Mathematical Methods of Operations Research (2024) 100:221–262
https://doi.org/10.1007/s00186-023-00843-y

ORIGINAL ARTICLE

A fast and robust algorithm for solving biobjective mixed integer programs

Diego Pecin[1] · Ian Herszterg[2] · Tyler Perini[3] · Natashia Boland[2] · Martin Savelsbergh[2]

Received: 30 January 2023 / Revised: 14 November 2023 / Accepted: 20 November 2023 /
Published online: 12 January 2024

Abstract

We present a fast and robust algorithm for solving biobjective mixed integer linear programs. Two existing methods are studied: ϵ-Tabu Method and the Boxed Line Method. By observing structural characteristics of nondominated frontiers and computational bottlenecks, we develop enhanced versions of each method. Limitations of the current state of test instances are observed, and a new body of instances are generated to diversify computational standards. We demonstrate efficacy with a computational study. The enhancement to ϵ-Tabu Method offers an average speed-up factor of 3 on some instances; the enhancement to Boxed Line Method offers an average speed-up factor of 18 on some instances. A hybrid, two-phase method is designed to leverage the strengths of each method with its corresponding enhancement, thus having a robust approach to a wider range of instances; it outperforms on all instances with a typical speed-up factor of 2–3. We also demonstrate that it is capable of producing a high-quality approximation of the nondominated frontier in a fraction of the time required to produce the complete nondominated frontier.

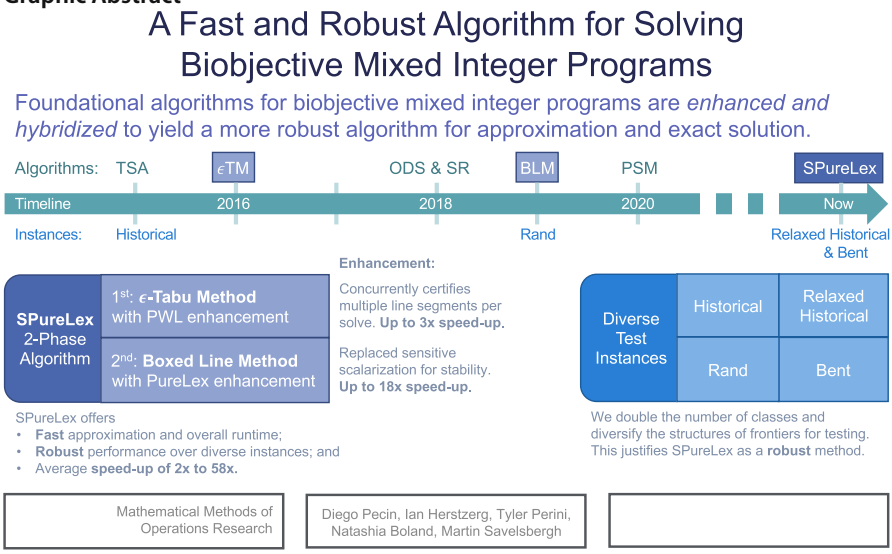
✉ Tyler Perini
tyler.perini@rice.edu

1 Econometric Institute, Erasmus University Rotterdam, Rotterdam, The Netherlands

2 H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, USA

3 Mathematics Department, United States Naval Academy, Annapolis, USA

Graphic Abstract



Keywords Biobjective mixed integer program · Criterion space search · Approximation

1 Introduction

In multiobjective optimization, the goal is to generate a set of solutions that induces the nondominated frontier (NDF), also known as the Pareto front (Pareto 1996). The NDF is the set of nondominated points (NDPs), where an NDP is a vector of objective values evaluated at a feasible solution with the property that there exists no other feasible solution that is at least as good in all objective values and is better in at least one of them. This paper focuses on multiobjective mixed integer linear programs (MOMIPs) defined by: continuous and integer variables, linear constraints, and two linear objective functions; it also focuses on exact algorithms which produce the complete NDF.

Multiobjective integer linear programming problems have been studied for many decades; see, for example, the surveys of Ehrgott and Gandibleux (2000), Gandibleux (2006), Stidsen et al. (2014) and Halffmann et al. (2022). In this paper, we restrict our discussion to *criterion space search* methods, in which the search for the NDF operates in the space of the vectors of objective values, known as the criterion space. These methods benefit from advances in single-objective mathematical programming solvers, since single-objective linear programs (LPs) and integer linear programs (IPs) are repeatedly solved.

To date, there is a rich history of criterion space search algorithms for generating the NDF of biobjective (pure) IPs (Chankong et al. 2008; Ralphs et al. 2004; Boland et al. 2015a; Dai and Charkhgard 2018) and/or more objectives (Sylva and Crema 2004; Przybylski et al. 2010b; Kirlik and Sayın 2014; Klamroth et al. 2015; Dächert and

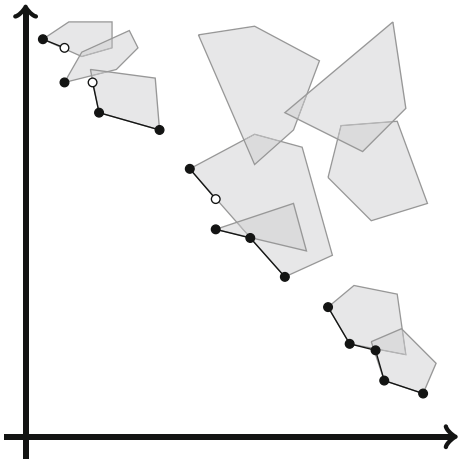


Fig. 1 The nondominated frontier of a BOMIP with minimization objectives. The shaded regions represent the images of feasible solutions with a common integer part. The nondominated frontier is darkened

Klamroth 2015; Boland et al. 2016, 2017; Tamby and Vanderpooten 2020). The *mixed* integer case poses quite different challenges. One challenge is that MOMIP frontiers can have a complex structure. The frontier of a biobjective mixed integer program (BOMIP), for example, can contain open, half-open, and closed line segments, in addition to isolated points (see Fig. 1 for an NDF of a BOMIP). The presence of these open, half-open, and closed line segments introduces many numerical challenges in the implementation of algorithms for generating the NDF of a BOMIP. In the last decade, several computationally effective, criterion space algorithms have appeared for BOMIPs (Boland et al. 2015b; Soylu and Yıldız 2016; Fattahi and Turkay 2018; Soylu 2018; Perini et al. 2019; Emre 2020).[1]

Recent work in MOMIPs includes generalizing to three or more objectives (Rasmi et al. 2017; Rasmi and Türkay 2019; Pettersson and Ozlen 2019; Ceyhan et al. 2023) and/or nonlinear objective functions (Cabrera-Guerrero et al. 2021; Diessel 2022; Eichfelder et al. 2023b).

Two-phase algorithms have played an important role in biobjective (pure) IPs; see, for example, Przybylski et al. (2010b) and Dai and Charkhgard (2018). The computational improvement of our two-stage approach (over the component methods) align with the recent two-stage application of balanced box method and the ϵ-constraint method (Dai and Charkhgard 2018).

Our work also contributes to expanding the library of BOMIP test instances; deeper investigation of generating instances for MOMIPs has only gained recent attention (Eichfelder 2023a).

In this paper, we offer the following contributions:

1. We extend and merge ideas from two BOMIP algorithms, namely the ϵ-Tabu Method (Soylu and Yıldız 2016) and the Boxed Line Method (Perini et al. 2019). The result is a two-stage hybridization whose performance is fast and, importantly, robust for generating the NDF of a BOMIP; it is robust in the sense that its two

[1] See Halffmann et al. (2022) for a more complete listing.

hyperparameters may be tuned so that it works well across a wide range of instances with different characteristics.
2. We demonstrate that the algorithm can produce a high-quality subset of the NDF in a fraction of the time it takes to generate the complete NDF. We rigorously explore this *approximation* of the NDF of a BOMIP and propose metrics to assess the quality of such an approximation.
3. We augment existing classes of test instances (from two pre-existing classes to four classes, total), which diversifies the set of test instances for BOMIP algorithms and justifies our claim that the algorithm is robust to a wide range of structural properties of the NDF.
4. We publish the code for this algorithm on GitHub (Pecin et al. 2022), which provides a widely applicable and highly efficient BOMIP algorithm with just two tunable hyperparameters.

The remainder of the paper is organized as follows. In Sect. 2, we introduce notation, key definitions, and review existing methods, including the ϵ-Tabu Method and the Boxed Line Method. In Sect. 3, we propose enhancements to the ϵ-Tabu Method and the Boxed Line Method, which improve their performance and robustness. In Sect. 4, we present the two-stage algorithm that merges ideas from these two methods. In Sect. 5, we discuss how the algorithm can be used to quickly produce an approximation of the NDF and metrics that assess the quality of that approximation. In Sect. 6, we present the results of a thorough computational study (full details included in the Appendices). We conclude in Sect. 7 with some final remarks.

2 Definitions and overview of component methods

We consider the biobjective mixed integer linear program (BOMIP)

$$\min_{x \in \mathcal{X}} \{z(x) := (z_1(x), z_2(x))\} \tag{1}$$

where $z_1(x), z_2(x)$ are linear in x and the feasible region $\mathcal{X} \subseteq \mathbb{Z}^{n_I} \times \mathbb{R}^{n_C}$ is assumed to be nonempty and bounded. To differentiate between the integer and continuous components of $x \in \mathcal{X}$, we use the convention $x = (x_I, x_C)$ where $x_I \in \mathbb{Z}^{n_I}$ and $x_C \in \mathbb{R}^{n_C}$. Let the projections of $\mathcal{X}$ onto the set of integer and real vectors be defined as $\mathcal{X}_I := \{x_I \in \mathbb{Z}^{n_I} : (x_I, x_C) \in \mathcal{X},\ x_C \in \mathbb{R}^{n_C}\}$ and $\mathcal{X}_C := \{x_C \in \mathbb{R}^{n_C} : (x_I, x_C) \in \mathcal{X},\ x_I \in \mathbb{Z}^{n_I}\}$, respectively. The feasible region $\mathcal{X}$ lies in the *decision space*, $\mathbb{R}^{n_I+n_C}$. The image of $\mathcal{X}$ under $z(\cdot)$, denoted by $\mathcal{Y} := \{y \in \mathbb{R}^2 : y = z(x),\ x \in \mathcal{X}\}$, lies in the *criteria space*, $\mathbb{R}^2$.

For $x^1, x^2 \in \mathcal{X}$, if $z_i(x^1) \leq z_i(x^2)$ for $i = 1, 2$ and $z(x^1) \neq z(x^2)$, then $z(x^1)$ *dominates* $z(x^2)$. If $x^N \in \mathcal{X}$ and there does not exist $x \in \mathcal{X}$ such that $z(x)$ dominates $z(x^N)$, then $z(x^N)$ is a *nondominated point* (NDP) and x^N is *efficient*. The union of all NDPs is the *nondominated frontier* (NDF), which we denote by $\mathcal{N}$. Hereon, we focus on biobjective optimization, since our methods depend on a natural ordering of points in the NDF.

A single NDP of (1) can be found by solving single-objective IPs[2] $\mathcal{X}$ for example with lexicographic optimization or a weighted scalarization. The lexicographic IP hierarchically minimizes two objectives in turn. We denote the case of minimizing $z_1(x)$ and then $z_2(x)$ by

$$\eta = \text{lexmin}\{(z_1(x), z_2(x)) : \ x \in \mathcal{X}\}. \tag{2}$$

Solving (2) requires solving two IPs in sequence: $\eta_1 = \min\{z_1(x) : \ x \in \mathcal{X}\}$ and then $\eta_2 = \min\{z_2(x) : \ z_1(x) \leq \eta_1, \ x \in \mathcal{X}\}$ are solved, resulting in an NDP $\eta = (\eta_1, \eta_2)$ of (1). In practice, the second IP tends to solve very quickly. For given vector $\lambda \in \mathbb{R}^2_+$, we refer to

$$\min\{\lambda^T z(x) : \ x \in \mathcal{X}\} \tag{3}$$

as the *scalarized IP with respect to* λ (Zadeh 1963). If x^λ is an optimal solution to (3) with positive λ, then $z(x^\lambda)$ is an NDP of (1) (Geoffrion 1968). Not every NDP in the NDF can be found by such an IP: if, for a given NDP $z(x_N)$, there exists positive vector λ such that x^N is an optimal solution to (3), then the NDP is *supported*; otherwise, the NDP is *unsupported*.

