar{y})$ is not ND, then we have identified $\bar{y}$ as an efficient integer solution by applying Algorithm 1.Then, $\bar{y} \in Y_{exc}$ cannot be a reason for not identifying $\bar{y}$ as an efficient integer solution. □

We remark that since our method finds all efficient integer solutions, it provides all integer solutions associated with a same ND point. For example, assume that $\bar{z}$ is ND and equal to $z(x^1, y^1)$ and $z(x^2, y^2)$ ($x^1 \in S(y^1)$ and $x^2 \in S(y^2)$). Then, our method finds both y^1 and y^2.

3.2 Step 2: solving the sub-MOLPs

In this step, integer variables are fixed. Then, we aim to solve the associated sub-MOLP. For this purpose, we exploit the multi-criteria simplex method (Yu and Zeleny 1975). Simplex method for single objective linear programs starts with a number of decision variables as the basic variables (x_B) which are non-zero. Then, at each iteration of a maximization problem, it selects a decision variable such as x_j (the current value is zero) from the non-basic variables (x_N) as the entering decision variable. The values of the reduced costs are the main selection criterion for the entering decision variable. Then, the entering variable will take a positive value in the next iteration. Let RC_j be the reduced cost of *jth* decision variable (x_j); the value

of RC_j shows that entering x_j into the basis will increase the objective function value with the slope of $-RC_j$. Hence, the decision variable with the most negative RC is selected as the entering decision variable. The leaving variable is selected using the minimum ratio test such that the feasibility is not violated in the next iteration. At each iteration, the current solution is adjacent with the solution of the previous iteration.

When we deal with more than one objective function, the RC_j changes to RC_{ij} as the reduced cost corresponds to the *ith* objective function and *jth* decision variable ($i = 1, \ldots, k$). Assume that we are interested in a tri-objective LP and the current basis results in a ND extreme point with values $z^{cr} = (z_1^{cr}, z_2^{cr}, z_3^{cr})$. The current efficient extreme solution has a number of adjacent solutions where some of them are efficient. By selecting different non-basic variables as the entering variable, these adjacent solutions are found. We can modify the selecting criteria such that the next adjacent solution is an efficient solution. For example, assume that x_1 is a non-basic variable and $RC_{i1} > 0$ for all $i = 1, 2, 3$. Therefore, selecting x_1 as the entering variable results in z^1 such that $z_i^1 < z_i^{cr}$ for all $i = 1, 2, 3$. Hence, selecting x_1 as the entering variable results in a dominated point. On the other hand, assume that x_2 is another non-basic variable and $(RC_{12}, RC_{22}, RC_{32}) = (2, -3, 0)$. Hence, entering x_2 to the basis results in z^2 such that $z_1^2 < z_1^{cr}$, $z_2^2 > z_2^{cr}$, and $z_3^2 = z_3^{cr}$. Then, if x^2 enters the basis, the new extreme solution will be potentially efficient.

Yu and Zeleny (1975) provide an algorithm to find all ND extreme points (*NDEP* set) and ND facets (*NDFC* set) of a MOLP. They also present a number of mathematical conditions for selecting the entering decision variable and identifying the extreme points which belong to one facet.

3.3 Step 3: finding the ND segments of each edge

We find the efficient integer solutions in Step 1 and *NDEP*/*NDFC* sets corresponding to the efficient integer solutions in Step 2. The edges of polyhedra in the objective space are the line segments between pairs of adjacent ND extreme points. In this section, we provide a method to find the ND segments of these edges.

Figure 3 shows an illustrative example in the objective space for a maximization BOMILP instance; five polyhedra correspond to five integer solutions $(y^1, \ldots, y^5)$ exist. In this example, we are interested in finding the ND segments on the edge between ① and ②. Note that ① and ② are two adjacent ND extreme points of sub-MOLP(y^1). Moreover, let ④ $= \lambda$① $+ (1 - \lambda)$② for a $\lambda \in [0, 1]$. Then, ④$^{-\epsilon}$ denotes point $(\lambda - \epsilon)$① $+ (1 - \lambda + \epsilon)$② in the objective space where ϵ is a sufficiently small positive number. We use this notation in the description of the illustrative example provided in Fig. 3.

We solve DZ(①, $\{y^1\}, \emptyset$) to check the dominance of ①. ① is dominated and solving DZ results in some point(s) such as $z(x, y^2)$ where $x \in S(y^2)$. Then, we aim to find a segment in the convex hull of ① and ② (edge [①, ②]) starting from ① which is dominated by the points in $\{z(x, y^2) | x \in S(y^2)\}$. The linear program given in (4) results in an optimal value of α that gives ③ if $z' :=$ ①, $z'' :=$ ⓨ2, and $\bar{y} := y^2$. Let

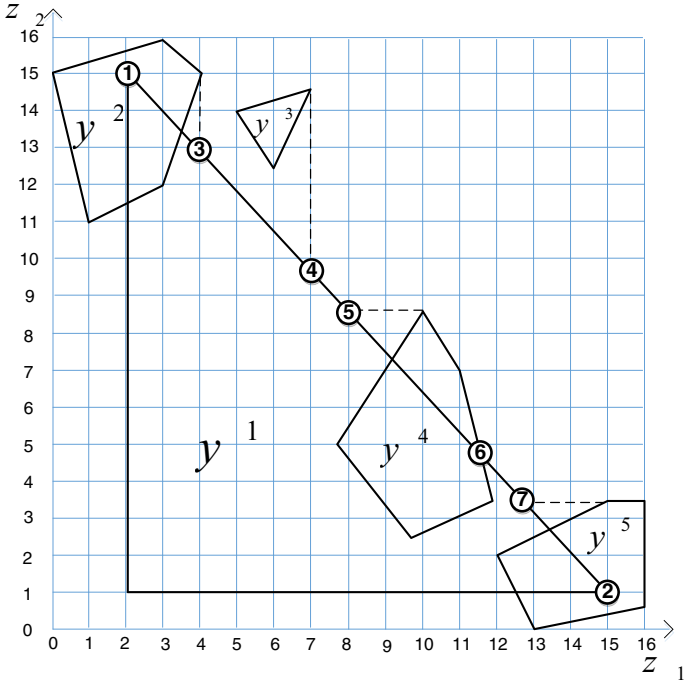

Fig. 3 An illustrative example for finding the ND segments of the edge $[①, ②]$ in Step 3

