GoNDEF: A New Exact Method to Generate All Non-Dominated Points of Multi-Objective Mixed-Integer Linear Programs

Seyyed Amir Babak Rasmi · Metin Türkay

Abstract Most real-life decision problems involve more than a single criterion that are often conflicting with each other. This category of problems are called multi-criteria/multi-objective optimization problems (MCOP/MOOP). One of the most important tasks for MOOPs is finding the non-dominated (ND) points in the objective space or efficient solutions in the decision space. A ND point results in objective function values that cannot be improved without worsening another objective function. In this paper, we present a novel method to generate the set of entire ND points for a multi-objective mixed-integer linear program (MOMILP). The Generator of ND and Efficient Frontier for MOMILPs (GoNDEF) finds the ND points represented as points, line segments, and facets consisting of all types of the ND points. First, GoNDEF finds integer values for integer variables that generate the ND points. Then, our method fixes integer variables to specific values. The resulting problem is 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. 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

Seyyed Amir Babak Rasmi
Department of Industrial Engineering, Koç University, Istanbul, Turkey, 34450
E-mail: rasmi14@ku.edu.tr

Metin Türkay
Department of Industrial Engineering, Koç University, Istanbul, Turkey, 34450
Tel.: +90 212-338-1586
E-mail: mturkay@ku.edu.tr
1 Introduction

Many decision making processes concern more than a single criterion. There are a wide range of decision problems in engineering, business, health care, medicine, and chemistry including genetic network stability, facility location, forward/reverse logistics, urban transportation, and portfolio optimization inherently have more than one decision criterion. Moreover, even single objective problems may be converted to multiple-objective problems to explore trade-offs between different issues. In the literature of multi-objective optimization problems (MOOP), a variety of solution approaches have been applied to solve real-life problems such as resource management problems (Can et al, 2014; Vadenbo et al, 2014), (sustainable) location/transportation problems (Abouaccer et al, 2014; Anvari and Turkay, 2017; Pascual-González et al, 2016), 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 Climaco (2007) and Miettinen et al (2016) study interactive methods for MOOPs. On the other hand, exact methods are designed to find all or a subset of non-dominated (ND) points or efficient solutions set. Moreover, evolutionary algorithms find approximate solutions to cover a large subset of the ND points set (e.g., Deb (2001); Deb et al (2002); Zitzler (1999)).

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 (assume that \( n \geq k \)) 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 are present, such as mixed-integer programming problems, this image becomes a finite number of at most \( k \)-dimensional polyhedra in the objective space (we discuss about each polyhedron of this case in section 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 solutions of MOLPs are used by Evans and Steuer (1973) and Yu and Zeleny (1975). They provide a thorough theoretical analysis of MOLPs that are continued by other studies (Rudloff et al, 2017;
Generating Non-Dominated Points of MOMILPs

Many researchers have 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 MOLP are provided (see Clímaco and Gouveia (2005) and Wieck et al (2016) for more discussions).

MOIPs are more complicated than MOLPs. These problems have attracted a lot of attention to generate a subset, an approximation, or the entire set of the ND points set 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; Tolici and Razav, 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 are applied to MOIPs (Jorge, 2009; Lokman and Köksalan, 2013; Sylv and Crema, 2004) and other studies used the characteristics of $\epsilon$-constraint method for solving MOIPs (Mavrotas and Florios, 2013; Özlen and Azizoğlu, 2009; Özlen et al, 2014). Boland et al (2014, 2016, 2017b) present different algorithms to generate the entire ND points 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 contains 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 have been addressed in the literature infrequently. MOMILPs are very practical for real-life 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); Boland et al (2015); Fattahi and Turkay (2018); Soylu and Yldz (2016); Sülen et al (2014); Vincent et al (2013) propose exact algorithms to generate all ND points of BOMILPs. These algorithms use different methods to find integer solutions that generate some subsets of the ND points set. 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, an approach based on branch and bound to solve these problems is provided that compares the solutions with 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 Clímaco, 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 solutions of the remaining MOLP, and removing the dominated points by previously-found ND points. These studies do not address the entire ND points in the form of the ND facets of MOMILPs. One of important types of ND points is 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 and Köksalan (2010) and Przybylski et al (2010) present algorithms to find the set of 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 the entire 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 the entire ND points set in practice. In this paper, we present a novel method to find the entire ND points of a general MOMILP.

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

2 Problem Definition

A general MOMILP is given in (1).

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

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:
Generating Non-Dominated Points of MOMILPs

\[ \text{sub-MOLP}(\bar{y}) : \]
\[
\begin{align*}
\max \ z(x, \bar{y}) &= C_C x + C_Z \bar{y} \\
\text{s.t. } A_C x &\leq b - A_Z \bar{y}, x \in \mathbb{R}^n.
\end{align*}
\]

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 \( \{ x \in S(\bar{y}), y \in \mathbb{Z}^q \} \). 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, \textit{integer solution}.

In general, the objective functions of a MOOP conflict with each other; hence, the concept of optimality is replaced by the Pareto optimality where one aims to generate the set of the ND points. 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 \( \{ x \in S(y), y \in \mathbb{Z}^q, z_i(x, y) \geq \hat{z}_i, i = 1, ..., k \} \) and at least one \( z_i(x, y) > \hat{z}_i \). Consequently, let \( z(\hat{x}, \hat{y}) = \hat{z} \) 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 solutions. Yu and Zeleny (1975) propose a multi-criteria simplex method to solve MOLPs. This method provides all ND extreme solutions, all ND facets, and identifies adjacent ND extreme solutions. Let \( NDES_\bar{y} \) and \( NDFC_\bar{y} \) be the set of all ND extreme solutions and all ND facets, respectively. Next, we define an efficient integer solution and a ND facet for MOMILPs.

**Definition 1** If \( \bar{z} \in S(\bar{y}) \) exists such that \( (\bar{x}, \bar{y}) \) is an efficient solution of (1), then \( \bar{y} \) is an \textit{efficient integer solution}.

**Definition 2** Let \( F \) be the convex hull of some ND extreme solutions of (2) and a ND facet for the sub-MOLP. If \( \bar{z} \in F \) exists such that \( \bar{z} \) is ND for MOMILP given in (1), then \( F \) is a \textit{ND facet} for MOMILP.