The NDF can be described by nondominated line segments, vertical gaps, and horizontal gaps. Define $L(z^1, z^2)$ to be the line segment connecting endpoints $z^1, z^2 \in \mathbb{R}^2$, where the endpoints are ordered from left to right so that $z^1_1 \leq z^2_1$. The line segment may be open at both ends, half-open, or closed. Thus

$$\{\xi z^1 + (1-\xi)z^2 : 0 < \xi < 1\} \subseteq L(z^1, z^2) \subseteq \{\xi z^1 + (1-\xi)z^2 : 0 \leq \xi \leq 1\}.$$

For each $i = 1, 2$, we refer to the endpoint z^i of $L(z^1, z^2)$ as *closed* if z^i belongs to the line segment, i.e., if $z^i \in L(z^1, z^2)$, and as *open* otherwise. In the case that $z^1 = z^2$, the line segment $L(z^1, z^2)$ consists of a single NDP. If all the points in $L(z^1, z^2)$ are nondominated and $L(z^1, z^2)$ is maximal with respect to set inclusion, then we call $L(z^1, z^2)$ a *nondominated line segment (NLS)*.[3]

The NDF may be described as a finite sequence of NLSs, $L(y^1, z^1), \ldots, L(y^K, z^K)$, say, for some $K \geq 1$, with $z^k_1 \leq y^{k+1}_1$ and $z^k_2 \geq y^{k+1}_2$ for all $k = 1, \ldots, K-1$, and for which

$$y^1_1 < y^2_1 < \cdots < y^K_1 \qquad \text{and} \qquad y^1_2 > y^2_2 > \cdots > y^K_2.$$

A gap may appear between second and first endpoints, respectively, of two consecutive NLSs in the NDF: for some k, it may be that $z^k_1 < y^{k+1}_1$, which is a gap in the horizontal direction, and/or $z^k_2 > y^{k+1}_2$, which implies a gap in the vertical direction. In the case that $z^k_1 < y^{k+1}_1$, we define the interval $(z^k_1, y^{k+1}_1) \subset \mathbb{R}$ to be a *horizontal*

[2] We use the term integer program (IP) to refer to any single-objective problem that has integer variables, including mixed integer linear programs.

[3] A NLS is maximal with respect to set inclusion if neither endpoint can be extended and still contain only NDPs.

gap. In this case, there exists no NDP p with $p_1 \in (z_1^k, y_1^{k+1})$ and z^k must be an NDP and hence must be a closed endpoint. In the case that $z_2^k > y_2^{k+1}$, we define the interval $(y_2^{k+1}, z_2^k) \subset \mathbb{R}$ to be a *vertical gap*. In this case, there exists no NDP p with $p_2 \in (y_2^{k+1}, z_2^k)$ and y^{k+1} must be an NDP and hence a closed endpoint. If there is a horizontal gap but no vertical gap between z^k and y^{k+1}, then y^{k+1} must be an open endpoint, and if there is a vertical gap but no horizontal gap between z^k and y^{k+1}, then z^k must be an open endpoint.

Given $x_I \in \mathcal{X}_I$, the BOLP obtained from fixing the integer variables to x_I is called the *slice problem for* x_I (Belotti et al. 2013). We call the NDF of a slice problem, which consists of a connected set of closed line segments, a *slice*.[4] The NDF of a slice problem for x_I is called the *slice for* x_I. Common methods for solving the BOLP slice problem include dichotomic search (Aneja and Nair 1979) (see "Appendix A.1" for reference) or parametric simplex (Yu and Zeleny 1975) which return the slice for one x_I. The NDF of the BOMIP is contained in the union, over all $x_I \in \mathcal{X}_I$, of the slice for x_I. The NDF consists of all points in this union that are not dominated by any other point in the union.

2.1 ϵ-Tabu method

The ϵ-Tabu Method (ϵTM) is an extension of the ϵ-constraint method for biobjective IPs (Haimes 1971; Chankong et al. 2008), which requires additional no-good or "Tabu" constraints for continuous portions of the frontier, and generates a BOMIP frontier from left to right (or vice-versa). An abbreviated pseudocode (Algorithm 1) is included in "Appendix A"; for more details see Soylu and Yıldız (2016).

To initialize, ϵTM solves a lexicographic IP to find the upper left NDP of the frontier, $z^L = \text{lexmin}\{(z_1(x), z_2(x)) : x \in \mathcal{X}\}$. Next, ϵTM repeats two steps: (1) solve the slice problem for the current integer solution, and (2) check (sequentially from left to right) whether each line segment in the resulting slice is dominated or not using a (modified) lexicographic IP. Once a line segment is found to be dominated, then return to (1) with the solution of the dominating image.

The latter step involves searching for the leftmost NDP "below" the line segment. If such an NDP is found, then ϵTM updates the rightmost endpoint of the line segment using the vertical projection of the new NDP onto the line segment; the line segment from the leftmost endpoint to the projected point is a NLS. Then, ϵTM processes the slice to which the new NDP belongs. If, on the other hand, no such NDP is found, then the full line segment is a NLS. Hence, ϵTM adds it to the frontier and proceeds to check the next line segment in the slice. If all line segments in the slice are confirmed to be nondominated, then ϵTM searches for the leftmost NDP "to the right" of the slice. If such an NDP is found, it defines the next slice. If no such NDP is found, the complete NDF has been found and ϵTM finishes.

If an NDF contains many consecutive NLSs from the same slice, as is the case in some benchmark instances, then it may be advantageous (more efficient) to check

[4] Note that our definition of a slice differs from Belotti et al. (2013), where it is defined as the *feasible set* for the slice problem as opposed to the resulting NDF.

whether a *set* of consecutive line segments from a slice is dominated or not. This new enhancement is presented in more detail in Sect. 3.1.

2.2 Boxed line method

The Boxed Line Method (BLM) (Perini et al. 2019) extended the Balanced Box Method (Boland et al. 2015a), which solves biobjective pure IPs, to handle continuous variables. Multiple variants of BLM were presented. The *basic* variant, described next is also summarized by pseudocode (Algorithm 2) in "Appendix A".[5] To initialize, BLM solves two lexicographic IPs to find the upper left and lower right NDPs of the frontier. These define a rectangular region, or "box", that is added to a queue. The outer loop of BLM iteratively processes boxes in this queue until it is empty; once an iteration completes and the queue remains empty, the algorithm terminates.

The first step in processing a box is to search for the leftmost NDP in the lower half of the box. This is achieved by solving a lexicographic IP constrained to the region of the criterion space below a horizontal line that splits the box. If the NDP found lies strictly below the split line, the outer loop solves a mirrored lexicographic IP to find the rightmost NDP that is in the box and to the left of the NDP already found. Up to two new boxes are added to the queue with a newfound NDP as a corner point[6]; boxes with a width/height smaller than some numerical tolerance are discarded.

Otherwise, the NDP lies on the split line and it must be that the split line intersects a NLS whose endpoints are yet unknown. To find these endpoints, BLM will first generate an *overestimate* (i.e., a superset) of the NLS by computing a line segment from a slice that contains the NDP. The inner loop is invoked to *refine* this line segment, eliminating dominated sections of it until it is proved to be a NLS. The original inner loop procedure in BLM accomplishes this by solving weighted-sum scalarization IPs. Computational experiments have revealed that solve times for these IPs are unpredictable and can sometimes be (too) long. To eliminate this unpredictability, we present replacement IPs in Sect. 3.2.

BLM quickly partitions the criterion space into regions that are empty or dominated and boxes that are unexplored. If an iteration of the outer loop begins processing a box with area X, the sum of the areas of the new boxes added to the queue is at most $X/2$. Furthermore, since each box can be processed independently, BLM is easily parallelizable. These strengths facilitate a rapid approximation of the NDF.

An additional variant, the *recursive* variant, of BLM is included during our computational study; it is described in detail in Perini et al. (2019).

2.3 Diversity of methods and singularity of instances

The literature on BOMIP algorithms has diverged far more than the set of test instances with which to compare. The original algorithm to solve BOMIP was the Triangle

[5] Readers are recommended to use Fig. 7 as a reference, and to seek Perini et al. (2019) for further details.

[6] The other corner point originates from original box. For instance, suppose within the original box, defined by corner points z^L and z^R, NDPs z^ℓ and z^r are found such that the images ordered from left-to-right are z^L, z^ℓ, z^r, z^R. Then one new box is defined by z^L and z^ℓ; the other is defined by z^r and z^R.

Splitting Algorithm (TSA) (Boland et al. 2015b); however, direct comparison to TSA has been ill-advised for reasons including the use of a relative error tolerance, as discussed in Fattahi and Turkay (2018) and Perini et al. (2019). The One-Direction Search (ODS) algorithm (Fattahi and Turkay 2018) is very close in design and spirit to ϵTM, and so we do not discuss it further. The computational study for the Search-and-Remove (SR) algorithm (Soylu 2018) compared SR against TSA and ϵTM on instances from Boland et al. (2015b). Emre (2020) presents a BOMIP method using the Pascoletti-Serafini scalarization, and the computational comparisons (although not directly on the same machine) are with SR and ODS.

The algorithms tested in this paper are coded efficiently for large-scale instances. The library of large-scale, linear instances from Boland et al. (2015b) remain the standard for computational experimentation of BOMIP algorithms. While randomly generated, their NDFs have a consistent, monolithic structure, as first discussed in Perini et al. (2019). We argue that this simplicity of NDF structure leads to a lack in understanding of the robustness of algorithms to different structural characteristics.

3 Enhancements

This section presents enhancements for both component methods.

3.1 An enhanced implementation of the ϵ-Tabu method

An enhanced implementation of ϵTM, denoted by ϵTM-PWL, solves a single lexicographic IP based on a mixed integer programming model of piecewise linear (PWL) functions. This single IP simultaneously considers multiple line segments of a slice. That is, rather than investigating the line segments one by one, from "left to right", we investigate them simultaneously, by solving a single lexicographic IP.

Suppose that from NDP $z^0 = z(x^0)$, the slice problem is solved for x_I^0 to generate a representative list of $K \geq 1$ line segments $\{L(z^0, z^1), L(z^1, z^2), \ldots, L(z^{K-1}, z^K)\}$ ordered from left to right.[7] To check a single line segment, say $L(z^i, z^{i+1})$, where z^i is known to be nondominated, ϵTM solves the following lexicographic IP:

$$\begin{aligned} \hat{z} = \text{lexmin} \quad & (z_1(x), z_2(x)) \\ \text{s.t.} \quad & z_1(x) \leq \lambda z_1^i + (1-\lambda) z_1^{i+1}, \\ & z_2(x) \leq \lambda z_2^i + (1-\lambda) z_2^{i+1}, \\ & x_I \neq x_I^0, \\ & x \in \mathcal{X}, 0 \leq \lambda \leq 1, \end{aligned} \tag{4}$$

where $x_I \neq x_I^0$ represents a no-good constraint. If (4) is feasible, then $\hat{z}$ is the leftmost NDP from a *different* slice that dominates some part of the line segment $L(z^i, z^{i+1})$.

[7] Note that the number of line segments, K, is readily determined from the method used to solve the slice problem, e.g., dichotomic search. Section 6.2 also gives an empirical estimate of the mean observed K value.

For ϵTM-PWL, we propose a new formulation that considers a set of (at most) K line segments in the slice simultaneously and models the piecewise linear slice function using binary variables (see, for example, Wolsey and Nemhauser 2014). Let $K > 0$ be the first hyperparameter we introduce.[8] We introduce K binary variables, y_i for $i = 1, \ldots, K$ where y_i is associated with $L(z^{i-1}, z^i)$, and $K + 1$ continuous variables, λ_i for $i = 0, \ldots, K$, one for each of the end points of the line segments. The following minimization problem computes the "left-most" point from a different slice that dominates any of the line segments:

$$\begin{aligned}
\hat{z}_1 = \min \quad & z_1(x) && \text{(5a)}\\
\text{s.t.} \quad & \sum_{i=0}^{K} \lambda_i = 1 && \text{(5b)}\\
& \sum_{i=1}^{K} y_i = 1 && \text{(5b)}\\
& \lambda_0 \le y_1 && \text{(5b)}\\
& \lambda_i \le y_i + y_{i+1} \quad \forall i = 1, \ldots, K-1 && \text{(5b)}\\
& \lambda_K \le y_K && \text{(5b)}\\
& z_1(x) = \sum_{i=0}^{K} \lambda_i z_1^i && \text{(5c)}\\
& z_2(x) \le \sum_{i=0}^{K} \lambda_i z_2^i && \text{(5d)}\\
& x_I \ne x_I^0, \quad x \in \mathcal{X}\\
& y_i \in \{0, 1\} \quad \forall i = 1, \ldots, K\\
& 0 \le \lambda_i \le 1 \quad \forall i = 0, \ldots, K
\end{aligned}$$

In a feasible solution of this model, $y_i = 1$ indicates that the NDP found is "below" line segment $L(z^{i-1}, z^i)$. Constraints (5b), together with the binary constraints on y_i variables, ensure that at most two convex multipliers (λ_i variables) may be positive, the rest must be zero, and that any positive values must be consecutive: $y_i = 1$ for exactly one i, which forces $y_j = 0$ for all $j \ne i$ and hence $\lambda_j = 0$ for all $j \notin \{i-1, i\}$. Since $\lambda_{i-1} + \lambda_i = 1$, $(\sum_{j=0}^{K} \lambda_j z_1^j, \sum_{j=0}^{K} \lambda_j z_2^j)$ is a convex combination of the points z^{i-1} and z^i, and hence is a point on the ith line segment, $L(z^{i-1}, z^i)$.

Constraints (5c) and (5d) thus ensure that the image $z(x)$ either dominates or coincides with a point on this line segment, while the objective function ensures that x gives the left-most such image. If the IP is infeasible, then the slice is nondominated (as there exists no feasible solution "below" its frontier). Otherwise the IP is feasible, and solving

[8] We discuss choosing the value of K in Sect. 6.2. If there are fewer than K line segments in a given slice, then the algorithm easily adapts to this, so we do not discuss this case further.