α^E be the optimal value of α. Then, $③ = \alpha^E① + (1-\alpha^E)②$. Due to Proposition 2, $[①, ③)$ is a dominated segment.

$$\begin{aligned} &\text{EDG1} \quad (z', z'', \bar{y}): \\ &\quad \min \quad \alpha \\ &\quad \text{s.t.} \quad x \in S(\bar{y}), \\ &\qquad z_i(x, \bar{y}) \geq \alpha z'_i + (1-\alpha) z''_i, i = 1, \ldots, k, \\ &\qquad 0 \leq \alpha \leq 1. \end{aligned} \tag{4}$$

Proposition 2 *Let z' and z'' be two adjacent ND extreme points. z' is dominated by some $z(x, \bar{y})$ such that $x \in S(\bar{y})$. Moreover, let α^E be the optimal value of α in solving (4). Then, segment $[z', \alpha^E z' + (1-\alpha^E) z'')$ is dominated.*

Proof Assume that $z(x', \bar{y})$ dominates z' and $z_i(x^\alpha, \bar{y}) \geq \alpha^E z'_i + (1-\alpha^E) z''_i$, for all $i = 1, \ldots, k$ $(x', x^\alpha \in S(\bar{y}))$. Let $\hat{z}$ be an arbitrary point in segment $[z', \alpha^E z' + (1-\alpha^E) z'')$ such that $\hat{z} = \lambda z' + (1-\lambda)\left(\alpha^E z' + (1-\alpha^E) z''\right)$ and $\lambda \in (0, 1]$. Moreover, let $\bar{z} := z(\lambda x' + (1-\lambda) x^\alpha, \bar{y}) = \lambda z(x', \bar{y}) + (1-\lambda) z(x^\alpha, \bar{y})$.[4] Then, $\bar{z}$ dominates $\hat{z}$

[4] Note that $\lambda x' + (1-\lambda) x^\alpha \in S(\bar{y})$ since $S(\bar{y})$ is a convex set.

because $\lambda z_i(x', \bar{y}) + (1-\lambda) z_i(x^{\alpha}, \bar{y}) \geq \lambda z'_i + (1-\lambda)\left(\alpha^E z'_i + (1-\alpha^E) z''_i\right)$ for all $i = 1, \ldots, k$, and $\bar{z} \neq \hat{z}$. □

Next, we calculate DZ($③^{-\epsilon}, \{y^1\}, \emptyset$). $③^{-\epsilon}$ is dominated by some points in the polyhedron associated with y^3. Then, we solve EDG1($③^{-\epsilon}, ②, y^3$) that results in an optimal value of α which gives point ④. Hence, segment $[③, ④)$ is dominated. Note that ④ is dominated and $④^{-\epsilon}$ is ND (④ is a weakly ND point). Then, we aim to find a segment in the convex hull of $④^{-\epsilon}$ and ② starting from $④^{-\epsilon}$ that is ND. The MILP given in (5) results in an optimal value of α that gives ⑤ if $z' := ④^{-\epsilon}$, $z'' := ②$, and $Y_{exc} := \{y^1\}$. The optimal value of α (α^E) is a value such that $⑤ = \alpha^E ④^{-\epsilon} + (1-\alpha^E)②$. Due to Proposition 3, $[④^{-\epsilon}, ⑤)$ is a ND segment.

$$\begin{aligned} &\text{EDG2} \quad (z', z'', Y_{exc}): \\ &\quad \max \quad \alpha \\ &\quad \text{s.t.} x \in S(y), y \in \mathbb{Z}^q \backslash Y_{exc}, \\ &\qquad z_i(x,y) \geq \alpha z'_i + (1-\alpha) z''_i, i = 1, \ldots, k, \\ &\qquad 0 \leq \alpha \leq 1. \end{aligned} \tag{5}$$

Proposition 3 *Let z' and z'' be two adjacent ND extreme points and z' is ND. Let z' and z'' correspond to integer solution y'. Assume that $Y_{exc} := \{y'\}$. Moreover, let α^E be the optimal value of α in solving (5). Then, segment $[z', \alpha^E z' + (1-\alpha^E) z'')$ is ND.*

Proof Assume to the contrary that a $\lambda \in (0, 1]$ exists such that $\lambda z' + (1-\lambda)\left(\alpha^E z' + (1-\alpha^E) z''\right)$ is dominated. Then, $(\lambda + \alpha^E) z' + (1-\lambda)(1-\alpha^E) z''$ is dominated. We conclude that optimal value of α in EDG2 given in (5) is greater than or equal to $\lambda + \alpha^E$ that contradicts the optimality of α^E. □

In the following lemma, we explain the reason behind excluding y' in Proposition 3.

Lemma 1 *Under the conditions described in Proposition 3, if we do not exclude y', then EDG2 $(z', z'', \emptyset)$ results in $\alpha^E = 1$.*

For finding the ND segments in segment $[⑤, ②]$, note that ⑤ and $⑤^{-\epsilon}$ are dominated (⑤ is a weakly ND point). Similar to the discussed procedure for finding a ND segment starting from ①, we again solve EDG1($⑤^{-\epsilon}, ②, y^4$) to find ⑥. ⑥ and $⑥^{-\epsilon}$ is ND. Then, we solve EDG2 ($⑥^{-\epsilon}, ②, \{y^1\}$), which results in ⑦. We identify $[⑥, ⑦)$ as a ND segment. Then, we identify that ⑦ and $⑦^{-\epsilon}$ are dominated. Hence, we solve EDG1($⑦^{-\epsilon}, ②, y^5$), which results in $\alpha^E = 0$ and shows that there is no ND segment in $[⑦^{-\epsilon}, ②]$. Then, we are finished with finding the ND segments in the edge $[①, ②]$. Segments $(④, ⑤)$ and $[⑥, ⑦)$ are the ND segments.

We show all steps discussed for solving example in Fig. 3 in an algorithmic presentation in Algorithm 3. This algorithm shows the process for finding the ND segments of the edge $[z', z'']$ associated with $\tilde{y}$. Note that if z' is ND and EDG2($z', z'', \{\tilde{y}\}$) is infeasible, then there is no $z(x, y)$ such that $x \in S(y)$,

$y \in \mathbb{Z}^q \backslash \{\tilde{y}\}$, and $z_i(x, y) \geq \alpha z'_i + (1 - \alpha) z''_i$ for all $i = 1, \ldots, k$. Hence, $[z', z'']$ will be a ND segment.