We provide an instance of MOMILP with three objective functions (TOMILP, \( k = 3 \)). Figure 1 shows this instance in the objective space from two different perspectives (see Appendix A for the formulation). We show the existing polyhedra by green, red, and blue colors. Note that each integer solution corresponds to a polyhedron. We denote these integer solutions by \( y_{GR} \), \( y_{RE} \), and \( y_{BL} \). 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 corresponds to a sub-MOLP.
In this paper, if \( z^1 \) and \( z^2 \) are 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 \([0, 1]^n\) and \((0, 1)^n\) to show closed and open intervals, respectively. For example, \([C, F]\) is a line segment that includes point \( C \) but not point \( F \).

In this illustrative example, \([A, B]\) is the ND points set of the green polyhedron. Then, we determine \( NDES_{yGR} = \{[A, B]\} \), \( NDFC_{yGR} = \{ConvexHull([A, B])\} \) and identify that \( A \) and \( B \) are adjacent.\(^1\) 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 \( C, F, G, H \). and \((H)\).\(^2\) 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 \([A, B]\) (see Definition 2). We find \( NDES_{yBL} = NDFC_{yBL} = \{[I]\} \), \( I = (8, 2, 5) \) by solving sub-MOLP(\( y_{BL} \)). Note that the ND facet of sub-MOLP(\( y_{BL} \)) is a 0-dimensional facet. Moreover, point \( I \) is not ND for the MOMILP since it is dominated by some points that belong to the red polyhedron. Then, there is no ND point in the blue polyhedron and hence, \( y_{GR} \) and \( y_{RE} \) are the efficient integer solutions of this instance.

In Figure 1, note that facet \((C-F-G-H)\) is formed by four edges (line segments) which are the boundaries of the facet. These edges are formed by the convex hull of pairs of adjacent ND extreme solutions. Specifying the ND segments of these edges (e.g., \([C, D]\), \([F, G]\), and \([F, H]\)) provides a better and more practical presentation of the ND facets. Specifically, when a facet is partially ND, identifying the ND segments of its edges provides a clearer presentation of that facet.

In the next section, we present our method, GoNDEF (Generator of ND and Efficient Frontier for MOMILPs), that generates the ND facets of a MOMILP (e.g., \((C-F-G-H)\) and \([A, B]\)) and the ND segments of the edges between pairs of adjacent ND extreme solutions (e.g., \([C, D]\) and \([E, F]\) which are the ND segments of the edge \([C, F]\)). Note that our method does not locate the dominated parts of the partially ND facets that cannot be shown as the convex combination of two adjacent ND extreme solutions.

\(^1\) \( A = (6, 3, 10) \) and \( B = (3, 6, 10) \)

\(^2\) \( C = (8, 0, 9), F = (1, 7, 9), G = (3, 9, 3), H = (10, 2, 3) \). Note that \( NDES_{yRE} = \{C, F, G, H\} \) and \( NDFC_{yRE} = \{ConvexHull([C, F, G, H])\} \).

\(^3\) Note that point \( D \) is not included in the set of ND points since \( D \) is dominated by \( A \). Point \( D \) is a weakly ND point. We do not address this type of points.
3 GoNDEF: A Novel Method For Solving MOMILPs

We present a novel method that effectively solves a general MOMILP with a novel algorithmic approach. The GoNDEF iteratively finds an integer solution, then it solves the sub-MOLP associated with this integer solution if this is an efficient integer solution (see Definition 1). Note that solving each sub-MOLP results in its entire ND points in the form of at most \((k - 1)\)-dimensional facets (set \(NDFC\)) and the ND extreme solutions (set \(NDES\)). 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 are more than one efficient solution associated with a ND point, they are generated. However, the computational efforts in the algorithms that work in the objective space are easier.

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.

Fig. 1: An illustrative example for the MOMILP given in (1)
Step 1: Find efficient integer solution \( \bar{y} \) such that \( \bar{y} \notin EIS \). Then, 
\[ EIS = EIS \cup \{ \bar{y} \}. \]

Step 2: Solve sub-MOLP(\( \bar{y} \)) that gives \( NDES_{\bar{y}} \) and \( NDFC_{\bar{y}} \).

Step 3: Identify the ND segments of each edge between pairs of adjacent ND extreme solutions in \( NDES_{\bar{y}} \).

Step 4: Identify the ND facets of the MOMILP in \( NDFC_{\bar{y}} \) and go to Step 1.

Assume that we find an efficient integer solution in Step 1 for a MOMILP, then some ND points of the sub-MOLP associated with this efficient integer solution are ND for the MOMILP. In the ND frontier of this sub-MOLP, there are edges between pairs of adjacent ND extreme solutions. Note that solving sub-MLOPs in Step 2 of GoNDEF could become very time-consuming. Hence, Step 1 generates integer solutions that are efficient in order to save computational effort. 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. Next, we provide the details of each step of our method.

### 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)\(^4\) and Mavrotas and Dinkoulaki (2005) to check the dominance of a point.

\[
DZ(\hat{z}, Y_{exc}, ExC) : \tag{3}
\]

\[
\max \sum_{i=1}^{k} \epsilon_i
\]

s.t. \( z_i(x, y) - \epsilon_i \geq \hat{z}_i, i = 1, \ldots, k \),
\( x \in S(y), y \in \mathbb{Z}^q \setminus Y_{exc}, ExC, \)
\( \epsilon_i \geq 0, i = 1, \ldots, k, \)

where \( \hat{z} \in \mathbb{R}^k \) is a point in the objective space. \( ExC \) is the set of constraints to exclude some regions from the feasible region. Let \( exd(\hat{z}) \) be the set of constraints that exclude the dominated cone of \( \hat{z} \in \mathbb{R}^k \) in the objective space.

---

\(^4\) This study provides a similar formulation to minimize Tchebycheff distance between the points in the objective space and the ideal point.
Then, \( \text{exd}(\hat{z}) := \{ \hat{z} + \delta \mid z_i(x, y) + Mt_i, \sum_{i=1}^{k} t_i \leq k - 1, t_i \in \{0, 1\}, i = 1, ..., k \} \) excludes the dominated cone of \( \hat{z} \) where \( M \) is a sufficiently large positive number and \( \delta \) is a sufficiently small positive number. Note that if \( \hat{z} \) is a ND point and we add \( \text{exd}(\hat{z}) \) to (3), then \( \delta = 0 \) may result in a point that is weakly ND. Moreover, \( Y_{\text{exc}} \) is the set of excluded integer solutions.\(^5\) Assume that (3) is feasible, then \((x^D, y^D)\) denotes the solution that results in optimal objective function value and \( z^D := z(x^D, y^D) \).