Table 1 Average and maximum relative increase in total running time of the variations of ϵTM for all sets of instances

Algorithm	Average				Max			
	Historical	Relaxed	Rand	Bent	Historical	Relaxed	Rand	Bent
ϵTM	1.87	1.10	0.00	0.00	2.14	1.33	0.00	0.00
ϵTM-PWL	0.00	0.00	0.64	1.93†	0.00	0.00	0.65	2.57†

Zero indicates the superior method: the enhancement is superior on the historical and relaxed instances, but not for the Rand and Bent instances. †—on one instance of the set, the runtime limit of 86,400 s was exceeded

$$\hat{z}_2 = \min\{z_2(x) \ : \ z_1(x) \leq \hat{z}_1, \ x_I \neq x_I^0, \ x \in \mathcal{X}\},$$

ensures that the result, $\hat{z}$, is a NDP. By construction, there is a unique $i \in \{1, \ldots, K\}$ such that $z_1^{i-1} < \hat{z}_1 \leq z_1^i$. Then $L(z^{j-1}, z^j)$ is a NLS for all $j = 1, \ldots, i-1$ and $L(z^{i-1}, \tilde{z})$ is a NLS, where $\tilde{z}$ is the vertical projection of $\hat{z}$ onto $L(z^{i-1}, z^i)$.

The proposed formulation adds $K + 3$ new constraints and $2K + 1$ new variables (of which K are binary). Note that one may decide to work with a partial slice, i.e., with the left-most $\hat{K} < K$ line segments obtained when solving the slice problem. That is, stop solving the slice problem after $\hat{K}$ line segments have been found and check if that portion of the frontier is dominated. If no dominated point is found, then resume solving the slice problem starting from the $(\hat{K} + 1)$-th point in the frontier, and repeat the process until we find either a dominating point or we reach the end of the frontier. Observe that when $\hat{K} = 1$, this enhancement collapses to the original ϵTM. Although we treat the hyperparameter K as a fixed value, it could instead be dynamically obtained and adjusted throughout the execution of the algorithm, as in Herszterg (2020).

The computational results of this enhancement are reported in Sect. 6.2 and summarized in Table 1.

3.2 A purely lexicographic boxed line method

Recall that in the basic variant of BLM, the line generation procedure returns a line segment from a slice, which is then refined to the nondominated portion of the line segment by the inner loop that solves one or more scalarized IPs (see Perini et al. 2019 for details). Experiments with BLM have revealed that solving scalarized IPs can be computationally expensive; they are typically more expensive than solving a lexicographic IP (and sometimes *much* more expensive). Figure 2 shows for three instances (described in more detail in Sect. 6.1) all lexicographic and scalarized IPs solved during the execution of the basic variant of BLM. The x-axis represents the area of the active box when the IP is solved, and the y-axis represents the solve time of the IP. We observe that for all lexicographic IPs, the solve time is in the order of a few seconds, whereas for some of the scalarized IPs, the solve time is close to 500 s. Furthermore, for some instances (Rand and Bent), a correlation of at least 0.7 indicates that the solve time for scalarized IPs increases with box size, and hence BLM

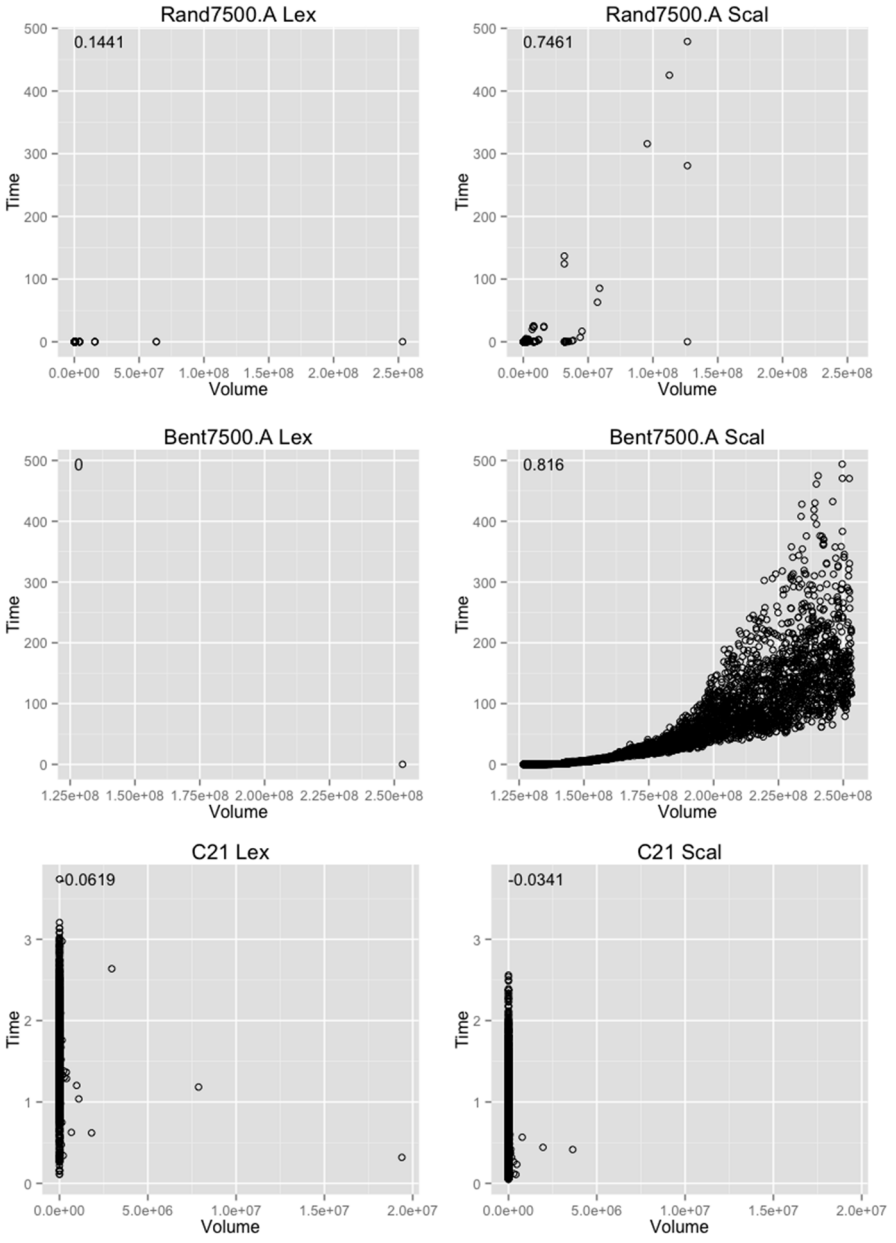


Fig. 2 Comparing solve times of lexicographic IP (6) (left) and scalarized IPs (right) across instances with respect to the area of boxed regions. Times are measured during BLM (basic variant) and plotted as points. Correlation coefficients are given in the upper left of each graph. Note that only one lexicographic IP is solved on the Bent instance, so the correlation is not well defined even though it is marked as 0. The correlation is at least 0.7 for scalarized IPs solved on the generated instances, i.e., Rand and Bent; elsewhere, the correlation is negligible

performance suffers early when boxes are still large. This suggests that a variant of BLM that only solves lexicographic IPs (i.e., avoids solving scalarized IPs) may be faster on average or, at least, have more "stable" solve times.

Therefore, we explore the benefits of a Purely Lexicographic Boxed Line Method (PURELEX). PURELEX replaces the weighted sum scalarization IPs for refining the initial overestimate of the line segment with lexicographic IPs.[9] It does so in a way similar to ϵTM, using one lexicographic IP solve per endpoint. Given a line segment $L(z^1, z^2)$ containing NDP $z^* = z(x^*)$ for some $x^* \in \mathcal{X}$ in its (relative) interior, let $\vec{w}$ represent the gradient of $L(z^1, z^2)$. PURELEX solves the following lexicographic IP to update z^1:

$$\begin{aligned} z^\alpha = \text{lexmin} \quad & (z_2(x), z_1(x)) \\ \text{s.t.} \quad & \vec{w}^T z(x) < \vec{w}^T z^*, \\ & z_1(x) \leq z_1^*, \\ & z_2(x) \leq z_2^1, \\ & x_I \neq x_I^*, \quad x \in \mathcal{X}. \end{aligned} \tag{6}$$

Note that (4) and (6) both model a lexicographic optimization of two objectives over a criterion space set with the same structure. In both cases, the criterion space set is defined by the intersection of three half spaces: two defined by the upper bounds on each objective and the third defined by the requirement to lie below a line (segment). They model this common structure in two different ways. The formulation (6) represents the three half-space constraints directly. The formulation (4) expresses them using a convex combination of the line segment endpoints, with a new continuous variable to model the weights in the convex combination. Both formulations impose a no-good constraint on the integer components of the solution. Unlike formulation (4), the formulation (6) uses a strict inequality (implemented as $\vec{w}^T z(x) \leq \vec{w}^T z^* - \epsilon$, for small $\epsilon > 0$) to make the current line segment infeasible. In theory, therefore, the no-good constraint is redundant. However, computational experiments have shown that including the no-good constraint in (6) provides better numerical stability. When running ϵTM where (4) is replaced with (6), experiments indicate that instances are solved slightly faster with this new formulation (an average improvement of 1% in computational runtime). Note also that we solve either two single-objective IPs, and find an NDP that dominates a portion of $L(z^1, z^*)$, or one (infeasible) single-objective IP, and prove that $L(z^1, z^*)$ is nondominated.

If (6) is infeasible, then endpoint z^1 does not need to be updated. Otherwise, the NDP z^α is used to update the endpoint z^1, using the horizontal projection of z^α onto $L(z^1, z^*)$, after which $L(z^1, z^*)$ is guaranteed to be nondominated. Solving the second IP in the lexicographic minimization is needed only because z^α is required to be nondominated in order to define the next box; to update the line segment, it suffices to solve the first IP.

[9] Compare procedure LINERESTRICTION.INNERLOOP in Algorithm 2 to LINERESTRICTION.PURELEX in Algorithm 3.

A similar lexicographic IP is used to update z^2:

$$\begin{aligned} z^\beta = \text{lexmin} \quad & (z_1(x), z_2(x)) \\ \text{s.t.} \quad & \vec{w}^T z(x) < \vec{w}^T z^*, \\ & z_1(x) \leq z_1^2, \\ & z_2(x) \leq z_2^*, \\ & x_I \neq x_I^*, \quad x \in \mathcal{X}. \end{aligned} \tag{7}$$

If (7) is infeasible, then endpoint z^2 does not need to be updated. Otherwise, the NDP z^β is used to update the endpoint z^2, using the vertical projection of z^β onto $L(z^*, z^2)$. After the update, $L(z^*, z^2)$ is guaranteed to be nondominated.

Thus, PureLex identified the nondominated portion of a line segment by solving a minimum of two and a maximum of four single objective IPs, compared to one or more (possibly expensive) scalarized IPs solved in BLM. Algorithm 3 presents the pseudo-code of PureLex.

4 A hybrid, two-phase algorithm

Reviewing the logic of ϵTM, we see that it is likely to solve fewer single-objective IPs than PureLex, because when it shortens a line segment to its nondominated portion, it does so by solving either one (infeasible) IP or two single-objective IPs (one if the entire line segment belongs to the frontier and two when the line segment has to be shortened). However, solving fewer single-objective IPs will not always results in shorter solve times, because run time also depends on the time it takes to solve the single-objective IPs, which is impacted by the size of the box. The plots in Fig. 3 show a clear correlation between the solve time of a single-objective IP and the area of the active box. This explains, in part, why (as we shall see from the results in Sect. 6) ϵTM is more effective than PureLex when the frontier lies in a "small" region of the criterion space, i.e, when the initial box $B(z^L, z^R)$ is "small". Another contributing factor is the fact that the frontier in a small region of the criterion space is likely to have fewer horizontal gaps, which is beneficial for ϵTM. Whenever ϵTM encounters a horizontal gap in the frontier, it needs to solve a lexicographic IP to find a new "starting" point, i.e., a new slice, which requires solving a lexicographic IP (and this lexicographic IP is not restricted to a small part of the box, which is usually the case when ϵTM determines whether a line segment belongs to the frontier or needs to be shortened).