Algorithm 3 Find the ND segments of the edge $[z', z'']$ associated with $\tilde{y}$

$z^\alpha := z'$
while $z^\alpha \neq z''$ **do**
  Solve DZ$(z', \{\tilde{y}\}, \emptyset) \rightarrow z^D := z(x^D, y^D)$
  if z' is ND **then**
    Solve EDG2$(z', z'', \{\tilde{y}\}) \rightarrow \alpha^E, z^\alpha := \alpha^E z' + (1 - \alpha^E) z''$
    if EDG2 is feasible **then**
      $[z', z^\alpha)$ is a ND segment
      Check the dominance of z^α solving DZ$(z^\alpha, \{\tilde{y}\}, \emptyset)$.
      $z' := (\alpha^E - \epsilon) z' + (1 - \alpha^E + \epsilon) z''$*
    else
      $[z', z'']$ is a ND segment. Exit the loop.
  else
    Solve EDG1$(z', z'', y^D) \rightarrow \alpha^E, z^\alpha := \alpha^E z' + (1 - \alpha^E) z''$
    $[z', z^\alpha)$ is a dominated segment.
    Check the dominance of z^α solving DZ$(z^\alpha, \{\tilde{y}\}, \emptyset)$.
    if $\alpha \neq 0$ **then** $z' := (\alpha^E - \epsilon) z' + (1 - \alpha^E + \epsilon) z''$

* ϵ is a sufficiently small positive number.

We can use some information obtained in Step 3 for generating efficient integer solutions. We mention this issue in the following remark.

Remark 2 While finding dominated/ND segments of the edges in Step 3, we find some ND points that dominate some segments of the edges. These points are associated with some efficient integer solutions which might not be found due to the value of δ in *ExC* constraints. Hence, we can use this information and identify them as efficient integer solutions.

3.4 Step 4: identifying the ND facets

The ND points set of a MOMILP includes the ND points in the form of k'-dimensional facets ($0 \leq k' \leq k - 1$). Hence if $k \geq 3$, we cannot address the entire ND points by showing points and line segments only. In Step 2, we generate the ND facets of the sub-MOLPs while we have not identified if they are ND for the MOMILP. Then, in this section, we aim to characterize the entire ND facets and filter out the facets which are completely dominated using the following:

FC1 Information obtained from Step 3 (ND points in the form of points and line segments).
FC2 Using excluding constraints iteratively to find a ND point in a facet.

Assume that we are interested in finding the ND points associated with $\bar{y}$. Let F be a ND facet of sub-MOLP$(\bar{y})$ in the objective space ($F \in NDFC_{\bar{y}}$). We characterize F by its corners that are ND extreme points. Let z^j, $j = 1, \ldots, n_F$, be the extreme points of F. Then, F is $ConvexHull\{z^1, \ldots, z^{n_F}\}$. Note that per extreme point z^j, $j = 1, \ldots, n_F$, there is at least one adjacent extreme point in $\{z^p, p \in \{1, \ldots, n_F\}, p \neq j\}$. In Sects. 3.4.1 and 3.4.2, we show the processes to check the dominance of facet F.

3.4.1 FC1

Assume that we aim to find 2-dimensional ND facets of the instance associated with Fig. 1. Facet Ⓒ−Ⓕ−Ⓖ−Ⓗ is generated in Step 2. We can supply this facet as a ND facet if Ⓒ, Ⓕ, Ⓖ, *or* Ⓗ is ND—although it may be a partially ND facet. Now, assume that all of these points are dominated and facet Ⓒ−Ⓕ−Ⓖ−Ⓗ has a ND segment in its edges. Again we can claim that this facet is ND.

In general, we identify all ND extreme points and the ND segments of the edges in Sect. 3.3 (Step 3). We admit that facet F is ND if at least one z^j, $j = 1, \ldots, n_F$, is ND. If z^j is dominated for all $j = 1, \ldots, n_F$, then we look for a ND segment in the edges between adjacent ND extreme points of F. If at least one ND segment exists, then F is a ND facet.

Let z^j be dominated for all $j = 1, \ldots, n_F$. Moreover, assume that there is no ND segment in the edges between pairs of adjacent ND extreme points associated with F. Then, we may identify F as a dominated facet using Proposition 4.

Proposition 4 *Let all extreme points of facet F be dominated by the points that are associated with the same integer solution* $(\hat{y})$*. Then, F is completely dominated.*

Proof Let $z(x^j, \hat{y})$ dominate z^j and $x^j \in S(\hat{y})$ for all $j = 1, \ldots, n_F$. Moreover, assume that $\bar{z} = \sum_{j=1}^{n_F} \lambda_j z^j$, $\sum_{j=1}^{n_F} \lambda_j = 1$, and $\lambda_j \in [0, 1]$ for all $j = 1, \ldots, n_F$ $(\bar{z} \in F)$. Then, $\hat{z} = \sum_{j=1}^{n_F} \lambda_j z(x^j, \hat{y})$ dominates $\bar{z}$ since $\lambda_j z_i(x^j, \hat{y}) \geq \lambda_j z_i^j$ for all $i = 1, \ldots, k$, and $j = 1, \ldots, n_F$. Note that at least for one i and j, $\lambda_j z_i(x^j, \hat{y}) > \lambda_j z_i^j$. Moreover, $\hat{z}$ is the convex combination of $z(x^j, \hat{y})$, $j = 1, \ldots, n_F$. Then, $x \in S(\hat{y})$ exists such that $z(x, \hat{y}) = \hat{z}$. □

3.4.2 FC2

Assume that we cannot identify the dominance of F using the discussed methods in Sect. 3.4.1. In Sect. 3.1.1, we examine the efficiency of an integer solution $(\bar{y})$ using excluding constraints iteratively. We find a ND point such as $z(x, \bar{y})$ such that $x \in S(\bar{y})$. In the current section, we present an algorithm similar to Algorithm 1 to find a ND point such as $z(x, \bar{y})$ that is in F. Therefore, $z(x, \bar{y})$ is in the convex hull of z^1, …, and z^{n_F} in the objective space. We provide a MILP in (6) for finding a potentially ND point in F. The solution of the formulation given in (6) results in points which are in the convex hull of F's corners. Hence, these points will be in the facet F. Moreover, the set of constraints in *ExC* guarantees that the point which is the result of DF is not in the dominated cone of previously found ND points.