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

**DZ1** Assume that both \( Y_{\text{exc}} \) 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 \( \text{DZ} \) 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 \( \hat{z} \).

**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 \( Y_{\text{exc}} = ExC = \emptyset \). Then, \( z^D \) is ND.

**Proof.** The optimal objective function value is \( \sum_{i=1}^{k} (z^D_i - \hat{z}_i) \). 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 \( z^{ND}_i \geq z^D_i, i = 1, ..., k \), and at least for one \( i \), \( z^{ND}_i > z^D_i \). Then, \( \sum_{i=1}^{k} (z^D_i - \hat{z}_i) < \sum_{i=1}^{k} (z^{ND}_i - \hat{z}_i) \) which contradicts that the optimal objective function value is \( \sum_{i=1}^{k} (z^D_i - \hat{z}_i) \). \( \square \)

We next define the NDR point \( (z^S) \) as an approximation of the Nadir point of (1) such that \( z^S \) is dominated by the Nadir point. For example, \( z^S = \min \{ z_i(x, y) \mid x \in S(y), y \in \mathbb{Z}^q \} \), for \( i = 1, ..., k \), provides \( z^S \) that is dominated by or the same as the Nadir point. 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. In Algorithm 1, we develop a method to check the efficiency of \( y^* \). If we set \( Y_{\text{exc}} := \mathbb{Z}^n \setminus y^* \), then \( \text{DZ}(z^S, Y_{\text{exc}}, ExC) \) in the second line of Algorithm 1 results in \( z^D \) that is ND for sub-MOLP(\( y^* \)).\(^6\) If \( z^D \) is ND for the MOMILP, then \( y^D \) is an efficient integer solution. Otherwise,

\(^5\) Note that we exclude a specific vector of integer values \( \hat{y} \) by using no-good constraints (Hooker, 2011, 1994; Rasani et al, 2018; Soylu and Yıldız, 2016).

\(^6\) Note that \( y \in \mathbb{Z}^n \{ \mathbb{Z}^n \setminus y^* \} \) is equivalent to fixing \( y \) to \( y^* \).
we find a point such as $z^{D1}$ that dominates $z^D$. Then, we add the constraints that exclude the dominated cone of $z^{D1}$ to $DZ$ in 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: $DZ(z^S, y^*, ExC) \rightarrow z^D := z(x^D, y^*)$
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: $DZ(z^D, \{y^*\}, \emptyset) \rightarrow z^{D1} = z(x^{D1}, y^{D1})$ (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 that tighten the region $\{x \in S(y^*)\}$. These new constraints exclude the dominated regions in $\{z(x, y) | x \in S(y^*)\}$. We continue this procedure to find a ND point or see the infeasibility of $DZ$. The infeasibility of $DZ$ shows that all solutions in the region $\{x \in S(y^*)\}$ are inefficient.

#### 3.1.2 Finding all efficient integer solutions

In this section, we present a method to generate all efficient integer solutions in Step1. As shown in Algorithm 2, we solve $DZ(z^S, Y_{exc}, ExC)$ iteratively. In each iteration, the solution of the problem given in (3) leads to a potentially ND point. 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 $Y_{exc}$ and $ExC$. Then, we cannot guarantee that $z^D$ is ND. $z^D$ may be dominated by some points as $\{z(x, y) | x \in S(y), y \in Y_{exc}\}$. Our method solves $DZ(z^D, y^D, \emptyset)$ to check the dominance of $z^D$. 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 to check whether $y^D$ is efficient or not. 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. We continue this process until the infeasibility of $DZ(z^S, Y_{exc}, ExC)$ happens.

Without loss of generality, for the simplicity of presentation, we illustrate this part of our method on a BOMILP example with maximization provided in Figure 2. There are six polyhedra corresponding to different integer solutions in Figure 2. We set NDR point to $(0, 1)$ and show by the blue point. At the first iteration, we solve $DZ(z^S, \emptyset, \emptyset)$. Optimal objective function value is 19 ($14 + 5$) and found at $1$ (Figure 2(b)). Let $1$ be associated with integer solution $y^1$. $1$ is ND and hence, $y^1$ is an efficient in-
Algorithm 2: Generate all efficient integer solutions

1: \( \text{ExC} = \emptyset, \text{Y}_{exc} = \emptyset \)
2: \( \text{DZ}(z^S, \text{Y}_{exc}, \text{ExC}) \rightarrow z^D := z(x^D, y^D) \)
3: if \( \text{DZ} \) is infeasible then
4: Stop.
5: else
6: Check the dominance of \( z^D \) by solving \( \text{DZ}(z^D, \emptyset, \emptyset) \rightarrow z^{D1} := z(x^{D1}, y^{D1}) \)
7: \( \text{Y}_{exc} := \text{Y}_{exc} \cup \{y^D\} \)
8: if \( z^D \) is ND then
9: \( y^D \) is an efficient integer solution.
10: \( \text{ExC} := \text{ExC} \cup \text{exd}(z^D) \)
11: else
12: Check the efficiency of \( y^D \) by applying Algorithm 1
13: if \( y^D \) is efficient integer solution then \( \text{ExC} := \text{ExC} \cup \text{exd}(z^D) \)
14: go to line 2

\(^8\) In line 13, \( z^D \) is found in Algorithm 1.

Integer solution. At the next iteration (Figure 2(c)), \( \text{ExC} := \{\text{exd}(1)\} \) and \( \text{Y}_{exc} := \{y^1\}. \) Solution of \( \text{DZ}(z^S, \{y^1\}, \text{exd}(1)) \) happens at (2) which is ND. Then, we update \( \text{ExC} \) and \( \text{Y}_{exc}. \) At the next iteration (Figure 2(d)), we solve \( \text{DZ}(z^S, \{y^1, y^2\}, \{\text{exd}(1), \text{exd}(2)\}) \). Solution is (3) that is ND. Hence, \( y^3 \) is an efficient integer solution. We update \( \text{Y}_{exc} \) and \( \text{ExC}. \) At the next iteration (Figure 2(e)), solving \( \text{DZ}(z^S, \text{Y}_{exc}, \text{ExC}) \) 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 (Figure 2(f)). Then, we set \( \text{Y}_{exc} := \{y^1, y^2, y^3, y^4\} \) and \( \text{ExC} := \{\text{exd}(1), \text{exd}(2), \text{exd}(3), \text{exd}(5)\} \). At the next iteration (Figure 2(g)), \( \text{DZ} \) is infeasible and hence, there is no more efficient integer solution.