This suggests that an algorithm that *switches* from PureLex to ϵTM at some point during the generation of the frontier may be able to exploit the respective strengths of these methods and allay their respective weaknesses. We propose such an algorithm, which we call SPureLex, as follows. SPureLex starts by executing PureLex to quickly decompose the criterion space into many small as-yet unexplored boxes, and, then, when the total as-yet unexplored area of the criterion space becomes small, i.e., less than a fraction ρ ($0 < \rho < 1$) of the area of the initial box $B(z^l, z^U)$, it switches

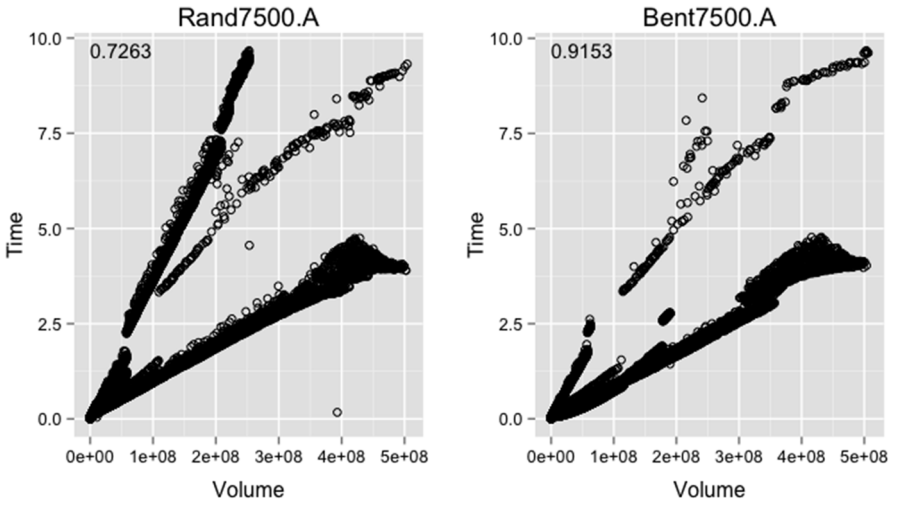


Fig. 3 The area of the box and the solve time for the lexicographic IPs solved by ϵTM, separated by instance, are plotted as points. The resulting correlation coefficient is in the upper left of each graph

to ϵTM to rapidly generate the frontier in the remaining boxes (defining the as-yet unexplored area of the criterion space). Early computational experiments indicated that small values for ρ were ideal; in Sect. 6, we use $\rho = 0.005$. Algorithm 4 presents the pseudo-code of SPureLex. We call SPureLex- PWL the variant of SPureLex using the ϵTM-PWL method.

5 Approximating a mixed integer nondominated frontier

In the literature on multiobjective optimization, an *approximation* of the NDF can take any of several different forms. Here, we use it to describe a subset of the NDF. In particular, we compare the parts of the NDF discovered by alternative exact algorithms prior to their completion. To do so, we require metrics that measure the quality of such an approximation. We introduce the following metrics illustrate the progress of the algorithms as well as the quality of the improvement of approximation:

1. The as-yet unexplored area of the criterion space (i.e., the area of the criterion space that may still contain parts of the frontier) as a fraction of the area of the initial box $B(z^L, z^R)$;
2. The fraction of isolated NDPs found;
3. The fraction of NLSs found;
4. The fraction of the total length of the NLSs found; and
5. The fraction of slices that contribute to the frontier found.

Note that metrics 2, 3, 4, and 5 are relative to the complete nondominated frontier, which is assumed to be known.

When computing the complete NDF, the order in which boxes are processed is immaterial. However, processing the boxes in the queue in nonincreasing order of

their area has two advantages when computing an approximation of the NDF: (1) it often introduces diversification, in the sense that different parts of the criterion will be explored, and (2) as a feature of the BLM design (see Sect. 2.2), the unexplored area of the criterion space reduces as fast as possible. Therefore, terminating the algorithm after a fixed amount of time, or terminating the algorithm when the unexplored area of the criterion space drops below a certain threshold (e.g., below some fraction of the area of the initial box $B(z^L, z^U)$), are both effective strategies to quickly produce a high-quality approximation of the NDF.

6 Computational study

The goal of our computational study is twofold. First, it demonstrates the efficacy of ϵTM-PWL and SPureLex. Second, it investigates SPureLex's ability to efficiently produce high-quality approximations to the NDF. All algorithms are coded in C++ and solve the linear and integer programs using IBM CPLEX Optimizer 12.6. All experiments were conducted in a single thread of a dedicated Intel Xeon ES-2630 2.3 GHz with 50 B RAM, running Red Hat Enterprise Linux Server 7.4. Runtime limit was chosen to be 86,400 s.

6.1 Test instances

While the algorithms presented work for general integer variables, our test instances are restricted to binary variables. Four sets of instances were used: (1) the five largest instances (C320) from the benchmark instances proposed by Mavrotas and Diakoulaki (1998), which we refer to as "Historical", (2) five modified versions of these instances, which we refer to as "Relaxed Historical", (3) three large randomly generated instances using the generation scheme proposed by Perini et al. (2019), which we refer to as "Rand", and (4) three new randomly generated instances, which we refer to as "Bent".

The framework for the Historical instances, originally published by Mavrotas and Diakoulaki (1998), includes: half of the variables are binary and half are continuous; a randomly generated objective vector; and a set of knapsack constraints (the number of which equals the number of variables) with randomly generated coefficients and right hand sides. Each binary variable appears in exactly one knapsack constraint; half of the constraints include only continuous variables. In addition, a constraint ensures that no more than a third of the binary variables can be set to 1. This framework was adapted to the biobjective case in Boland et al. (2015b) by generating an additional objective vector. This set of instances has since been used in many published studies of BOMIP algorithms, including Soylu and Yıldız (2016), Fattahi and Turkay (2018), Soylu (2018), and Perini et al. (2019). We use the largest instances in this set.[10] All instances in the set share similar features: relatively few integer solutions contribute to the NDF, each of which produces a slice with many short line segments, and little dominance between slices. This structure poses complications from an accuracy perspective, due

[10] For reference, the largest instances have 320 constraints and 320 variables, where half of the variables are binary and half are continuous.

to numerical rounding errors (see Fattahi and Turkay 2018; Perini et al. 2019 for further discussion).

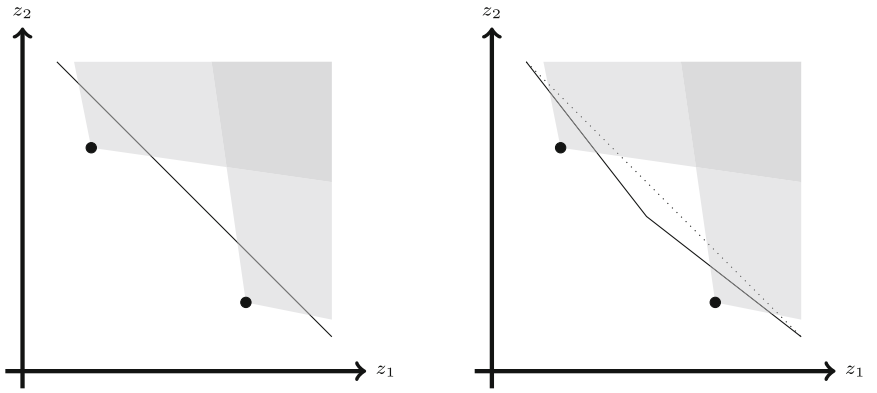
To increase diversity in the benchmark data set, we modify the Historical instances by relaxing the constraints, as follows. First, the constraints involving only continuous variables are removed. Second, the proportion of binary variables that can be set to 1 is increased from a third to a half. These modifications maintain the same number of integer and continuous variables (160 each) while reducing the number of constraints by approximately half. In four of the five instances, this relaxation had the effect of increasing the number of distinct integer solutions whose slice (at least in part) appears in the NDF by about 21% on average, while decreasing the number of NLSs per integer solution in the NDF by between 20–34%, with an average of 26%. In the case of the historical instance numbered 24, its NDF structure barely changed: its ratio of NLSs per integer solution was reduced by less than 2% in the relaxed version. Thus, we omit it. The remaining four instances constitute the set we call Relaxed Historical.

Perini et al. (2019) introduced the instance generation scheme that produces the Rand instances to complement the Historical instances. For example, their ratio of NLSs per integer solution in the NDF is an order of magnitude less than for the Historical instances. The Rand instances were designed with the slices and final NDF in mind, then the BOMIP was "reverse-engineered." The slices in criterion space include: (1) a long line segment traversing from the top left to the bottom right of the NDF and (2) boundaries of cones with vertices that dominate (a point in) the line segment (see Fig. 4a). The NDF alternates between portions of the long line segment and boundaries of each cone, which includes many intersections. Classes of these instances are defined by the number of vertices or cones chosen to dominate the long line segment, which we simply call n. We include three relatively large instances of size $n = 7,500$ in this study, labeled "A", "B", and "C." Each of the n vertices is chosen randomly, so NDFs from instances of the same size still vary.

Here we introduce a slight modification to the structure of the Rand instances that results in significant differences in computational comparison of BOMIP algorithms. The slice consisting of the long line segment is replaced with a "bent" line segment, i.e., two line segments, whose gradients have small difference, joined at a central vertex (see Fig. 4b). Note that this slice is now a cone, but one that is much wider than the cones associated with other integer solutions. Details of the method for generating the Bent instances are given in "Appendix B". Computationally, the slight difference in gradient vectors in the two line segments that form the "bent" line segment makes it much more difficult for algorithms that solve scalarized IPs. Our computational study uses three instances, with $n = 5000$, 7500, and 10,000, respectively.

6.2 Computing exact frontiers

We start by evaluating the benefits of the PWL enhancement to ϵTM. The results are summarized in Table 1. Here, for each set of instances, we report the average and maximum relative increase in running time of the algorithm over that of whichever algorithm is best for the instance: ϵTM or ϵTM-PWL. Specifically, if T_i denotes the running time for ϵTM on the ith instance in a set, and T_i^{PWL} denotes the running time



(a) The Rand instances include one slice in the form of a line segment (bold). For each additional integer solutions, the image of its feasible set is a cone (in gray) that dominates a portion of the line segment.

(b) The Bent instances include a slice that is a "bent" line segment (bold), which is actually the boundary of a very wide cone (i.e., two conjoined line segments).

Fig. 4 Random cone width instances (Perini et al. 2019) and the new Bent instances for $n = 2$

for ϵTM-PWL on the same instance, then in the row of Table 1 for the ϵTM method, we will report

$$\frac{1}{N}\sum_i \frac{T_i - T_i^*}{T_i^*} \quad \text{and} \quad \max_i \frac{T_i - T_i^*}{T_i^*}$$

in the "Average" and "Max" columns corresponding to that set of instances, where N is the number of instances in the set and $T_i^* := \min\{T_i, T_i^{PWL}\}$ is the best of the two runtimes. A zero relative increase indicates that the algorithm was best for every instance in the set; a positive value indicates that the algorithm was not best for at least one instance in the set. A high value indicates that the alternative algorithm gives a speed-up of a factor of one plus this value, on average (e.g., for a value of 1.87, the speed-up factor is 2.87). For the exact running times of the algorithms and other performance metrics, please refer to "Appendix C" (Tables 4, 5).

Recall that the NDF of a Historical instance is composed of many small line segments but is defined by a small number of integer solutions. On average, the number of integer solutions defining the frontier of the Historical instances is about 370, and each contributes about 45 line segments. Hence, we set the hyperparameter K to 50 in our computational experiments. The enhancements were specifically designed to exploit that situation, and explains why the enhanced versions of ϵTM perform so well on this set of instances. The same is true for the Relaxed set of instances, to a lesser extent. In all Historical and Relaxed instances, the enhanced version was faster, with average speed-up factor of around 2–3.

Unfortunately, these impressive gains are not observed for the Rand and Bent instances. Rather than the NDF of these instances being determined by a few integer solutions contributing many line segments, they are determined by many integer

Table 2 Average and maximum relative increase of total running time of the variations of BLM for all instance sets

Algorithm	Average				Max			
	Historical	Relaxed	Rand	Bent	Historical	Relaxed	Rand	Bent
BLM-basic	< 0.01	0.10	0.62	$18.31^{\ddagger}$	< 0.01	0.18	0.65	$33.02^{\ddagger}$
BLM-recursive	0.02	0.01	0.05	$18.11^{\S}$	0.05	0.06	0.08	$32.44^{\S}$
PureLex	0.16	0.07	0.00	0.00	0.24	0.19	0.00	0.00

§—on two instances of the set, the runtime limit of 86,400 s was exceeded. ‡—on three instances of the set, the runtime limit of 86,400 s was exceeded

solutions contributing few line segments. Hence, K was fixed to 1 in our computational experiments. The overhead that comes with setting up and solving an integer program to determine the "left-most" segment of a slice that is nondominated is too large when slices consist of only a few line segments. For the largest Bent instance, the variant of ϵTM with the enhancement is unable to solve the instance within 24 h.

Next, we evaluate the benefits of the variant of BLM in which only lexicographic IPs are solved, PureLex. The results are summarized in Table 2, which again gives statistics for each set of instances for the increase in running time of each variant relative to that of the best BLM variant for the instance.

The PureLex variant is clearly the most robust, being not much slower than the best in the cases where it is not best. For the Rand and Bent instances, the PureLex variant substantially outperforms the basic version. It also outperforms the recursive variant on the Bent instances: BLM-recursive struggles to solve the Bent instances within the 24 h time limit for each, whereas PureLex can solve all of them in less than 3 h. For the Historical and Relaxed instances, the recursive variant of BLM is faster than PureLex, but not by much. Note that BLM-recursive is faster than BLM on all but the Historical instances, and on those is not much slower; BLM-recursive is the most robust of the BLM variants prior to this work.

Finally, we compare the performance of the two best performing variants of ϵTM and BLM with two variants of the hybrid, two-phase method: SPureLex with $\rho = 0.005$ and SPureLex- PWL with $\rho = 0.005$. The results can be found in Table 3.