$$
\begin{aligned}
&\mathrm{DF}(z^1,\ldots,z^{n_F},\bar{y},ExC):\\
&\quad \max \quad \sum_{i=1}^{k}\epsilon_i\\
&\quad \text{s.t.} \quad z_i(x,\bar{y})-\epsilon_i \geq z_i^S, i=1,\ldots,k,\\
&\qquad\quad z_i(x,\bar{y})=\sum_{j=1}^{n_F}\lambda_j z_i^j, i=1,\ldots,k,\ \sum_{j=1}^{n_F}\lambda_j=1,\\
&\qquad\quad 0\leq\lambda_j\leq 1, j=1,\ldots,n_F,\\
&\qquad\quad x\in S(\bar{y}), ExC,\\
&\qquad\quad \epsilon_i\geq 0, i=1,\ldots,k.
\end{aligned}
\tag{6}
$$

Let $ExC := \emptyset$. Then, the feasible region of (6) is $x \in S(\bar{y})$ such that their images onto the objective space are in $ConvexHull\{z^1,\ldots,z^{n_F}\}$. Algorithm 4 works similar to Algorithm 1. It finds a point in F that is not dominated by previously found ND points and is ND.

Algorithm 4 Check the dominance of $F := ConvexHull\{z^1,\ldots,z^{n_F}\}$ associated with $\bar{y}$

$ExC = \emptyset$
$\mathrm{DF}(z^1,\ldots,z^{n_F},\bar{y},ExC) \rightarrow \lambda^D := (\lambda_1^D,\ldots,\lambda_{n_F}^D), z^D := \sum_{j=1}^{n_F}\lambda_j^D z^j$
if DF is infeasible **then**
    F is a dominated facet.
else
    if z^D is ND **then**
        F is a ND facet.
    else
        $\mathrm{DZ}(z^D,\{\bar{y}\},\emptyset) \rightarrow z^{D1} := z(x^{D1},y^{D1})$
        $ExC := ExC \cup exd(z^{D1})$
        go to line 2

4 Numerical experiments

We illustrate the effectiveness of the GoNDEF on a set of MOMILP instances. These instances are generated similar to the instances used by Boland et al. (2015) and Mavrotas and Diakoulaki (2005). The mathematical formulation of the instances and the range of parameters are provided in Appendix 2. We implement our algorithm in MATLAB R2017b and ILOG CPLEX 12.5 optimizer using a PC with Pentium IV processor at 3.00 GHz and with 32.0 GB of RAM.

We classify our instance problems based on the number of objective functions (k), the number of constraints (m), the number of continuous variables (n), and the number of integer variables (q). In this section, instance size denotes m, n, and q for short. We also randomly generate five different instances per each problem and report average values. Note that we also indicate the number of solved LPs and MILPs (# LP and # MILP), the number of the efficient integer solutions (# eff. int.

Table 1 Solving MOMILP instances with binary variables using Algorithm 2 for finding the efficient integer solutions ($\Delta = 0.1$)

k	m	n	q	# MILP	# eff. int. sol.	CPUT (sec.)	# MILP per eff. int. sol.	CPUT per eff. int. sol. (sec.)
2	10	5	5	10	1.6	0.10	6.25	0.06
2	20	10	10	41.8	7.2	1.37	5.81	0.19
2	30	15	15	52.4	6.8	1.84	7.71	0.27
2	40	20	20	109	13.8	4.05	7.90	0.29
2	50	25	25	261.6	19.6	18.41	13.35	0.94
3	10	5	5	15.4	3.6	0.19	4.28	0.05
3	20	10	10	105.6	26.2	4.84	4.03	0.18
3	30	15	15	379.6	47.4	34.93	8.01	0.74
3	40	20	20	830.4	55	181.19	15.10	3.29
3	50	25	25	3459.8	151.4	5603.94*	22.85	37.01
4	10	5	5	16	4.2	0.19	3.81	0.04
4	20	10	10	184.8	36.6	11.32	5.05	0.31
4	30	15	15	995.6	125.6	492.79	7.93	3.92
4	40	20	20	1987.4	193.4	2554.68	10.28	13.21
4	50	25	25	3470.8	549.8	7207.98*	6.31	13.11

*There are instances that took more than 2 h

sol.),[5] and total CPU time in seconds (CPUT (sec.)). Since the complexity of the problem increases by instance size, we show the average cost of finding an efficient integer solution. We report the average number of solved MILPs (# MILP per eff. int. sol.) and the average CPU time that is consumed for finding an efficient integer solution (CPUT per eff. int. sol. (sec.)). Then, the number of MILPs per efficient integer solution and CPU time per efficient integer solution refer to "# MILP per eff. int. sol." and "CPUT per eff. int. sol. (sec.)", respectively.

The value of δ in the excluding constraints of the set ExC modifies excluding regions (it is discussed in Sect. 3.1); the larger values of δ result in the larger excluding regions. However, the probability of missing an efficient integer solution increases. We test our method with δ_i for the ith objective function where $i = 1, .., k$. Let δ_i be equal to the multiplication of Δ and the range of ith objective function. In the instances that we used, the average value of δ_i for $i = 1, \ldots, 4$, is $(\delta_1, \delta_2, \delta_3, \delta_4) = (1.42, 1.36, 1.36, 1.34)$ if $\Delta = 0.001$.

Tables 1, 2, and 3 show the numerical results for MOMILP instances to find the efficient integer solutions with $\Delta = 0.1$, $\Delta = 0.01$, and $\Delta = 0.001$, respectively. In these tables, we solve instances for finding the efficient integer solutions without solving the sub-MOLPs. Hence, the number of ND segments and facets are not reported.

[5] This number is the computed number of the efficient integer solutions for the instances terminated due to the time limit.