Note that Algorithm 2 iterates four times and there are six integer solutions. Since we exclude some regions by \( \text{ExC} \), the algorithm does not enumerate all integer solutions and performs effectively. For example, Figure 2(g) shows that \( y^5 \) and \( y^6 \) are not in the set of excluded integer solutions. They are excluded due to \( \text{ExC} \) not \( \text{Y}_{exc}. \) 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 of an instance of (1), then Algorithm 2 identifies \( \bar{y} \) as an efficient integer solution at one of the iterations.

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 \( \text{ExC}. \) \( \text{ExC} \) 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 set \( \text{ExC} \) do not exclude \( \bar{z}. \) Note that we assume that the value of \( \delta \) in constraints \( \text{exd} \) is sufficiently small. Then, a ND \( \bar{z} \) is not excluded in the objective space due to the large value of \( \delta. \)

\(^8\) We hatch the regions that are excluded by \( \text{ExC} \) and use dashed-lines for the polyhedra that are excluded because of \( \text{Y}_{exc}. \)
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 $DZ(z^{S}, Y_{exc}, ExC)$ to $Y_{exc}$. Hence if $\bar{y} \in Y_{exc}$ then at an iteration that $\bar{y} \notin Y_{exc}$, $z(\hat{x}, \bar{y})$ where $\hat{x} \in S(\bar{y})$ have been
found by solving $\text{DZ}(z^S, Y_{exc}, ExC)$. At that iteration, if $z(\hat{x}, \hat{y})$ is ND, then $\hat{y}$ have been found as an efficient integer solution. If $z(\hat{x}, \hat{y})$ is not ND, then we have identified $\hat{y}$ as an efficient integer solution by applying Algorithm 1. Then, $\hat{y} \in Y_{exc}$ 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 $y^1$ and $y^2$.

3.2 Step3: Finding the ND segments of each edge

We find the efficient integer solutions in Step1 and $\text{NDES}/\text{NDFC}$ corresponding the efficient integer solutions in Step2. The edges of polyhedra in the objective space that correspond to the efficient integer solutions are the line segments between pairs of adjacent ND extreme solutions. 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 ($y^3, ..., y^5$). In this example, we are interested in finding the ND segments of the edge between $1$ and $2$. Note that $1$ and $2$ are two adjacent ND extreme solutions of sub-MOLP($y^1$). Moreover, let $4 = \lambda 1 + (1 - \lambda) 2$ for $\lambda \in [0, 1]$. Then, $4 - \epsilon$ denotes point $(\lambda - \epsilon) 1 + (1 - \lambda + \epsilon) 2$ in the objective space where $\epsilon$ is a sufficiently small positive number. We use this notation in the description of the illustrative example provided in Figure 3.

We solve $\text{DZ}((1), \{y^1\}, \emptyset)$ to check the dominance of $(1)$. $(1)$ is dominated and solving $\text{DZ}$ 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 $(1)$ that is dominated by $\{z(x, y^2) | x \in S(y^2)\}$. The linear program given in (4) results in an optimal value of $\alpha$ that gives $(3)$ if $z' := 1$, $z'' := 2$, and $\bar{y} := y^2$. Let $\alpha^E$ be the optimal value of $\alpha$. Then, $(3) = \alpha^E 1 + (1 - \alpha^E) 2$. Due to Proposition 2, $[1, 3]$ is a dominated segment.

$$\text{EDG1}(z', z'', \bar{y}) : \quad \text{(4)}$$

$$\min \alpha$$

subject to $x \in S(\bar{y})$,

$$z_i(x, \bar{y}) \geq \alpha z'_i + (1 - \alpha) z''_i, i = 1, ..., k,$$

$$0 \leq \alpha \leq 1.$$
Proposition 2 Let \( z' \) and \( z'' \) be two adjacent ND extreme solutions. \( z' \) is dominated by some \( z(x, \bar{y}) \) such that \( x \in S(\bar{y}) \). Moreover, let \( \alpha^E \) be the optimal value of \( \alpha \) 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 \( \bar{z} := z(\lambda x' + (1 - \lambda) x^\alpha, \bar{y}) = \lambda z(x', \bar{y}) + (1 - \lambda) z(x^\alpha, \bar{y}). \)

Then, \( \bar{z} \) dominates \( \hat{z} \) 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 \( \text{DZ}(\{3\} - \epsilon, \{y^1\}, \emptyset) \). \( \{3\} - \epsilon \) is dominated by some points in the polyhedron associated with \( y^3 \). Then, we solve \( \text{EDG1}(\{3\} - \epsilon, \{2\}, y^3) \) that results in an optimal value of \( \alpha \) which gives point \( \{4\} \). Hence, segment \( [\{3\}, \{4\}] \) is dominated. Note that \( \{4\} \) is dominated and \( \{4\} \) is ND.\(^9\) Then, \( \{4\} \) is a weakly ND point.

\(^9\) Note that \( \lambda x' + (1 - \lambda) x^\alpha \in S(\bar{y}) \) since \( S(\bar{y}) \) is a convex set.