The first thing to observe is that the PWL enhancement to ϵTM still has a significant impact on the performance of the hybrid, two-phase method, SPureLex, even though ϵTM is only used to find (part of) the NDF in a box that is relatively small (i.e., when the area is less than half a percent of the area of the original box). The second thing to observe is that PureLex performs well on the Historical instance and outperforms SPureLex-0.005, although not by much. Finally, as expected, we see that the version of SPureLex that does not use the enhanced version of ϵTM outperforms the one that does, as the benefits of using the enhancement are insufficient to overcome the overhead incurred when using the enhancement. However, the difference is very small. These results indicate that SPureLex- PWL-0.005, the hybrid, two-phase method that uses PureLex in the first phase and used the enhanced version of ϵTM in the second phase, is the most robust and efficient method for solving BOMIPs.

Table 3 Average and maximum increase in total running time of various solution methods relative to the fastest of the methods for the instance, for all instance sets

Algorithm	Average				Max			
	Historical	Relaxed	Rand	Bent	Historical	Relaxed	Rand	Bent
ϵTM	2.50	2.03	19.26	9.41	2.74	3.89	19.32	12.68
ϵTM-PWL	0.22	0.45	32.23	27.89†	0.29	1.44	32.46	32.12†
BLM-Recursive	1.26	1.13	2.32	57.92§	1.39	2.37	2.41	100.55§
PureLex	1.56	1.20	2.13	2.11	1.81	2.17	2.14	2.16
SPureLex−0.005	2.17	1.89	0.00	0.00	2.42	3.47	0.00	0.00
SPureLex- PWL−0.005	0.00	0.00	0.02	0.02	0.00	0.00	0.03	0.03

†—one instance of the group exceeded the time limit of 86,400 s. §—two instances of the group exceeded the time limit of 86,400 s

It is interesting to observe that, compared to the variants of BLM, the number of times the Same-Integer-Solution enhancement is invoked by the variants of the hybrid, two-phase method is small. This indicates that a more careful tuning of the point at which the hybrid, two-phase methods switches methods may have some pay-off. By switching too early, the algorithm may miss out on opportunities to invoke the Same-Integer-Solution enhancement.

Overall, ϵTM solves the smallest number of IPs for these instances. However, because solving lexicographic IPs for large-area boxes can be costly (as shown in Fig. 3), even though ϵTM solves significantly fewer IPs than PureLex, ϵTM takes more time as it solves several IPs in large-area boxes. This happens because ϵTM generates the nondominated frontier from left-to-right, and solves a lexicographic IP with respect to as-yet unprocessed portion of the line segment.

6.3 Computing approximate frontiers

Next, we compare the performance, in terms of their ability to produce high-quality approximations of the NDF, of ϵTM, the recursive variant of BLM, PureLex, and SPureLex with $\rho = 0.005$. Rather than enforcing early termination of an algorithm to obtain an approximation of the NDF, we report statistics of the approximation of the NDF at different points in time during the execution of the algorithms. We do this for three instances, one from each of the set of Historical, Bent, and Rand instances: 21, Bent7500.A, and Rand7500.A, respectively. The results are representative of what happens for the other instances in the corresponding set. Figure 5 summarizes the following statistics with respect to time (seconds) on a *log* scale: **Area**, the resolved area as a percentage of the initial box $B(z^L, z^R)$; **INDP**, the percentage of isolated NDPs found; **NLS**, the percentage of the NLSs found; **fTNLS**, the fraction of the total length of NLSs found; and, **Slice**, the percentage of slices that contribute to the frontier found. Tables 6, 7, 8 and 9 in "Appendix D" report more statistics.

We observe that, as expected, PureLex and SPureLex produce a high-quality approximation of the NDF much more quickly than ϵTM. In less than 10 min, PureLex and SPureLex have explored more than 99.5% of the area of $B(z^L, z^U)$, whereas ϵTM

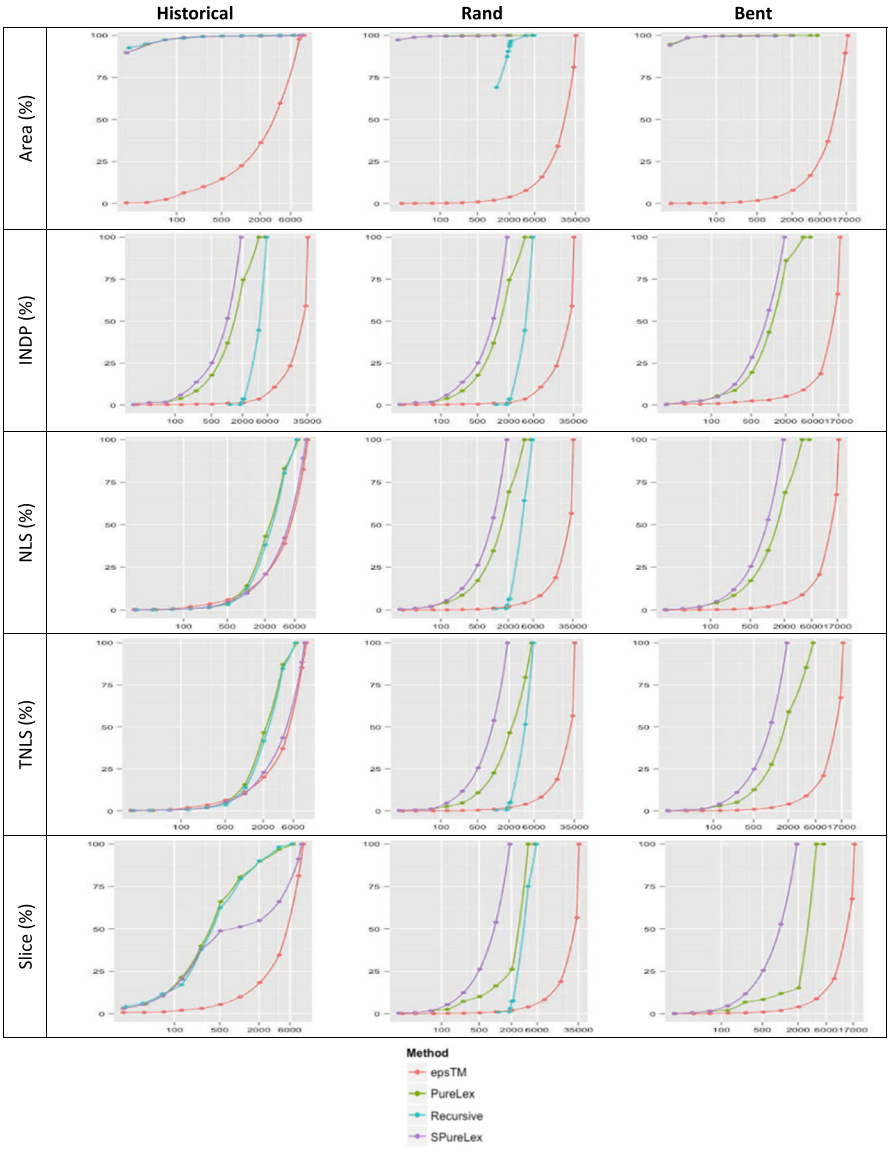


Fig. 5 Approximation results for ϵTM, the recursive variant of BLM, PureLex, and SPureLex with $\rho = 0.005$. Percentages are plotted on the y-axis; time (in s) is plotted on the x-axis on a log scale. There are no isolated NDPs in the NDF of the Historical instance, so this space is blank. No approximation metrics are reported for the recursive variant (blue) of BLM on the Bent instance

has explored a little more than 5% for the Historical instance, and 1% or less for Rand and Bent instances.

Especially for Rand and Bent instances, multiple metrics illustrate how slowly ϵTM advances at the beginning (from an approximation perspective) but speeds up towards the end. This is due, in part, to the fact that in the beginning, the boxes cover a large area in criterion space, and the lexicographic IPs are time-consuming. As the area of the boxes decreases, the lexicographic IPs are solved faster, and, consequently, the statistics improve more rapidly. For the Historical instance, this is less noticeable because the solve times for the lexicographic IPs tend to be smaller (as the generated line segments are small).

Even though the recursive variant of BLM can be competitive when it comes to producing the complete nondominated frontier, it is not the ideal candidate for producing approximations. For the Historical instance, there is never any recursion, so the algorithm produces high-quality approximations throughout the execution. However, on the Rand instance, the first reported data point occurs late during the execution because the algorithm runs deep into recursion and only reports approximation metrics once it has returned to depth level zero. This is also the reason why no approximation metrics are reported for the Bent instance (since the entire execution time is spent on the first recursion without returning to depth level zero). After the algorithm returns to depth level zero for the first time, it quickly approximates the rest of the frontier, but the depth of recursion and resulting lag time of reporting makes the recursive variant of BLM less effective for approximating nondominated frontiers.

Another interesting observation is that PureLex and SPureLex quickly find a large fraction of the slices that contribute to the frontier for the Historical instance. This is a consequence of the structure of the nondominated frontier, in which each slice contributes many nondominated line segments to the frontier, and, because PureLex and SPureLex quickly decompose the criterion space into small boxes, they tend to find NDPs from different slices. Also observe from the Historical instance, however, that once SPureLex switches to ϵTM, the algorithm discovers new slices more slowly, at a rate comparable to ϵTM on the same instance. This is the trade-off: sometimes the switch from PureLex to ϵTM improves performance (e.g., Rand and Bent instances), and sometimes it worsens performance (e.g., Historical instances).

By unifying the two disparate algorithms into a two-phase hybrid approach, with just two hyperparameters to tune (ρ and K), SPureLex is robust to a wider variety of instances than either of the component algorithms. Tuning the first hyperparameter can be informed by some information about the NDF, i.e., how many NLS are expected per slice in the frontier, and/or it can be dynamically updated as the procedure evolves. We found that the second hyperparameter performs well for small values, e.g. $\rho = 0.005$, but this could be more rigorously tested via parameter tuning. Altogether, these results show that also when it comes to finding approximate nondominated frontiers, SPureLex (or one of its variants) is fast and robust and the algorithm of choice.

7 Final remarks

The algorithm presented and analyzed in this paper shows how structural observations of a NDF and computational bottlenecks can lead to tactical enhancements to algorithm design. Evidence included in this work calls for a more diverse range of test instances to test and improve robustness of multiobjective algorithms.

Acknowledgements This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1650044.

Declarations

Conflict of interest This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1650044. The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A: Algorithms

Appendix A.1: Dichotomic search

We present the classic method of dichotomic search as an instructive example to highlight a procedure for solving the slice problem for a fixed integer solution x_I. This procedure was co-discovered and published by Cohon (1978) and Aneja and Nair (1979). See Ehrgott (2005) or Przybylski et al. (2010a) for a detailed summary of the procedure, which is illustrated here and in Fig. 6.

Dichotomic search iteratively solves the scalarized IP (3) to discover all extreme supported ND (ESND) images. At each step of the procedure, a pair of ESND images are compared: either they are certified as adjacent on the convex hull of $\mathcal{Y}$, or a new ESND image is found which proves them to be nonadjacent. The procedure is initialized with the left-most and right-most ND images, say $y^r = \text{lexmin}\{(f_1(x), f_2(x)) : x \in \mathcal{X}\}$ and $y^s = \text{lexmin}\{(f_2(x), f_1(x)) : x \in \mathcal{X}\}$. Then the (positive) gradient vector for the line between y^r and y^s is computed, denoted by λ, and used as the weight vector for the weighted sum scalarization. Figure 6a illustrates this subproblem. One of two cases occur:

First, if the optimal value of (3) with respect to λ is less than $\lambda^T y^r = \lambda^T y^s$, then a new ESND image has been found. Denote the image by y^t. Now, two new pairs of images must be tested for adjacency: (y^r, y^t) and (y^t, y^s); see Fig. 6b. Second, if the optimal value of (3) with respect to λ is equal to $\lambda^T y^r = \lambda^T y^s$, then no new ESND image has been found. The two images, y^r and y^s, are certified as adjacent.

Dichotomic search is generally implemented recursively, and it terminates when the full set of ESND images are found and all adjacencies are certified. The expected output from dichotomic search is the set of line segments belonging to the slice for x_I, i.e., the NDF for the slice problem. In the algorithms that follow, the function denoted `ComputeSlice` is assumed to use dichotomic search.