Table 2 Solving MOMILP instances with binary variables using Algorithm 2 for finding the efficient integer solutions ($\Delta = 0.01$)

k	m	n	q	# MILP	# eff. int. sol.	CPUT (sec.)	# MILP per eff. int. sol.	CPUT per eff. int. sol. (sec.)
2	10	5	5	34	2	0.42	17.00	0.21
2	20	10	10	48.8	9.6	1.53	5.08	0.16
2	30	15	15	60.8	8.2	2.15	7.41	0.26
2	40	20	20	119	15.8	5.17	7.53	0.33
2	50	25	25	256.2	20.8	22.34	12.32	1.07
3	10	5	5	61.2	3.8	0.84	16.11	0.22
3	20	10	10	190.6	32.2	7.19	5.92	0.22
3	30	15	15	496.2	57.2	41.25	8.67	0.72
3	40	20	20	1112.8	80.2	144.62	13.88	1.80
3	50	25	25	4395.4	221.6	5642.99*	19.83	25.46
4	10	5	5	35.8	4.4	0.55	8.14	0.13
4	20	10	10	419.6	46.8	17.58	8.97	0.38
4	30	15	15	1759	164	590.00	10.73	3.60
4	40	20	20	3164.6	254	2981.48	12.46	11.74
4	50	25	25	6015.8	629.8	7210.12*	9.55	11.45

*There are instances that took more than 2 h

Table 3 Solving MOMILP instances with binary variables using Algorithm 2 for finding the efficient integer solutions ($\Delta = 0.001$)

k	m	n	q	# MILP	# eff. int. sol.	CPUT (sec.)	# MILP per eff. int. sol.	CPUT per eff. int. sol. (sec.)
2	10	5	5	264.4	2	2.77	132.20	1.38
2	20	10	10	195.6	10.2	3.65	19.18	0.36
2	30	15	15	244	9.6	4.83	25.42	0.50
2	40	20	20	317.4	18.6	9.06	17.06	0.49
2	50	25	25	479.6	30	20.72	15.99	0.69
3	10	5	5	516.4	3.8	7.47	135.89	1.96
3	20	10	10	916.4	32.8	21.10	27.94	0.64
3	30	15	15	2292.2	66.6	88.82	34.42	1.33
3	40	20	20	6537	92	314.90	71.05	3.42
3	50	25	25	21005.8	263.8	6009.70*	79.63	22.78
4	10	5	5	200.8	4.6	3.58	43.65	0.78
4	20	10	10	2651	48.2	62.02	55.00	1.29
4	30	15	15	10289.6	181.4	899.01	56.72	4.96
4	40	20	20	18106.6	281.6	3793.56	64.30	13.47
4	50	25	25	32365.6	670.6	7208.62*	48.26	10.75

*There are instances that took more than 2 h

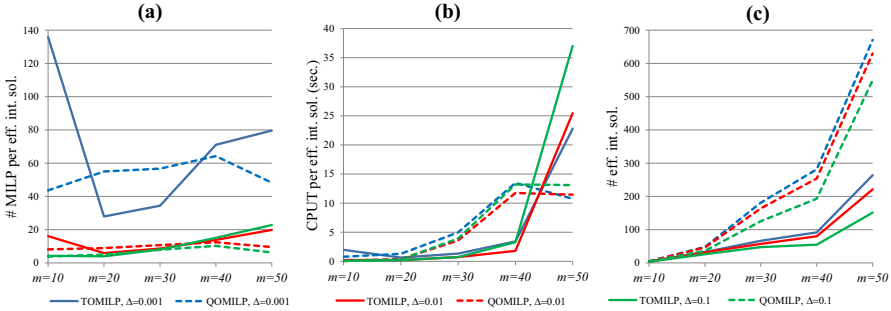

Fig. 4 The results of the GoNDEF performance by changing Δ, the number of objective functions, and instance size

We aim to show the impact of changing the number of objective functions and instance size on the performance of the GoNDEF for finding the efficient integer solutions. First, note that if we consider the results of Tables 1, 2, and 3, the number of the efficient integer solutions increases faster than the number of integer variables. Second, there is a large increase in the number of the efficient integer solutions when a new objective function is added to an instance. Hence, our method requires more CPU time and solves more MILPs to generate more efficient integer solutions for larger instances with larger number of objective functions. In terms of the number of MILPs per efficient integer solution, we show that finding an efficient integer solution is not significantly costlier when the number of objective functions or instance size increases. For example, in some instances, the number of MILPs per efficient integer solution decreases by the number of objective functions or instance size. Then, we conclude that our method shows a reasonable performance on the provided set of instances regarding the number of MILPs per efficient integer solution. Note that our method does not show a similar performance in terms of CPU time per efficient integer solution.

In Fig. 4, we summarize the results of solving TOMILP and quad-objective MILP (QOMILP) instances shown in Tables 1, 2, and 3. We compare the performance of our method with different values of Δ. In this figure, horizontal axes denote different instance sizes. Moreover, vertical axes denote the number of MILPs per efficient integer solution, CPU time per efficient integer solution, and the number of the efficient integer solutions in Fig. 4a–c, respectively. We show the results associate TOMILP instances by continuous lines and the results of QOMILP instances by dashed lines. In addition, blue, red, and green colors associate $\Delta = 0.001$, $\Delta = 0.01$, and $\Delta = 0.1$, respectively. Note that the larger values of # eff. int. sol. denote a better performance of the GoNDEF with a specific value of Δ. However, the smaller values of # MILP per eff. int. sol. and # CPUT per eff. int. sol. indicate a better performance.

Figure 4a shows that the GoNDEF with $\Delta = 0.01$ and $\Delta = 0.1$ solves TOMILP and QOMILP instances significantly better than the GoNDEF with $\Delta = 0.001$. In terms of CPU time per efficient integer solution, Fig. 4b shows that in solving TOMILP and QOMILP instances with $m = 10$ and $m = 20$, the GoNDEF with $\Delta = 0.001$, $\Delta = 0.01$, and $\Delta = 0.1$ have similar performances. However, the

GoNDEF with $\Delta = 0.1$ performs significantly better than the GoNDEF with $\Delta = 0.01$ and $\Delta = 0.001$ in solving TOMILP and QOMILP instances with $m = 40$ and $m = 50$. On the other hand, in Fig. 4c, we show that the GoNDEF with $\Delta = 0.1$ performs worse than the GoNDEF with $\Delta = 0.01$ and $\Delta = 0.001$ in terms of the number of efficient integer solutions. Hence, we decide to use $\Delta = 0.01$ for the rest of numerical experiments. It provides a fair analysis for finding efficient integer solutions and the ND points set.

We summarize the issues mentioned in Fig. 4, in the following items:

- When the number of objective functions increases, the complexity of the problem and solution time significantly increase.
- For a real MOMILP, a good value for Δ can be decided based on the structure of the problem, the size of the problem, and what we aim to provide for the owner of the problem.
- In order to find the best value for Δ, testing a large number of values from 0.0001 to 0.3 (this range may change regarding the problem), deeply analyzing the trade-offs between the outputs, and considering time performances are helpful.
- Very small values for Δ result in a low time performance; however, they do not significantly increase the number of outputs. Hence, $\Delta \in [0.01, 0.05]$ can be reasonable.