\(^{10}\) \( \{4\} \) is a weakly ND point.
we aim to find a segment in the convex hull of \((4)^{-\varepsilon}\) and \((2)\) starting from \((4)^{-\varepsilon}\) that is ND. The MILP given in (5) results in an optimal value of \(\alpha\) that gives \((5)\) if \(z' := (4)^{-\varepsilon}, z'' := (2)\), and \(Y_{\text{exc}} := \{y^1\}\). The optimal value of \(\alpha^E\) is a value such that \((5) = \alpha^E(4)^{-\varepsilon} + (1 - \alpha^E)(2)\). Due to Proposition 3, \([(4)^{-\varepsilon}, (5)]\) is a ND segment.

\[
\text{EDG2}(z', z'', Y_{\text{exc}}) : \quad \begin{align*}
\max \alpha \\
\text{s.t.} \quad x \in S(y), \ y \in \mathbb{Z}^q \setminus Y_{\text{exc}}, \\
\quad z_i(x, y) &\geq \alpha z'_i + (1 - \alpha)z''_i, \ i = 1, \ldots, k, \\
\quad 0 &\leq \alpha \leq 1.
\end{align*}
\]

**Proposition 3** Let \(z'\) and \(z''\) be two adjacent ND extreme solutions and \(z'\) is ND. Let \(z'\) and \(z''\) correspond to integer solution \(y'\). Assume that \(Y_{\text{exc}} := \{y^1\}\). Moreover, let \(\alpha^E\) be the optimal value of \(\alpha\) 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 \(\alpha\) in EDG2 given in (5) is greater than or equal to \(\lambda + \alpha^E\) that contradicts the optimality of \(\alpha^E\). \(\square\)

For finding the ND segments in segment \([(5), (2)]\), note that \((5)\) and \((5)^{-\varepsilon}\) are dominated.\(^{11}\) Similar to the discussed procedure for finding a ND segment starting from \((1)\), we again solve EDG1((5)^{-\varepsilon}, (2), \{y\}) to find \((6)\) and \((6)^{-\varepsilon}\) is ND. Then, we solve EDG2((6)^{-\varepsilon}, (2), \{y^1\}) that results in \((7)\). We identify \([(6), (7)]\) as a ND segment. Then, we identify that \((7)\) and \((7)^{-\varepsilon}\) are dominated. Hence, we solve EDG1((7)^{-\varepsilon}, (2), \{y\}) that results in \(\alpha^E = 0\) and shows that there is no ND segment in \([(7)^{-\varepsilon}, (2)]\). Then, we are finished with finding the ND segments in edge \([(1), (2)]\). Segments \([(4), (5)]\) and \([(6), (7)]\) are the ND segments.

Algorithm 3 shows the process for finding the ND segments of the edge \([z', z'']\) associated with \(\vec{y}\). Note that if \(z'\) is ND and EDG2(\(z', z'', \{\vec{y}\}\)) is infeasible, then there is no \(x(y, y)\) such that \(x \in S(y), \ y \in \mathbb{Z}^q \setminus \vec{y}, \) and \(z_i(x, y) \geq \alpha z'_i + (1 - \alpha) z''_i\) for all \(i = 1, \ldots, k\). Hence, \([z', z'']\) is a ND segment.

\(^{11}\) \((4)\) is a weakly ND point.
Algorithm 3 Find the ND segments of the edge $[z', z'']$ associated with $\tilde{y}$

1: $z^o := z'$
2: while $z^o \neq z''$ do
3: Solve $\text{DZ}(z', (\tilde{y}), \emptyset) \rightarrow z^o := z(x^o, y^o)$
4: if $z'$ is ND then
5: Solve $\text{EDG2}(z', z'', (\tilde{y})) \rightarrow \alpha := \alpha^E z' + (1-\alpha^E)z''$
6: if EDG2 is feasible then
7: $[z', \alpha] \text{ is a ND segment}$
8: Check the dominance of $z'$ solving $\text{DZ}(z', (\tilde{y}), \emptyset)$.
9: $z' := (\alpha^E - \epsilon)z' + (1-\alpha^E + \epsilon)z''$
10: else
11: $[z', z''] \text{ is a ND segment. Exit the loop.}$
12: else
13: Solve $\text{EDG1}(z', z'', y^D) \rightarrow \alpha := \alpha^E z' + (1-\alpha^E)z''$
14: $[z', \alpha] \text{ is a dominated segment.}$
15: Check the dominance of $z'$ solving $\text{DZ}(z', (\tilde{y}), \emptyset)$.
16: if $\alpha \neq 0$ then
17: $z' := (\alpha^E - \epsilon)z' + (1-\alpha^E + \epsilon)z''$
18: $\epsilon$ is a sufficiently small positive number.

3.3 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 set by points and line segments. Then, in this section, we aim to characterize the entire ND facets using the following:

FC1 Information obtained from Step 3 (ND points in the forms 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 $\tilde{y}$. Let $F$ be a ND facet of sub-MOLP ($\tilde{y}$) in the objective space ($F \in \text{NDFC}_{\tilde{y}}$). We characterize $F$ by its corners that are ND extreme solutions. Let $z^j, j = 1, ..., n_F$ be the extreme solutions of $F$. Then, $F$ is $\text{ConvexHull}\{z^1, ..., z^n_F\}$. Note that per extreme solution $z^j, j = 1, ..., n_F$, there is at least one adjacent extreme solution in $\{z^p, p \in \{1, ..., n_F\}, p \neq j\}$. In sections 3.3.1 and 3.3.2, we show the processes to check the dominance of facet $F$.

3.3.1 FC1

In section 3.2, we identify all ND extreme solutions and the ND segments of edges. We admit that facet $F$ is ND if at least one $z^j, j = 1, ..., n_F$, is ND. If $z^* j$ is dominated for all $j = 1, ..., n_F$, then we look for a ND segment in the edges between pairs of adjacent ND extreme solutions of $F$. If at least one ND segment exists, then $F$ is a ND facet.

Let $z^j$ is dominated for all $j = 1, ..., n_F$. Moreover, assume that there is no ND segment in the edges between pairs of adjacent ND extreme solutions associated with $F$. Then, we may identify $F$ as a dominated facet using Proposition 4.
Proposition 4 Let all extreme solutions 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 $j = 1, ..., n_F$. 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 $j = 1, ..., n_F$. Then, $\bar{z} = \sum_{j=1}^{n_F} \lambda_j z(x^j, \hat{y})$ dominates $\bar{z}$ since $\lambda_j z_i(x^j, \hat{y}) \geq \lambda_j z_i^j$ for all $i = 1, ..., k$, and $j = 1, ..., n_F$. 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 $z(x^j, \hat{y})$, $j = 1, ..., n_F$. Then, $x \in S(\hat{y})$ exists such that $z(x, \tilde{y}) = \hat{z}$. \hfill $\square$