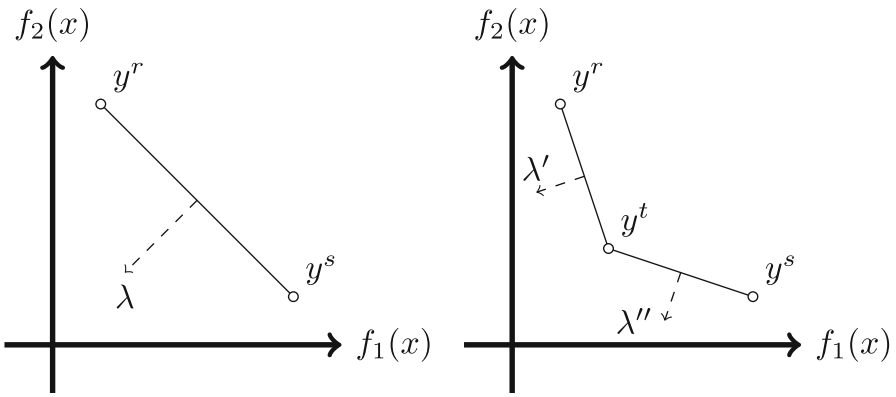


Fig. 6 Dichotomic search for a slice problem, adapted from Przybylski et al. (2010a)

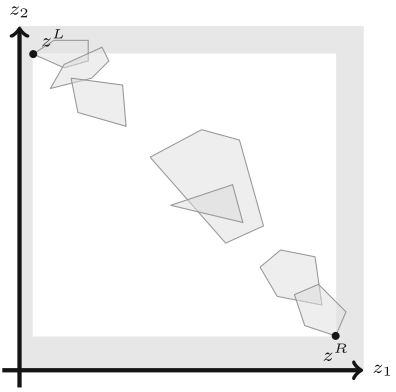
Algorithm 1 ϵ-Tabu Method

$Y_t \leftarrow \emptyset$ the set of all line segments of the slice $\bar{x}^t$
$Y^* \leftarrow \emptyset$ the set of all Pareto line segments
$\bar{x}^1 \leftarrow$ lexmin$\{z_1(x), z_2(x)\}$, find the first solution and slice
$Y_1 \leftarrow$ComputeSlice$(\bar{x}^1)$, via dichotomic search
$t \leftarrow 1$
$l \leftarrow 0$
for each line segment $[y^l, y^{l+1}] \in Y_t$ from left to right **do**
  $\bar{x}^{t+1} \leftarrow$ check if $[y^l, y^{l+1}]$ is dominated by another slice using a modified lexicographic IP with Tabu constraints
  if $\bar{x}^{t+1}$ exists **then**
    $Y_{t+1} \leftarrow$ComputeSlice$(\bar{x}^{t+1})$ starting from z^L_{t+1}, the upper-left corner of the slice x^{t+1} dominating $[y^l, y^{l+1}]$.
    if z^L_{t+1} lies on $[y^l, y^{l+1}]$ **then**
      $Y^* \leftarrow Y^* \cup \{[y^l, z^L_{t+1}]\}$
    else
      $y' \leftarrow$verticalProjection$(z^L_{t+1}, [y^l, y^{l+1}])$
      $Y^* \leftarrow Y^* \cup \{[y^l, y')\}$
    end if
    $t \leftarrow t + 1$
    $l \leftarrow 0$
  else
    $Y^* \leftarrow Y^* \cup \{[y^l, y^{l+1}]\}$
  end if
end for
return Y^*

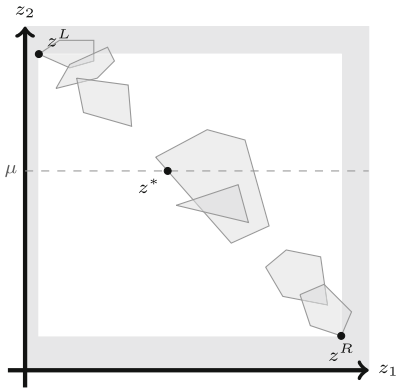
Appendix A.2: Boxed line method

See Fig. 7 for an illustration of the Boxed Line Method.

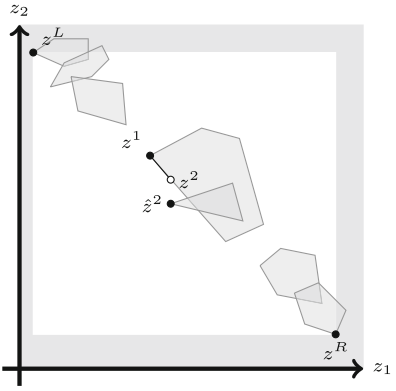
Key component functions include: (1) `element()` returns an element from the input list/set. (2) `LineGen` is a restricted version of dichotomic search that only computes one line segment from the slice (if input is z^*, then it returns a line segment containing z^* from the slice of z^*_l). (3) `LineRestriction.InnerLoop` performs a while loop which restricts the line segment generated from `LineGen` by updating the endpoints when an endpoint is dominated; it returns the ND portion of the line segment containing z^*.



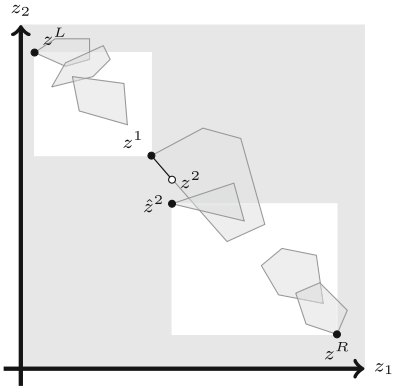
(a) Box $B(z^L, z^R)$ is chosen from Q.



(b) Minimizing z_1 then z_2 below the split line μ yields z^*. Here, $z_2^* = \mu$.



(c) The outer loop calls inner loop's `line restriction` function to compute the NLS containing z^*, indicating whether each endpoint z^1 and z^2 are either open or closed, and any points that dominate an open endpoint (e.g. $\hat{z}^2$ dominates z^2).



(d) The approximation of the ND frontier is updated with line segment $[z^1, z^2]$, and two new boxes are added to the queue, e.g., $B(z^L, z^1)$ and $B(\hat{z}^2, z^R)$ (assuming they are nontrivial).

Fig. 7 Outer loop procedure for BLM, adapted from Perini et al. (2019)

Appendix B: Instance generation

Here we elaborate on the details of the generation of the Bent instances. The nondominated frontier is bounded within $z_i(x) \in [-k, k]$ for $i = 1, 2$, and we choose large enough k, e.g. $k = 1.5 * (n + 1)$, in order to avoid numerical issues. We define the "unbent" line segment as $L = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 + x_2 = 0, -k \leq x_i \leq k \text{ for } i = 1, 2\}$. The NDPs are chosen from a line segment that is shifted downward from L by d, and from left to right, an NDP is chosen a horizontal distance of $2d + 1$ away from the previous NDP in order to avoid the cones from simultaneously dominating a portion of L (this only holds for bounding θ_1, θ_2 as done in the next step).

Algorithm 2 Boxed Line Method (Basic)

procedure LINERESTRICTION.INNERLOOP(z^*, L, R)
  $(z^1, z^2, \vec{w}^T) \leftarrow$ LineGen(z^*, L, R) limited dichotomic search which generates endpoints and gradient of one line segment containing z^* from slice of z_I^*
  $z^\alpha \leftarrow z^1, z^\beta \leftarrow z^2$ label endpoints temporarily
  $\bar{x} \leftarrow \text{argmin}\{\vec{w}^T z(x) : z_2(x) \le z_2^\alpha, z_1(x) \le z_1^\beta\}$ check if NDP dominates line segment
  while $\vec{w}^T z(\bar{x}) < \vec{w}^T z^*$ **do**
    $z(\bar{x})$ dominates a portion of line segment, so update either endpoint z^α or z^β
  end while
  return $(z^\alpha, z^\beta, z^1, z^2)$
end procedure

procedure BOXLINEMETHOD(L, R)
  $N \leftarrow \emptyset$ nondominated line segments
  $Q \leftarrow B(L, R)$ initial region defined by box L, R
  while $Q \neq \emptyset$ (outer loop) **do**
    $B(z^L, z^R) \leftarrow$element(Q)
    $Q \leftarrow$setminus(Q, $B(z^L, z^R)$)
    $\mu \leftarrow (z_2^L, z_2^R)/2$ horizontal line between z^L, z^R
    $z^* \leftarrow \text{lexmin}\{(z_1, z_2) : z_2(x) \le \mu\}$ find NDP in lower half
    if $\mu > z_2^*$ (NDP below horizontal line) **then**
      $Q \leftarrow Q \cup B(z^*, z^R)$ add lower region to the queue
      $\hat{z} \leftarrow \text{lexmin}\{(z_2, z_1) : z_1(x) \le z_1^* - \epsilon\}$
      $Q \leftarrow Q \cup B(z^L, \hat{z})$ add upper region to the queue
    else (NDP on horizontal line)
      $(\hat{z}^1, \hat{z}^2, z^1, z^2) \leftarrow$LineRestriction.InnerLoop(z^*, z^L, z^R)
      $N \leftarrow N \cup [z^1, z^2]$
      $Q \leftarrow Q \cup B\left(\hat{z}^2, z^R\right)$ add lower region to the queue
      $Q \leftarrow Q \cup B\left(z^L, \hat{z}^1\right)$ add upper region to the queue
    end if
  end while
  return N
end procedure

Given corner point (a, b), a cone in criteria space is generally defined by $\{z \in \mathbb{R}^2 : \theta_1 z_1 + (1-\theta_1)z_2 \ge \theta_1 a + (1-\theta_1)b, \theta_2 z_1 + (1-\theta_2)z_2 \ge \theta_2 a + (1-\theta_2)b\}$, where $\theta_1 \in [\frac{3}{4}, 1]$ and $\theta_2 \in [0, \frac{1}{4}]$. Let π be the probability that a cone is orthogonal (i.e., $\theta_1 = 1$ and $\theta_2 = 0$); $U(a, b)$ denotes a uniformly randomly generated value in the open interval (a, b), and we use $\pi = 0.05$.

The "bent" line segment has corner point $(a_0, b_0) = (-d/4, -d/4)$, which may be dominated or nondominated, $\theta_1^0 = (k + d/4)/(2k)$, and $\theta_2^0 = (k - d/4)/(2k)$. Then the bent line segment $\hat{L}$ fits the previous definition for a cone while substituting the corner point (a_0, b_0), θ_1^0, and θ_2^0. Let the generated NDPs be $\{(a_i, b_i)\}_{i=1,2,\ldots,n}$ and their cones be defined by $\{(\theta_1^i, \theta_2^i)\}_{i=1,2,\ldots,n}$. We then have the following BOMILP for the Bent instance:

Algorithm 3 PureLex algorithm

procedure LINERESTRICTION.PURELEX(z^*, L, R)
  $(z^1, z^2, \vec{w}^T, w\_known) \leftarrow$ LineGen(z^*, L, R) to generate endpoints and gradient of one line segment containing z^*
  $M \leftarrow \emptyset$ to hold found NDPs
  $z^\alpha \leftarrow z^1, z^\beta \leftarrow z^2$ to label endpoints temporarily
  if w_known **then**
    $z^\alpha \leftarrow$ lexmin$(z_2(x), z_1(x))$ lexicographic IP (6)
    if $feasible$ **then**
      $M \leftarrow M \cup \{z^\alpha\}$
      $z^1 \leftarrow$horizontalProjection$(z^\alpha, Line(z^1, z^*))$
      $z1\_open$
    end if
    $z^\beta \leftarrow$ lexmin$(z_1(x), z_2(x))$ lexicographic IP (7)
    if $feasible$ **then**
      $M \leftarrow M \cup \{z^\beta\}$
      $z^2 \leftarrow$verticalProjection$(z^\beta, Line(z^*, z^2))$
      $z2\_open$
    end if
  end if
  return $(z^\alpha, z^\beta, z^1, z^2, M)$
end procedure

procedure BOXLINEMETHOD(L, R)
  $N \leftarrow \emptyset$ nondominated line segments
  $Q \leftarrow B(L, R)$ initial region defined by box L, R
  while $Q \neq \emptyset$ **do**
    $B(z^L, z^R) \leftarrow$element(Q)
    $Q \leftarrow$setminus$(Q, B(z^L, z^R))$
    $\mu \leftarrow (z_2^L, z_2^R)/2$ horizontal line between z^L, z^R
    $z^* \leftarrow$ lexmin$\{(z_1, z_2) : z_2(x) \leq \mu\}$ find NDP in lower half
    if $\mu - z_2^* > \epsilon$ **then**
      $Q \leftarrow Q \cup B(z^*, z^R)$ add lower region to the queue
      $\hat{z} \leftarrow$ lexmin$\{(z_2, z_1) : z_1(x) \leq z_1^* - \epsilon\}$
      $Q \leftarrow Q \cup B(z^L, \hat{z})$ add upper region to the queue
    else
      $(\hat{z}^1, \hat{z}^2, z^1, z^2, M) \leftarrow$LineRestriction.PureLex$(z^*, z^L, z^R)$
      $N \leftarrow N \cup M \cup line(z^1, z^2)$
      $Q \leftarrow Q \cup B\left(\hat{z}^2, z^R\right)$ add lower region to the queue
      $Q \leftarrow Q \cup B\left(z^L, \hat{z}^1\right)$ add upper region to the queue
    end if
  end while
  return N
end procedure

$$\text{minimize} \qquad (x_1, x_2) \tag{8}$$

$$\text{s.t.} \quad \theta_1^i x_1 + (1-\theta_1^i)x_2 \geq \theta_1^i a_i + (1-\theta_1^i)b_i - 2k(1-y_i) \qquad \forall i \tag{9}$$

$$\theta_2^i x_1 + (1-\theta_2^i)x_2 \geq \theta_2^i a_i + (1-\theta_2^i)b_i - 2k(1-y_i) \qquad \forall i \tag{10}$$

$$\sum_{i=0}^{n} y_i = 1 \tag{11}$$

$$-k \leq x_i \leq k \qquad \forall i = 1, 2 \tag{12}$$

$$y \in \{0, 1\}^{n+1}. \tag{13}$$

Algorithm 4 SPURELEX algorithm

procedure PHASETWO(Q)
  $N' \leftarrow \emptyset$
  while $Q \neq \emptyset$ **do**
    $B(z^L, z^R) \leftarrow$element(Q)
    $Q \leftarrow$setminus($Q, B(z^L, z^R)$)
    $N' \leftarrow N' \cup \epsilon$TM($B(z^L, z^R)$) finds NDF with ϵTM or enhanced ϵTM
  end while
  return N'
end procedure