Let U be the upper bound of integer variables ($y_j \leq U$ for all $j = 1, \ldots, q$). In Table 4, we test the GoNDEF for solving MOMILP instances with $U = 2$ and $\Delta = 0.01$. We compare the results of Table 4 with Table 2 in Table 5. When we change $U = 1$ to $U = 2$, two main issues appear that increase the complexity of problem: 1—a larger feasible region due to more integer solutions, and 2—necessity of using no-good constraints for integer variables that are more complex than no-good constraints for binary variables.

Table 5 shows the percentage of changes in the number of integer solutions (% change in # of integer solutions), the number of solved MILPs (% change in # MILP), and total CPU time (% change in CPUT) when $U = 1$ changes to $U = 2$. Note that we provide the changes for the classes in which all instances are completely solved. Regarding this table, the increases in the number of solved MILPs are smaller than the increases in the number of integer solutions. However, changes are significantly larger in terms of CPU time for solving TOMILP and QOMILP instances (not BOMILP instances).

In Table 6, we report the results of solving MOMILP instances with binary variables and $\Delta = 0.01$. Hence, we report the number of ND segments (# ND segments) and the number of ND facets (# ND facets). Note that the number of ND segments refers to the sum of separate single points and line segments. The last four columns show the performance of the GoNDEF in terms of CPU time. We report the percentage of CPU time that is consumed for solving the sub-MOLPs (% MOLP CPUT). Moreover, in the last two columns, we show the spent CPU time for finding a ND facet averagely. Column 12 shows the average CPU time for finding a ND

Table 4 Solving MOMILP instances with integer variables ($U = 2$) using Algorithm 2 for finding the efficient integer solutions ($\Delta = 0.01$)

k	m	n	q	# MILP	# eff. int. sol.	CPUT (sec.)	# MILP per eff. int. sol.	CPUT per eff. int. sol. (sec.)
2	10	5	5	34	2	0.37	17.00	0.18
2	20	10	10	58.8	9	1.78	6.53	0.20
2	30	15	15	59.2	9.8	2.75	6.04	0.28
2	40	20	20	165.4	19.6	10.28	8.44	0.52
2	50	25	25	520.2	26.6	91.98	19.56	3.46
3	10	5	5	61.2	3.8	0.88	16.11	0.23
3	20	10	10	218.4	36.8	11.57	5.93	0.31
3	30	15	15	560.8	79.4	203.74	7.06	2.57
3	40	20	20	1321.8	107.8	577.44	12.26	5.36
3	50	25	25	3646.4	255	6991.34*	14.30	27.42
4	10	5	5	35.8	4.4	0.52	8.14	0.12
4	20	10	10	536.6	62.2	30.83	8.63	0.50
4	30	15	15	2269.2	238.8	2470.97	9.50	10.35
4	40	20	20	3371.4	359	7216.00*	9.39	20.10
4	50	25	25	4286	652.4	7215.94*	6.57	11.06

*There are instances that took more than 2 h

Table 5 Comparison between Tables 2 ($U = 1$) and 4 ($U = 2$)

k	m	n	q	% change in # of integer solutions	% change in # MILP	% change in CPUT
2	10	5	5	0.0	0.0	− 12.0
2	20	10	10	56.8	20.5	16.1
2	30	15	15	172.3	− 2.6	27.6
2	40	20	20	222.5	39.0	98.8
2	50	25	25	476.8	103.0	311.7
3	10	5	5	0.0	0.0	5.2
3	20	10	10	56.8	14.6	60.9
3	30	15	15	172.3	13.0	394.0
3	40	20	20	222.5	18.8	299.3
3	50	25	25	476.8	–	–
4	10	5	5	0.0	0.0	− 5.7
4	20	10	10	56.8	27.9	75.3
4	30	15	15	172.3	29.0	318.8
4	40	20	20	222.5	–	–
4	50	25	25	476.8	–	–

facet and column 13 shows the average CPU time without considering the consumed CPU time for solving the sub-MOLPs for finding a ND facet.

Table 6 shows that there is a large difference between BOMILPs and MOMILPs (with $k \geq 3$) in terms of complexity. Moreover, by increasing instance size, the

Table 6 Solving MOMILP instances with binary variables by the GoNDEF ($\Delta = 0.01$)

k	m	n	q	# LP	# MILP	# eff. int. *sol.***	# ND segments	# ND facets	% MOLP CPUT	CPUT (sec.)	CPUT per facet	CPUT (no MOLP) per facet
2	10	5	5	17.2	45.6	2	4.4	0	9.67	0.52	–	–
2	20	10	10	251	160.2	9.8	35.2	0	9.66	4.24	–	–
2	30	15	15	407.8	215	8.6	61.2	0	10.89	6.30	–	–
2	40	20	20	1698.2	649.2	17.2	179	0	12.31	24.10	–	–
2	50	25	25	6286.6	1824.6	29.2	459.6	0	12.65	97.37	–	–
3	10	5	5	108.8	122.6	3.8	32.2	11.6	13.31	1.58	0.136	0.118
3	20	10	10	4307.4	1622.4	32.2	751.8	277.2	12.67	42.40	0.153	0.134
3	30	15	15	21036.4	6166.8	58.6	2949.4	1188.4	13.02	226.42	0.191	0.166
3	40	20	20	133484.8	22780.2	84.8	9953.2	4720	21.16	1125.20	0.238	0.188
3	50	25	25	1009587.6	146867.4	292.4	63347	29885.6	14.30	30170.18*	1.010	0.865
4	10	5	5	188.6	142	4.4	56.6	9.8	17.27	2.14	0.218	0.181
4	20	10	10	49740.4	6541.2	47.8	3263.6	1049	29.69	223.05	0.213	0.150
4	30	15	15	1200758.6	69665	166.8	37616	18814.2	53.78	7916.96	0.421	0.194
4	40	20	20	2506057.2	131860.2	146.2	80073.6	45340.2	72.99	32152.60*	0.709	0.192
4	50	25	25	1604889.4	69674.6	36.6	47054.8	32005.6	89.07	38279.40*	1.196	0.131

*There are instances that took more than 10 h

**The small differences in this column comparing with Table 2 are due to Remark 2

number of ND segments and facets significantly increase. Regarding solving the sub-MOLPs, in Sects. 3 and 3.1, we describe that solving them is a time-consuming part of our method (e.g., when $k = 4$ and $m = 40/50$, 73%/89% of the total CPU time are consumed for solving the sub-MOLPs). Therefore, we compare the results of the columns 12 and 13 to discuss the impact of solving the sub-MOLPs on CPU time per ND facet. Although the average CPU time for finding a ND facet without considering the solution time of the sub-MOLPs fairly increases by instance size, this increase is significantly smaller than the increase in column 12. Then, CPU time for solving the sub-MOLPs significantly affects the average CPU time for finding a ND facet.