3.3.2 FC2

Assume that we cannot identify the dominance of $F$ using the discussed methods in section 3.3.1. In section 3.1.1, we examine the efficiency of an integer solution $(\bar{y})$ by 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 $z^1, ..., z^{n_F}$ in the objective space. We provide a MILP in (6) for finding a potentially ND point in $F$.

$$\text{DF}(z^1, ..., z^{n_F}, \bar{y}, ExC) :$$

\[
\text{max} \quad \sum_{i=1}^{k} \epsilon_i \\
\text{s.t.} \quad z_i(x, \bar{y}) - \epsilon_i \geq z_i^\bar{y}, i = 1, ..., k, \\
\quad z_i(x, \bar{y}) = \sum_{j=1}^{n_F} \lambda_j z_i^j, i = 1, ..., k, \quad \sum_{j=1}^{n_F} \lambda_j = 1, \\
\quad 0 \leq \lambda_j \leq 1, j = 1, ..., n_F, \\
\quad x \in S(\bar{y}), ExC, \\
\quad \epsilon_i \geq 0, i = 1, ..., k.
\]

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 $\text{ConvexHull}(z^1, ..., z^{n_F})$. Algorithm 4 works similar to Algorithm 1. It finds a point in $F$ that is not dominated by previously found ND points. Note that $ExC$ is the set of constraints that exclude the dominated cone of some previously found ND points.

\[\text{Note that } \tilde{z} \text{ is a point in } F.\]
Algorithm 4 Check the dominance of \( F := \text{ConvexHull}\{z^1, \ldots, z^n_F\} \) associated with \( \bar{y} \)

1: \( \text{ExC} = \emptyset \)
2: \( \text{DF}(z^1, \ldots, z^n_F, \text{ExC}) \to \lambda^D := (\lambda^D_1, \ldots, \lambda^D_{n_F}), z^D := \sum_{j=1}^{n_F} \lambda^D_j z^j \)
3: if \( \text{DF} \) is infeasible then
4: \( F \) is a dominated facet.
5: else
6: if \( z^D \) is ND then
7: \( F \) is a ND facet.
8: else
9: \( \text{DZ}(z^D, \{\bar{y}\}, \emptyset) \to z^{D1} := z(x^{D1}, y^{D1}) \)
10: \( \text{ExC} := \text{ExC} \cup \text{ext}(z^{D1}) \)
11: go to line 2

4 Numerical Experiments

We illustrate the effectiveness of 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 instances and the range of parameters are provided in Appendix B. We implement our algorithm in MATLAB R2015b 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 \((\# \text{ LP and } \# \text{ MILP})\), the number of the efficient integer solutions \((\# \text{ eff. int. sol.})\), and total CPU time in seconds \((\text{CPUT (sec.)})\). Since the complexity of problem increases by instance size, we show the average cost of finding an efficient integer solution. We report the average number of solved MILPs \((\# \text{ MILP per eff. int. sol.})\) and the average CPU time that is consumed for finding an efficient integer solution \((\text{CPUT per eff. int. sol. (sec.)})\). Then, the number of MILPs's per efficient integer solution and CPU time per efficient integer solution refer to \( ^n \# \text{ MILP per eff. int. sol.} \) and \( ^n \text{CPUT per eff. int. sol. (sec.)} \), respectively.

The value of \( \delta \) in the used excluding constraints of the set \( \text{ExC} \) modifies excluding regions; the larger values of \( \delta \) result in the larger excluding regions. However, the probability of missing an efficient integer solution increases. We test our method with \( \delta_i \) for the \( i^{th} \) objective function where \( i = 1, \ldots, k \). Let \( \delta_i \) be equal to the multiplication of \( \Delta \) and the range of \( i^{th} \) objective function.

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. Then, ND segments and facets are not generated and hence, the number of them are not reported.

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

<table>
<thead>
<tr>
<th>$k$</th>
<th>$m$</th>
<th>$n$</th>
<th>$q$</th>
<th>$#$ MILP</th>
<th>$#$ eff. int. sol.</th>
<th>CPU (sec.)</th>
<th>$#$ MILP per eff. int. sol.</th>
<th>CPU per eff. int. sol. (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>1.6</td>
<td>0.10</td>
<td>6.25</td>
<td>0.06</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>41.8</td>
<td>7.2</td>
<td>1.37</td>
<td>5.81</td>
<td>0.19</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>52.4</td>
<td>6.8</td>
<td>1.84</td>
<td>7.71</td>
<td>0.27</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>109</td>
<td>13.8</td>
<td>4.05</td>
<td>7.90</td>
<td>0.29</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>261.6</td>
<td>19.6</td>
<td>18.41</td>
<td>13.35</td>
<td>0.94</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>15.4</td>
<td>3.6</td>
<td>0.19</td>
<td>4.28</td>
<td>0.05</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>105.6</td>
<td>26.2</td>
<td>4.84</td>
<td>4.03</td>
<td>0.18</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>379.6</td>
<td>47.4</td>
<td>34.93</td>
<td>8.01</td>
<td>0.74</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>830.4</td>
<td>55</td>
<td>181.19</td>
<td>15.10</td>
<td>3.29</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>3470.8</td>
<td>151.4</td>
<td>5603.94*</td>
<td>6.31</td>
<td>13.11</td>
</tr>
</tbody>
</table>

* There are instances that took more than 2 hours.

integer solutions. First, note that regarding 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 works effectively on the set of provided 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 Figure 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 $\Delta$. 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 efficient integer solutions in Figures 4(a), 4(b), and 4(c), respectively. We show the results associate TOMILP instances by continuous lines and associate 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 $\Delta$. However, the smaller
Table 2: Solving MOMILP instances with binary variables using Algorithm 2 for finding the efficient integer solutions ($\Delta = 0.01$)

<table>
<thead>
<tr>
<th>$k$</th>
<th>$m$</th>
<th>$n$</th>
<th>$q$</th>
<th>$# \text{MILP}$</th>
<th>$# \text{eff. int. sol.}$</th>
<th>CPU (sec.)</th>
<th>$# \text{MILP per eff. int. sol.}$</th>
<th>CPU per eff. int. sol. (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>34</td>
<td>2</td>
<td>0.42</td>
<td>17.00</td>
<td>0.21</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>46.8</td>
<td>9.6</td>
<td>1.53</td>
<td>5.08</td>
<td>0.16</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>60.8</td>
<td>8.2</td>
<td>2.15</td>
<td>7.41</td>
<td>0.26</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>119</td>
<td>15.8</td>
<td>5.17</td>
<td>7.53</td>
<td>0.33</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>256.2</td>
<td>20.8</td>
<td>22.34</td>
<td>12.32</td>
<td>1.07</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>61.2</td>
<td>3.8</td>
<td>0.84</td>
<td>16.11</td>
<td>0.22</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>190.6</td>
<td>32.2</td>
<td>7.19</td>
<td>5.92</td>
<td>0.22</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>496.2</td>
<td>57.2</td>
<td>41.25</td>
<td>8.67</td>
<td>0.72</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>1112.8</td>
<td>80.2</td>
<td>141.62</td>
<td>13.88</td>
<td>1.80</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>4395.4</td>
<td>629.8</td>
<td>7210.12*</td>
<td>19.83</td>
<td>25.46</td>
</tr>
</tbody>
</table>

* There are instances that took more than 2 hours.