procedure PHASEONE(L, R)
  $N \leftarrow \emptyset$ nondominated line segments
  $Q \leftarrow B(L, R)$ initial region defined by box L, R
  $totalVolume \leftarrow volume(B(L, R))$
  $unsolved \leftarrow totalVolume$
  while $Q \neq \emptyset$ **do**
    $B(z^L, z^R) \leftarrow element(Q)$ is the box with largest volume
    if $(unsolved - volume(B(z^L, z^R)))/totalVolume \leq \rho$ **then**
      $N \leftarrow N \cup$ PhaseTwo(Q)
      return N
    end if
    $Q \leftarrow$setminus($Q, B(z^L, z^R)$)
    $unsolved \leftarrow unsolved - volume(B(z^L, z^R))$
    $\mu \leftarrow (z_2^L, z_2^R)/2$ horizontal line between z^L, z^R
    $z^* \leftarrow$ lexmin$\{(z_1, z_2) : z_2(x) \leq \mu\}$ find NDP in lower half
    if $\mu - z_2^* > \epsilon$ **then**
      $Q \leftarrow Q \cup B(z^*, z^R)$ add a new region to the queue
      $\hat{z} \leftarrow$ lexmin$\{(z_2, z_1) : z_1(x) \leq z_1^* - \epsilon\}$
      $Q \leftarrow Q \cup B(z^L, \hat{z})$
    else
      $(\hat{z}^1, \hat{z}^2, z^1, z^2, M) \leftarrow$LineRestriction.PureLex($z^*, z^L, z^R$)
      $N \leftarrow N \cup M \cup line(z^1, z^2)$
      $Q \leftarrow Q \cup B\left(z^L, \hat{z}^1\right)$ add to queue
      $Q \leftarrow Q \cup B\left(\hat{z}^2, z^R\right)$ add to queue
    end if
  end while
  return N
end procedure

Algorithm 5 Randomized Cone-Width NDP Generation

$d = k/(n+1) - 0.5$
$a_1 = -k + 0.5d$
$b_1 = -a_1 - d$
for $i = 2, ..., n$ **do**
  $a_i = a_{i-1} + 2d + 1$
  $b_i = -a_i - d$
end for

Algorithm 6 Randomized Theta Generation

thetalist= $\emptyset$
for $i = 1, 2, ..., n$ **do**
  if $U(0, 1) \leq \pi$ **then**
    $\theta_1 = 1$
    $\theta_2 = 0$
  else
    $\theta_1 = U(\frac{3}{4}, 1)$
    $\theta_2 = U(0, \frac{1}{4})$
  end if
  thetalist.append((θ_1, θ_2))
end for

Appendix C: Computational results

In Tables 4 and 5, we report the following statistics: the number of NDPs output by the algorithm (**nNDP**), the number of different integer solutions, i.e., the number of different slices appearing in the NDF (**nIPF**), the total time to discover the NDF (**TT**), the total time spent solving IPs (**IPT**), the total time spent solving LPs (**LPT**), the total number of IPs solved (**nIP**), the total number of lexicographic IPs (**nLex**), the total number of IPs solved during line restriction, i.e., as part of solving solving (6) and (7) in PureLex and SPureLex and as part of solving (7) in our implementation of ϵTM, (**RLIP**); the total number of scalarized IPs solved (**nScal**), the total number of IPs solved as part of the same-integer-solution enhancement, i.e., one such IP is solved per box with corner points generated by the same integer solution, (**nSIS**), the total number of LPs solved (**nLP**), the number of boxes processed (**nBox**), and, finally, the number of boxes with z^* on the horizontal split line, (**nZL**). The best time per instance is in bold. When not applicable, an entry in the tables is marked with "–".

Appendix D: Approximation results

In Tables 6, 7, 8, and 9, we report the following statistics: **TP**, the point in time (during the execution of the algorithm) that the statistics were collected; **fINDP**, the fraction of the number of isolated NDPs found; **fNLS**, the fraction of the number of nondominated line segments found; **fTNLS**, the fraction of the total length of nondominated line segments found; **fA**, the resolved area of the initial box $B(z^L, z^R)$ as a fraction; and, **fSlice**, the fraction of slices that contribute to the frontier found. When not applicable, an entry in the tables is marked with "–".

Table 4 Computational results for various solution methods

Instance		Algorithm	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	RLIP	nScal	nSIS	nLP	nBox	nZL
Historical	21		15,636	295	10050.9	7540.9	2505.9	15,923	43	15,837	0	0	86,952	1	0
	22		18,825	410	12925.5	10011.8	2909.0	19,205	24	19,157	0	0	98,130	1	0
	23	ϵTM	16,420	343	11192.4	8264.4	2923.6	16,752	25	16,702	0	0	102,407	1	0
	24		18,471	457	13834.5	10859.3	2971.3	18,968	36	18,896	0	0	101,213	1	0
	25		13,216	337	8365.9	6085.1	2278.0	13,518	21	13,476	0	0	79,127	1	0
	21		15,636	295	3370.1	871.1	2495.5	634	43	548	0	0	86,952	1	0
	22		18,825	410	4114.2	1056.6	3053.7	846	24	798	0	0	98,130	1	0
	23	ϵTM-PWL	16,420	343	4062.1	1051.5	3006.4	715	25	665	0	0	102,407	1	0
	24		18,507	460	4829.1	1726.5	3098.6	1178	36	1106	0	0	101,944	1	0
	25		13,216	337	3202.3	877.1	2322.6	697	21	655	0	0	79,127	1	0
	21		15,725	294	6462.0	5014.4	1446.4	6552	1374	0	2737	1067	45,193	2427	1353
	22		18,856	410	7820.4	6093.7	1725.4	8221	1772	0	3292	1385	53,663	3123	1716
	23	BLM-Recursive	16,463	343	7201.1	5595.2	1604.6	7097	1498	0	2959	1142	47,527	2626	1475
	24		18,563	460	9512.4	7625.7	1885.2	9635	2063	0	3937	1572	57,480	3621	2069
	25		13,248	337	5482.0	4261.4	1219.5	6417	1311	0	2786	1009	39,407	2288	1256
	21		15,685	294	7025.3	5649.8	1374.2	7918	1598	3493	0	1229	41,643	2813	1577
	22		18,824	410	8933.6	7287.5	1644.4	10,126	2046	4463	0	1571	49,956	3583	1990
	23	PureLex	16,439	343	7951.3	6382.1	1567.7	8656	1737	3859	0	1323	43,878	3047	1714
	24		18,526	460	11859.4	10033.4	1824.1	12,708	2464	5816	0	1964	54,459	4414	2515
	25		13,230	337	5970.9	4832.2	1137.4	7604	1533	3351	0	1187	35,526	2688	1478
	21		15,652	295	8964.6	7585.0	1376.3	16,046	196	15,642	0	12	44,467	295	148
	22		18,812	410	11837.0	9953.2	1880.0	19,618	177	19,262	0	2	61,637	298	145
	23	SpureLex−0.005	16,428	343	10100.1	8397.5	1699.1	17,145	166	16,811	0	2	50,220	280	139
	24		18,457	457	12841.0	10829.5	2007.5	19,234	191	18,845	0	7	62,806	298	147

Table 4 continued

Instance		Algorithm	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	RLIP	nScal	nSIS	nLP	nBox	nZL
	25		13,216	337	7370.1	6090.6	1276.7	13,839	172	13,487	0	8	43,122	287	139
	21		15,652	295	**2701.0**	1193.9	1505.2	1522	196	1118	0	12	44,467	295	148
	22		18,812	410	**3460.2**	1430.1	2027.7	1731	177	1375	0	2	61,637	298	145
	23	SPURELEX- PWL−0.005	16,428	343	**3159.2**	1296.0	1861.1	1550	166	1216	0	2	50,220	280	139
	24		18,482	460	**4216.2**	1961.6	2251.8	2056	191	1667	0	7	63,157	298	147
	25		13,216	337	**2586.3**	1146.8	1437.8	1554	172	1202	0	8	43,122	287	139
Relaxed historical	N-21		14,902	367	6529.1	4510.9	2014.6	15,249	14	15,221	0	0	93,364	1	0
	N-22		15,502	426	6054.9	4229.6	1821.6	15,906	25	15,856	0	0	86,955	1	0
	N-23	ϵTM	15,162	490	7102.3	4650.0	2448.5	15,643	15	15,613	0	0	110,082	1	0
	N-24		15,480	392	7017.7	5358.5	1655.5	15,879	36	15,807	0	0	70,351	1	0
	N-25		11,547	387	4650.3	3380.9	1266.5	11,919	30	11,859	0	0	60,953	1	0
	N-21		14,902	367	3256.9	733.3	2519.9	753	14	725	0	0	93,364	1	0
	N-22		15,502	426	2867.8	674.6	2189.9	880	25	830	0	0	86,955	1	0
	N-23	ϵTM-PWL	15,168	491	3512.4	890.2	2618.3	1021	15	991	0	0	110,333	1	0
	N-24		15,501	394	3006.4	898.2	2104.9	883	36	811	0	0	70,681	1	0
	N-25		11,550	387	2283.9	709.3	1572.0	826	30	766	0	0	60,957	1	0
	N-21		14,913	366	4499.6	3307.8	1190.0	6670	1416	0	2754	1084	43,014	2471	1368
	N-22		15,521	425	4303.4	3156.9	1144.5	8250	1675	0	3585	1315	47,103	2969	1642
	N-23	BLM-recursive	15,183	491	5044.3	3837.3	1205.1	9172	1949	0	3842	1432	48,849	3366	1931
	N-24		15,545	392	5013.9	3719.6	1292.4	7734	1596	0	3327	1215	47,182	2799	1588
	N-25		11,560	387	3953.8	3035.6	916.4	6814	1452	0	2862	1048	36,037	2478	1414
	N-21		14,896	366	4234.4	3377.3	855.5	8545	1724	3827	0	1270	40,228	2965	1679
	N-22		15,503	425	4710.2	3688.0	1020.3	9690	1931	4328	0	1500	42,186	3410	1899

Table 4 continued

Instance	Algorithm	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	RLIP	nScal	nSIS	nLP	nBox	nZL
N-23	PureLex	15,164	491	6021.2	4891.1	1127.9	11,983	2383	5485	0	1732	45,136	4101	2375
N-24		15,513	393	5009.1	4041.4	966.0	9964	1952	4568	0	1492	43,713	3433	1954
N-25		11,547	387	4211.6	3450.6	759.2	8744	1741	3981	0	1281	33,145	3004	1706
N-21		14,886	367	5965.0	4955.0	1006.5	15,397	178	15,031	0	10	46,496	314	156
N-22		15,487	426	6405.0	5131.1	1270.1	16,315	193	15,925	0	4	53,921	334	168
N-23	SPureLex−0.005	15,147	490	6377.2	5058.8	1314.7	16,222	173	15,875	0	1	54,682	317	158
N-24		15,466	392	6925.4	5742.8	1178.9	16,335	193	15,945	0	4	49,052	313	155
N-25		11,553	387	4439.5	3684.4	752.2	12,420	181	12,055	0	3	36,623	298	145
N-21		14,889	367	**1335.2**	636.1	697.9	1742	178	1376	0	10	45,941	314	156
N-22		15,470	426	**2524.8**	1029.9	1492.3	1962	197	1564	0	4	51,608	334	168
N-23	SPureLex- PWL−0.005	15,149	491	**3020.3**	1322.4	1695.3	2011	173	1664	0	1	53,327	317	158
N-24		15,476	394	**2823.0**	1295.5	1524.9	1883	193	1493	0	4	47,756	313	155
N-25		11,549	387	**1996.9**	989.3	1005.5	1730	181	1365	0	3	36,328	298	145

Table 5 Computational results for various solution methods for the Rand and Bent instances

Instance		Algorithm	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	RLIP	nScal	nSIS	nLP	nBox	nZL
Rand	A		22,502	7501	36296.0	35481.4	675.1	37,141	369	36,403	0	1	36,403	0	0
	B	ϵTM	22,508	7501	36222.6	35415.6	669.5	37,138	382	36,374	0	1	36,365	0	0
	C		22,506	7501	36086.3	35286.2	663.6	37,138	381	36,376	0	1	36,370	0	0
	A		22,501	7501	58944.1	58124.7	668.2	30,010	369	29,272	0	1	36,403	0	0
	B	ϵTM-PWL	22,500	7501	59383.5	58563.4	670.1	30,013	382	29,249	0	1	36,365	0	0
	C		22,502	7501	59776.6	58910.1	710.7	30,011	381	29,249	0	1	36,370	0	0
	A		22,502	7501	6101.1	3265.4	2648.3	44,513	5303	0	33,907	5275	110,266	1	5249
	B	BLM-Recursive	22,502	7501	5898.9	3088.8	2623.6	44,530	5381	0	33,768	5343	110,205	1	5307
	C		22,504	7501	5778.7	3004.2	2589.2	44,493	5259	0	33,975	5228	110,033	1	5199
	A		22,502	7501	5600.5	2667.8	2727.5	116,515	21,896	72,723	0	21,834	115,994	7954	21,766
	B	PURELEX	22,502	7501	5590.9	2661.6	2722.4	116,357	21,850	72,657	0	21,797	115,849	7939	21,742
	C		22,502	7501	5603.5	2663.7	2732.3	116,375	21,849	72,677	0	21,797	115,879	7960	21,747
	A		22,502	7501	**1790.9**	865.2	821.7	38,357	583	37,191	0	434	38,084	40	213
	B	SPURELEX−0.005	22,502	7501	**1782.6**	863.7	816.3	38,353	594	37,165	0	434	38,042	42	215
	C		22,502	7501	**1786.6**	866.0	813.1	38,354	597	37,160	0	434	38,032	46	211
	A		22,502	7501	1832.3	904.3	810.0	31,343	583	30,177	0	434	38,084	40	213