5 Conclusions

In this paper, we present the GoNDEF method to find all ND points of a general MOMILP. Our method presents the ND points in the objective space in the form of ND facets. The dimensions of these facets vary from 0 to $k - 1$. In order to provide a more clear and practical representation of the partially ND facets, the GoNDEF identifies the ND segments of the edges between pairs of adjacent ND extreme points.

The innovative characteristics of the GoNDEF are highly efficient and practical for solving MOMILPs. By choosing appropriate values for the ANP point, we can simply generate the ND points such that their associated objective function values are in some specific ranges. Moreover, we can modify the solution method of sub-MOLPs to generate efficient solutions of MOMILPs in the decision space. Note that integer/binary variables are more important than continuous variables in terms of managerial issues; our method can generate all efficient integer solutions. This characteristic allows us to generate all efficient integer solutions significantly faster than generating all ND points. The computational results from solving a set of instance problems indicate that the GoNDEF generates the entire set or a large subset of the ND points.

Finally, any progress in solving MOLPs improves the performance of the GoNDEF significantly. Moreover, a MOMILP includes a large number of the ND facets. Focusing on the solutions that are more interesting for decision makers is a crucial topic for future study.

Acknowledgements Financial support for this work by TUPRAS under grant OS.00054 is gratefully acknowledged. MT gratefully acknowledges the computational infrastructure support provided by the IBM Corporation through the IBM SUR award. The authors acknowledge valuable comments and suggestions provided by Emre Alper Yıldırım, Emre Mengi, Seyed Mojtaba Hosseini, Ali Fattahi, Matthias Ehrgott, and referees of Optimization and Engineering journal.

Appendix 1: The formulation of our illustrative instance given in Fig. 1

We provide a mathematical formulation for Fig. 1. Note that there may be other formulations to provide a feasible region corresponding to Fig. 1. Let M be a sufficiently large positive number, $x = (x_1, x_2, x_3) \geq 0$, and $y = (y_1, y_2, y_3) \in \{0, 1\}^3$.

$$\begin{aligned} &\max \quad z(x, y) = (x_1, x_2, x_3) \\ &\text{s.t.} \end{aligned}$$

$$\begin{aligned} &x_1 \leq 6 + M(1 - y_1), \\ &x_2 \leq 6 + M(1 - y_1), \\ &7 - M(1 - y_1) \leq x_1 + x_2 \leq 9 + M(1 - y_1), \\ &4 - M(1 - y_1) \leq x_3 \leq 10 + M(1 - y_1), \end{aligned}$$

$$\begin{aligned} &-6 - M(1 - y_2) \leq x_1 - x_2 \leq 8 + M(1 - y_2), \\ &8 - M(1 - y_2) \leq x_1 + x_2, \\ &3 - M(1 - y_2) \leq x_3, \\ &3x_1 + 3x_2 + 2x_3 \leq 42 + M(1 - y_2), \end{aligned}$$

$$\begin{aligned} &4 - M(1 - y_3) \leq x_1 \leq 8 + M(1 - y_3), \\ &x_2 \leq 2 + M(1 - y_3), \\ &x_3 \leq 5 + M(1 - y_3), \\ &y_1 + y_2 + y_3 = 1, \\ &x_i \geq 0, \ \forall i = 1, 2, 3, \end{aligned}$$

$$y_i \in \{0, 1\}, \ \forall i = 1, 2, 3.$$

Appendix 2: Generating instance problems

In the following mathematical formulation we have k objective functions, m constraints, q binary variables, and n continuous positive variables. The size of instance is displayed as $k \times m \times (n + q)$. U_j is an integer value that shows the upper bound of variable y_j for $j = 1, \ldots, q$.

$$max \;\; z_t(x,y) = \sum_{i=1}^{n} c_i^t x_i + \sum_{j=1}^{q} f_j^t y_j, \;\; \forall t = 1,\ldots,k$$

s.t.

$$\sum_{i=1}^{n} a_{ij} x_i + a'_j y_j \leq b_j, \;\; \forall j = 1,\ldots,q,$$

$$\sum_{i=1}^{n} a_{ij} x_i \leq b_j, \;\; \forall j = q+1,\ldots,m-1,$$

$$\sum_{j=1}^{q} y_j \leq \frac{q}{3},$$

$$x_i \in \mathcal{R}^+, \;\; \forall\, i = 1,\ldots,n,$$

$$y_j \in \{0,1,\ldots,U_j\}, \;\; \forall\, j = 1,\ldots,q,$$

where, in the described benchmarks, the objective function coefficients of the continuous variables, binary variables, the right hand sides of the constraints, and the matrix of coefficients (for both continuous and binary variables) are drawn from uniformly distributed random numbers in the ranges $[-10, 10]$, $[-200, 200]$, [50, 100], and $[-1, 20]$, respectively. In addition, the sparsity of coefficient matrix is 40 percent.

References

Abounacer R, Rekik M, Renaud J (2014) An exact solution approach for multi-objective location-transportation problem for disaster response. Comput Oper Res 41:83–93

Alves MJ, Clımaco J (2000) An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. Eur J Oper Res 124(3):478–494

Alves MJ, Clímaco J (2007) A review of interactive methods for multiobjective integer and mixed-integer programming. Eur J Oper Res 180(1):99–115

Alves MJ, Costa JP (2016) Graphical exploration of the weight space in three-objective mixed integer linear programs. Eur J Oper Res 248(1):72–83

Anvari S, Turkay M (2017) The facility location problem from the perspective of triple bottom line accounting of sustainability. Int J Prod Res 55(21):6266–6287

Armand P (1993) Finding all maximal efficient faces in multiobjective linear programming. Math Program 61(1–3):357–375

Armand P, Malivert C (1991) Determination of the efficient set in multiobjective linear programming. J Optim Theory Appl 70(3):467–489

Belotti P, Soylu B, Wiecek MM (2013) A branch-and-bound algorithm for biobjective mixed-integer programs. Optimization Online. http://www.optimization-online.org/DB_FILE/2013/01/3719.pdf

Boland N, Charkhgard H, Savelsbergh M (2014) A simple and efficient algorithm for solving three objective integer programs. Optimization Online. http://www.optimization-online.org/DB_FILE/2014/09/4534.pdf