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

<table>
<thead>
<tr>
<th>$k$</th>
<th>$m$</th>
<th>$n$</th>
<th>$q$</th>
<th>$# \text{MILP}$</th>
<th>$# \text{eff. int. sol.}$</th>
<th>CPU (sec.)</th>
<th>$# \text{MILP per eff. int. sol.}$</th>
<th>CPU per eff. int. sol. (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>264.4</td>
<td>2</td>
<td>2.77</td>
<td>1322.20</td>
<td>1.38</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>195.6</td>
<td>10.2</td>
<td>3.65</td>
<td>19.18</td>
<td>0.36</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>344</td>
<td>9.6</td>
<td>4.83</td>
<td>25.42</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>317.4</td>
<td>18.6</td>
<td>9.06</td>
<td>17.06</td>
<td>0.49</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>479.6</td>
<td>30</td>
<td>28.72</td>
<td>15.99</td>
<td>0.69</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>516.4</td>
<td>3.8</td>
<td>7.47</td>
<td>1355.89</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>916.4</td>
<td>32.8</td>
<td>21.10</td>
<td>27.94</td>
<td>0.64</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>2292.2</td>
<td>66.6</td>
<td>88.82</td>
<td>34.42</td>
<td>1.33</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>6537</td>
<td>92</td>
<td>314.90</td>
<td>71.05</td>
<td>3.42</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>21005.8</td>
<td>263.8</td>
<td>6009.70*</td>
<td>79.63</td>
<td>22.78</td>
</tr>
</tbody>
</table>

* There are instances that took more than 2 hours.

values of $\# \text{MILP per eff. int. sol.}$ and $\# \text{CPU per eff. int. sol.}$ indicate a better performance.

Figure 4(a) 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, Figure 4(b) 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
Fig. 4: The results of the GoNDEF performance by changing $\Delta$, the number of objective functions, and instance size

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 Figure 4(c), 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 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.

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- 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 $U = 2$. Regarding this table, the increases in the number of solved MILPs are smaller than the increases in the number of integer solutions. However, these 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 also report the number of ND segments (# ND segments) and the number of ND facets (# ND faces). Note that the

---

14 We provide the changes for the classes in which all instances are completely solved.
Table 4: Solving MOMILP instances with integer variables \((U = 2)\) using Algorithm 2 for finding the efficient integer solutions \((\Delta = 0.01)\)

<table>
<thead>
<tr>
<th>(k)</th>
<th>(m)</th>
<th>(n)</th>
<th>(q)</th>
<th># MILP</th>
<th># eff. int. sol.</th>
<th>CPU (sec.)</th>
<th># MILP per eff. int. sol.</th>
<th>CPU per eff. int. sol. (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>34</td>
<td>2</td>
<td>0.37</td>
<td>17.00</td>
<td>0.18</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>56.8</td>
<td>9</td>
<td>1.78</td>
<td>6.53</td>
<td>0.20</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>15</td>
<td>59.2</td>
<td>9.8</td>
<td>2.75</td>
<td>6.04</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>165.4</td>
<td>19.6</td>
<td>10.28</td>
<td>8.44</td>
<td>0.52</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>520.2</td>
<td>26.6</td>
<td>91.98</td>
<td>19.56</td>
<td>3.46</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>61.2</td>
<td>3.8</td>
<td>0.88</td>
<td>16.11</td>
<td>0.33</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>218.4</td>
<td>36.8</td>
<td>11.57</td>
<td>5.93</td>
<td>0.31</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>500.8</td>
<td>79.4</td>
<td>203.74</td>
<td>7.06</td>
<td>2.57</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>1321.8</td>
<td>107.8</td>
<td>577.44</td>
<td>12.26</td>
<td>5.36</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>3646.4</td>
<td>225</td>
<td>6991.34*</td>
<td>14.30</td>
<td>27.42</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>35.8</td>
<td>4.4</td>
<td>0.52</td>
<td>8.14</td>
<td>0.12</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>336.6</td>
<td>62.2</td>
<td>30.83</td>
<td>8.63</td>
<td>0.30</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>2269.2</td>
<td>238.8</td>
<td>2470.97</td>
<td>9.30</td>
<td>10.35</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>3371.4</td>
<td>350</td>
<td>577.44</td>
<td>12.26</td>
<td>5.36</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>4286</td>
<td>652.4</td>
<td>7215.94*</td>
<td>9.39</td>
<td>20.10</td>
</tr>
</tbody>
</table>

* There are instances that took more than 2 hours.

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

<table>
<thead>
<tr>
<th>(k)</th>
<th>(m)</th>
<th>(n)</th>
<th>(q)</th>
<th>% change in # of integer solutions</th>
<th>% change in # MILP</th>
<th>% change in CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>-12.0</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>30.8</td>
<td>20.5</td>
<td>16.1</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>15</td>
<td>59.2</td>
<td>9.8</td>
<td>2.75</td>
<td>6.04</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>165.4</td>
<td>19.6</td>
<td>10.28</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>520.2</td>
<td>26.6</td>
<td>91.98</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>61.2</td>
<td>3.8</td>
<td>0.88</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>218.4</td>
<td>36.8</td>
<td>11.57</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>500.8</td>
<td>79.4</td>
<td>203.74</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>1321.8</td>
<td>107.8</td>
<td>577.44</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>3646.4</td>
<td>225</td>
<td>6991.34*</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>35.8</td>
<td>4.4</td>
<td>0.52</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>336.6</td>
<td>62.2</td>
<td>30.83</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>2269.2</td>
<td>238.8</td>
<td>2470.97</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>3371.4</td>
<td>350</td>
<td>577.44</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>4286</td>
<td>652.4</td>
<td>7215.94*</td>
</tr>
</tbody>
</table>

Number of ND segments refers to the sum of separate single points and line segments. 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 our of total CPU time \((% \text{ MOLP CPU})\). Moreover, in the last two columns, we show the consumed CPU time for finding a ND facet averagely. Column 12 shows the average CPU time for finding a ND 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: Solving MOMILP instances with binary variables by the GoNDEF ($\Delta = 0.01$)

<table>
<thead>
<tr>
<th>$k$</th>
<th>$m$</th>
<th>n</th>
<th>q</th>
<th>$\varepsilon_{LP}$</th>
<th>$\varepsilon_{MILP}$</th>
<th>$\varepsilon_{eff. int.}$</th>
<th>$\varepsilon_{ND}$ segments</th>
<th>$\varepsilon_{ND}$ facets</th>
<th>$% MOLP$</th>
<th>CPU time (CPU per facet)</th>
<th>CPU time (CPU per facet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>17.2</td>
<td>26.0</td>
<td>0.4</td>
<td>4.4</td>
<td>0</td>
<td>0.01</td>
<td>42.2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>32.3</td>
<td>35.5</td>
<td>0.8</td>
<td>9.6</td>
<td>0</td>
<td>0.01</td>
<td>82.4</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>63.8</td>
<td>63.6</td>
<td>0.8</td>
<td>9.6</td>
<td>0</td>
<td>0.01</td>
<td>160.2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>104.6</td>
<td>104.6</td>
<td>0.8</td>
<td>9.6</td>
<td>0</td>
<td>0.01</td>
<td>251.2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>148.0</td>
<td>148.0</td>
<td>0.8</td>
<td>9.6</td>
<td>0</td>
<td>0.01</td>
<td>628.6</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>108.8</td>
<td>122.6</td>
<td>3.8</td>
<td>32.2</td>
<td>11.6</td>
<td>13.31</td>
<td>1.58</td>
<td>0.136</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>4304.6</td>
<td>6541.2</td>
<td>47.8</td>
<td>3263.6</td>
<td>277.2</td>
<td>12.67</td>
<td>42.40</td>
<td>0.153</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>21036.4</td>
<td>69665.8</td>
<td>58.6</td>
<td>37616</td>
<td>2949.4</td>
<td>13.02</td>
<td>226.42</td>
<td>0.191</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>1200758.6</td>
<td>69665.8</td>
<td>58.6</td>
<td>37616</td>
<td>2949.4</td>
<td>13.02</td>
<td>226.42</td>
<td>0.191</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>1009587.6</td>
<td>69665.8</td>
<td>58.6</td>
<td>37616</td>
<td>2949.4</td>
<td>13.02</td>
<td>226.42</td>
<td>0.191</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>188.6</td>
<td>142.2</td>
<td>4.4</td>
<td>35.0</td>
<td>9.8</td>
<td>17.27</td>
<td>2.14</td>
<td>0.218</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>49740.4</td>
<td>6541.2</td>
<td>47.8</td>
<td>3263.6</td>
<td>277.2</td>
<td>12.67</td>
<td>42.40</td>
<td>0.153</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>1200758.6</td>
<td>69665.8</td>
<td>58.6</td>
<td>37616</td>
<td>2949.4</td>
<td>13.02</td>
<td>226.42</td>
<td>0.191</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>2506057.2</td>
<td>69665.8</td>
<td>58.6</td>
<td>37616</td>
<td>2949.4</td>
<td>13.02</td>
<td>226.42</td>
<td>0.191</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>25</td>
<td>25</td>
<td>1604889.4</td>
<td>69665.8</td>
<td>58.6</td>
<td>37616</td>
<td>2949.4</td>
<td>13.02</td>
<td>226.42</td>
<td>0.191</td>
</tr>
</tbody>
</table>

*There are instances that took more than 10 hours.

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 number of ND segments and facets significantly increase. Regarding solving the sub-MOLPs, in sections 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 impacts on the average CPU time for finding a ND facet.

5 Conclusions

In this paper, we present a method, the Generator of ND and Efficient Frontier (GoNDEF), for finding 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 clearer and more practical representation of the partially ND facets, the GoNDEF identifies the ND segments of the edges between pairs of adjacent ND extreme solutions. A number of innovative characteristics of the GoNDEF lead to high performance and practicality in solving MOMILPs. By choosing appropriate values for NDR 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 due to their managerial inherent and our method can generate all efficient integer solutions. This characteristic allows us to generate all efficient
integer solutions significantly faster than generating the entire ND points set. The computational results of solving a set of instance problems display that the entire or a large subset of the ND points set is generated effectively by the GoNDEF.

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 critical study that can be discussed in the future.

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 Yildirim, Emre Mengi, Seyed Mojtaba Hosseini, Ali Fattahi, Matthias Ehrgott, and referees of Optimization and Engineering journal.

A The formulation of our illustrative instance given in Figure 1

We provide a mathematical formulation for Figure 1. Note that there may be other formulations to provide a feasible region corresponding to Figure 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{align*}
\text{max } & z(x, y) = (x_1, x_2, x_3) \\
\text{s.t.} & \\
& 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), \\
& -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), \\
& 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, \\
& y_i \in \{0, 1\}, \forall i = 1, 2, 3.
\end{align*}
B 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 \( j = 1, \ldots, q \).

\[
\max z_t(x, y) = \sum_{i=1}^{n} c^t_i x_i + \sum_{j=1}^{q} f^t_j y_j, \quad \forall t = 1, \ldots, k
\]

s.t.

\[
\sum_{i=1}^{n} a_{ij} x_i + a^t_j y_j \leq b_j, \quad \forall j = 1, \ldots, q,
\]

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

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

\( x_i \in \mathbb{R}^+, \quad \forall i = 1, \ldots, n \),

\( y_j \in \{0, 1, \ldots, U_j\}, \quad \forall j = 1, \ldots, q \),

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 percents.

References


Generating Non-Dominated Points of MOMILPs


Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: Methods and applications