Boland N, Charkhgard H, Savelsbergh M (2015) A criterion space search algorithm for biobjective mixed integer programming: the triangle splitting method. INFORMS J Comput 27(4):597–618

Boland N, Charkhgard H, Savelsbergh M (2016) The l-shape search method for triobjective integer programming. Math Program Comput 8(2):217–251

Boland N, Charkhgard H, Savelsbergh M (2017a) 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

Boland N, Charkhgard H, Savelsbergh M (2017b) The quadrant shrinking method: a simple and efficient algorithm for solving tri-objective integer programs. Eur J Oper Res 260(3):873–885
Can E, Erol S (2014) A multi-objective mixed integer linear programming model for energy resource allocation problem: the case of turkey. Gazi Univ J Sci 27(4):1157–1168
Charkhgard H, Takalloo M, Haider Z (2018) Bi-objective autonomous vehicle repositioning problem with travel time uncertainty http://www.optimization-online.org/DB_HTML/2017/06/6104.html
Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. Wiley, New York
Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evolut Comput 6(2):182–197
Ehrgott M (2006) A discussion of scalarization techniques for multiple objective integer programming. Ann Oper Res 147(1):343–360
Ehrgott M, Puerto J, Rodriguez-Chia A (2007) Primal-dual simplex method for multiobjective linear programming. J Optim Theory Appl 134(3):483–497
Evans JP, Steuer R (1973) A revised simplex method for linear multiple objective programs. Math Program 5(1):54–72
Fattahi A, Turkay M (2018) A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems. Eur J Oper Res 266(2):415–425
Gobbi M (2013) A k, k-ε optimality selection based multi objective genetic algorithm with applications to vehicle engineering. Optim Eng 14(2):345–360
Hooker J (2011) Logic-based methods for optimization: combining optimization and constraint satisfaction, vol 2. Wiley, New York
Hooker JN (1994) Logic-based methods for optimization. In: Principles and practice of constraint programming. Springer, Berlin, Heidelberg, pp 336–349
Jahanshahloo GR, Hosseinzadeh F, Shoja N, Tohidi G (2005) A method for generating all efficient solutions of 0–1 multi-objective linear programming problem. Appl Math Comput 169(2):874–886
Jorge JM (2009) An algorithm for optimizing a linear function over an integer efficient set. Eur J Oper Res 195(1):98–103
Lokman B, Köksalan M (2013) Finding all nondominated points of multi-objective integer programs. J Glob Optim 57(2):347–365
Lokman B, Köksalan M (2014) Finding highly preferred points for multi-objective integer programs. IIE Trans 46(11):1181–1195
Mavrotas G, Diakoulaki D (2005) Multi-criteria branch and bound: a vector maximization algorithm for mixed 0–1 multiple objective linear programming. Appl Math Comput 171(1):53–71
Mavrotas G, Florios K (2013) An improved version of the augmented ε-constraint method (augmecon2) for finding the exact pareto set in multi-objective integer programming problems. Appl Math Comput 219(18):9652–9669
Miettinen K, Hakanen J, Podkopaev D (2016) Interactive nonlinear multiobjective optimization methods. In: Multiple criteria decision analysis. Springer, New York, pp 927–976
Nadarajah SK, Tatossian C (2010) Multi-objective aerodynamic shape optimization for unsteady viscous flows. Optim Eng 11(1):67–106
Najafi M, Eshghi K, Dullaert W (2013) A multi-objective robust optimization model for logistics planning in the earthquake response phase. Transp Res Part E: Logist Transp Rev 49(1):217–249
Özlen M, Azizoğlu M (2009) Multi-objective integer programming: a general approach for generating all non-dominated solutions. Eur J Oper Res 199(1):25–35
Özlen M, Burton BA, MacRae CA (2014) Multi-objective integer programming: an improved recursive algorithm. J Optim Theory Appl 160(2):470–482
Özpeynirci Ö, Köksalan M (2010) An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. Manag Sci 56(12):2302–2315
Pascual-González J, Jiménez-Esteller L, Guillén-Gosálbez G, Siirola JJ, Grossmann IE (2016) Macro-economic multi-objective input-output model for minimizing CO2 emissions: application to the US economy. AIChE J 62(10):3639–3656
Przybylski A, Gandibleux X, Ehrgott M (2010) A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. INFORMS J Comput 22(3):371–386
Rudloff B, Ulus F, Vanderbei R (2017) A parametric simplex algorithm for linear vector optimization problems. Math Program 163(1–2):213–242
Saule C, Giegerich R (2015) Pareto optimization in algebraic dynamic programming. Algorithms Mol Biol 10(1):1

Sayin S (1996) An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. Oper Res Lett 19(2):87–94
Schechter M (2005) A correction to the connectedness of the Evans–Steuer algorithm of multiple objective linear programming. Found Comput Dec Sci 30(4):351–360
Soylu B, Yıldız GB (2016) An exact algorithm for biobjective mixed integer linear programming problems. Comput Oper Res 72:204–213
Steuer RE (1994) Random problem generation and the computation of efficient extreme points in multiple objective linear programming. Comput Optim Appl 3(4):333–347
Steuer RE, Choo EU (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26(3):326–344
Stidsen T, Andersen KA, Dammann B (2014) A branch and bound algorithm for a class of biobjective mixed integer programs. Manag Sci 60(4):1009–1032
Sylva J, Crema A (2004) A method for finding the set of non-dominated vectors for multiple objective integer linear programs. Eur J Oper Res 158(1):46–55
Tohidi G, Razavyan S (2012) An l1-norm method for generating all of efficient solutions of multi-objective integer linear programming problem. J Ind Eng Int 8(1):1–8
Vadenbo C, Hellweg S, Guillén-Gosálbez G (2014) Multi-objective optimization of waste and resource management in industrial networks—part I: model description. Resour Conserv Recycl 89:52–63
Vincent T, Seipp F, Ruzika S, Przybylski A, Gandibleux X (2013) Multiple objective branch and bound for mixed 0–1 linear programming: corrections and improvements for the biobjective case. Comput Oper Res 40(1):498–509
Wiecek MM, Ehrgott M, Engau A (2016) Continuous multiobjective programming. Springer, New York, pp 739–815
Yu PL, Zeleny M (1975) The set of all nondominated solutions in linear cases and a multicriteria simplex method. J Math Anal Appl 49(2):430–468
Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications, vol 63. Ithaca, Shaker