Table 5 continued

Instance		Algorithm	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	RLIP	nScal	nSIS	nLP	nBox	nZL
	B	SPureLex- PWL−0.005	22,502	7501	1820.3	901.0	804.3	31,351	594	30,163	0	434	38,042	42	215
	C		22,502	7501	1837.8	909.7	810.4	31,346	597	30,152	0	434	38,032	46	211
Bent	5000.A		15,003	5001	6004.5	5605.0	339.9	24,772	239	24,294	0	0	24,295	1	0
	7500.A	ϵTM	22,502	7501	18920.8	18058.4	716.8	37,142	370	36,402	0	0	36,403	1	0
	10000.A		30,004	10,001	42413.0	40947.7	1189.7	49,528	490	48,548	0	0	48,546	1	0
	5000.A		15,003	5001	21483.9	21105.3	315.4	20,056	239	19,578	0	0	24,295	1	0
	7500.A	ϵTM-PWL	22,502	7501	60424.9	59557.6	707.7	30,009	370	29,269	0	0	36,403	1	0
	10000.A		–	–	–	–	–	–	–	–	–	–	–	–	–
	5000.A		15,003	5001	84915.2	83826.7	986.1	29,559	4996	0	19,567	2	69,979	4994	4994
	7500.A	BLM-Recursive	–	–	–	–	–	–	–	–	–	–	–	–	–
	10000.A		–	–	–	–	–	–		–	–	–	–	–	–
	5000.A		15,003	5001	2539.1	1199.5	1248.6	77,607	14,620	48,367	0	5875	77,499	14,578	14,532
	7500.A	PureLex	22,502	7501	5764.6	2701.7	2856.4	116,393	21,830	72,733	0	8512	115,969	21,798	21,760
	10000.A		30,002	10,001	9681.7	4527.0	4810.1	155,070	29,202	96,666	0	11,660	154,872	29,126	29,044
	5000.A		15,003	5001	**836.2**	402.1	386.8	25,988	449	25,090	0	39	25,969	428	212
	7500.A	SpureLex−0.005	22,502	7501	**1824.5**	882.5	838.0	38,350	586	37,178	0	45	38,041	434	208
	10000.A		30,002	10,001	**3100.2**	1504.2	1416.8	50,711	704	49,303	0	52	50,286	436	216
	5000.A		15,003	5001	841.9	413.4	376.3	21,300	449	20,402	0	39	25,969	428	212
	7500.A	SPureLex- PWL−0.005	22,502	7501	1876.8	923.9	835.5	31,322	586	30,150	0	45	38,041	434	208
	10000.A		30,002	10,001	3206.9	1583.1	1421.9	41,319	704	39,911	0	52	50,286	436	216

Table 6 Approximation results for ϵTM

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
21	16.26	16.26	99.582673	–	0.24	0.30	0.42	0.68
	34.10	34.10	99.433120	–	0.33	0.38	0.57	0.68
	68.34	68.34	97.547610	–	0.66	0.77	2.45	1.02
	128.22	128.22	93.596498	–	1.74	1.97	6.40	2.03
	263.72	263.73	89.951496	–	3.33	3.52	10.05	3.05
	512.45	512.46	85.318206	–	5.88	6.46	14.68	5.42
	1024.26	1024.29	77.467321	–	11.02	11.19	22.53	9.83
	2057.03	2057.08	63.777420	–	20.82	20.08	36.22	18.31
	4096.58	4096.68	40.377154	–	38.86	37.06	59.62	34.58
	8192.29	8192.50	2.248251	–	82.51	85.20	97.75	81.36
	9690.87	9691.13	0.000000	–	100.00	100.00	100.00	100.00
Bent7500.A	16.77	16.77	99.977826	0.55%	0.02	0.01	0.02	0.03
	34.54	34.54	99.909084	0.55%	0.05	0.04	0.09	0.05
	64.12	64.12	99.800146	0.55%	0.11	0.10	0.20	0.11
	130.20	130.21	99.575419	0.82%	0.22	0.21	0.42	0.23
	258.36	258.37	99.152935	1.64%	0.44	0.41	0.85	0.44
	514.80	514.82	98.225619	2.46%	0.90	0.88	1.77	0.91

Table 6 continued

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
	1027.33	1027.38	96.304858	3.01%	1.88	1.88	3.70	1.88
	2049.65	2049.76	92.163824	5.19%	4.01	4.05	7.84	4.01
	4099.85	4100.13	83.302600	9.02%	8.74	8.87	16.70	8.75
	8192.22	8192.96	63.073380	18.58%	20.59	20.86	36.93	20.60
	16384.52	16386.97	10.473017	66.12%	67.65	67.40	89.53	67.64
	17977.38	17981.29	0.000000	100.00%	100.00	100.00	100.00	100.00
Rand7500.A	18.85	18.85	99.979689	0.27%	0.02	0.01	0.02	0.03
	33.15	33.15	99.953675	0.27%	0.04	0.02	0.05	0.04
	70.83	70.83	99.887789	0.27%	0.07	0.06	0.11	0.07
	131.37	131.37	99.768739	0.27%	0.13	0.12	0.23	0.13
	259.62	259.62	99.539288	0.55%	0.24	0.23	0.46	0.24
	512.96	512.97	99.061224	0.55%	0.48	0.47	0.94	0.48
	1028.21	1028.23	98.098272	1.09%	0.96	0.96	1.90	0.97
	2050.24	2050.29	96.153765	1.37%	1.96	1.96	3.85	1.96
	4100.09	4100.20	92.280887	3.55%	3.95	3.95	7.72	3.95
	8195.05	8195.32	84.232517	10.66%	8.23	8.18	15.77	8.24
	16385.06	16385.68	65.910199	23.22%	18.82	18.74	34.09	18.82
	32768.88	32771.07	18.845577	59.02%	56.60	56.54	81.15	56.61
	35789.02	35793.24	0.000000	100.00%	100.00	100.00	100.00	100.00

Table 7 Approximation results for PURELEX

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
21	16.22	16.22	2.060236	–	0.10	0.22	89.76	3.06
	36.15	36.15	0.694905	–	0.20	0.32	94.89	5.78
	65.88	65.88	0.165257	–	0.39	0.49	97.41	10.88
	128.38	128.38	0.043796	–	0.79	0.98	98.73	21.43
	257.16	257.17	0.010580%	–	1.53	1.94	99.38	39.80
	513.99	514.00	0.002459	–	3.86	5.01	99.71	65.99
	1024.45	1024.47	0.000749	–	14.03	15.68	99.86	80.61
	2049.89	2049.96	0.000172	–	43.22	46.53	99.96	89.80
	4100.56	4100.71	0.000011	–	82.93	87.12	100.00	96.94
	6777.20	6777.46	0.000000	–	100.00	100.00	100.00	100.00
Bent7500.A	16.00	16.01	0.389899	0.27%	0.19	0.13	94.74	0.11
	32.01	32.02	0.024330	1.64%	0.71	0.46	98.62	0.47
	64.08	64.11	0.006002	2.46%	1.78	1.07	99.46	1.45
	128.03	128.10	0.001483	5.46%	3.94	2.89	99.77	1.95
	256.12	256.30	0.000348	8.74%	8.41	5.09	99.90	6.76
	512.01	512.40	0.000074	19.40%	17.09	12.59	99.96	8.32
	1024.16	1025.00	0.000016	43.44%	34.72	27.62	99.99	11.81
	2048.14	2049.95	0.000002	86.07%	68.92	58.97	100.00	15.17
	4096.22	4100.11	0.000000	100.00%	100.00	85.28	100.00	100.00
	5442.12	5447.43	0.000000	100.00%	100.00	100.00	100.00	100.00

Table 7 continued

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
Rand7500.A	16.06	16.06	0.097490	0.27%	0.35	0.20	97.28	0.27
	32.26	32.27	0.024260	1.37%	0.87	0.54	98.87	0.55
	64.25	64.29	0.005985	1.64%	1.94	1.09	99.51	1.68
	128.19	128.27	0.001476	3.83%	4.10	2.55	99.78	2.53
	256.19	256.38	0.000348	8.47%	8.48	4.77	99.90	7.28
	512.00	512.44	0.000080	17.76%	17.16	10.81	99.96	10.05
	1024.18	1025.11	0.000016	36.89%	34.60	22.61	99.98	16.37
	2048.03	2049.93	0.000002	74.59%	69.34	46.47	100.00	26.13
	4096.02	4099.76	0.000000	100.00%	100.00	79.49	100.00	100.00
	5420.87	5426.26	0.000000	100.00%	100.00	100.00	100.00	100.00

Table 8 Approximation results for SPURELEX with $\rho = 0.005$

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
21	16.57	16.57	2.060236	–	0.10	0.22	89.76	3.05
	32.46	32.46	0.752281	–	0.19	0.31	94.51	5.42
	65.80	65.80	0.173392	–	0.38	0.48	97.33	10.51
	129.53	129.53	0.048528	–	0.75	0.90	98.65	20.34
	256.90	256.90	0.012639	–	1.44	1.85	99.34	37.97
	512.32	512.33	0.006956	–	4.45	5.37	99.51	48.81
	1024.33	1024.36	0.006956	–	9.61	10.29	99.53	51.19
	2048.02	2048.08	0.006956	–	21.08	22.94	99.58	54.92
	4096.39	4096.52	0.006956	–	42.20	43.49	99.66	66.10
	8192.23	8192.51	0.006956	–	89.11	88.52	99.91	91.19
	9120.40	9120.72	0.000000	–	100.00	100.00	100.00	100.00
Bent7500.A	16.07	16.07	0.389944	0.27%	0.16	0.12	94.16	0.07
	32.07	32.08	0.024330	1.64%	0.66	0.42	98.56	0.47
	64.24	64.27	0.006009	2.46%	1.67	1.00	99.42	1.37
	128.10	128.27	0.005985	4.92%	4.92	4.13	99.51	4.61
	256.09	256.57	0.005985	12.30%	11.79	11.05	99.54	11.64
	512.00	513.25	0.005985	28.42%	25.53	24.91	99.59	25.50
	1024.00	1026.81	0.005985	56.56%	52.93	52.62	99.70	52.73
	1907.78	1913.49	0.000000	100.00%	100.00	100.00	100.00	100.00

Table 8 continued

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
Rand7500.A	16.13	16.14	0.097490	0.27%	0.35	0.20	97.28	0.27
	32.05	32.07	0.024260	1.37%	0.86	0.53	98.85	0.55
	64.01	64.04	0.005986	1.64%	1.92	1.08	99.50	1.67
	128.01	128.19	0.005986	6.01%	5.46	4.62	99.52	5.28
	256.01	256.52	0.005986	13.66%	12.47	11.68	99.54	12.36
	512.00	513.31	0.005986	25.14%	26.22	25.62	99.60	26.26
	1024.10	1027.19	0.005986	51.64%	54.05	53.79	99.71	53.83
	1866.35	1872.18	0.000000	100.00%	100.00	100.00	100.00	100.00%

Table 9 Approximation results for the recursive variant of BLM

Instance	ST	TP	M1 (%)	fINDP	fNLS (%)	fTNLS (%)	fA (%)	fSlice (%)
21	18.03	18.03	1.209597	–	0.14	0.25	92.50	4.08
	32.76	32.76	0.694905	–	0.22	0.34	94.89	6.12
	65.46	65.46	0.149381	–	0.42	0.52	97.50	11.56
	129.23	129.23	0.070827	–	0.73	0.77	98.29	17.01
	257.06	257.06	0.014935	–	1.53	1.81	99.27	37.76
	512.90	512.90	0.003389	–	3.01	3.58	99.66	62.59
	1056.77	1056.78	0.000906	–	12.11	13.85	99.84	79.59
	2049.07	2049.14	0.000249	–	38.11	41.63	99.95	90.14
	4096.38	4096.55	0.000016	–	80.31	84.72	100.00	98.30
	6482.32	6482.60	0.000000	–	100.00	100.00	100.00	100.00
Rand7500.A	1151.45	1151.47	25.000000	0.27%	0.76	0.55	68.98	0.93
	1835.93	1835.96	4.246974	0.27%	0.87	0.60	87.42	1.09
	1916.55	1916.61	4.028313	1.37%	2.55	1.95	90.50	3.05
	2042.68	2043.04	1.496833	3.55%	6.11	4.87	93.50	7.16
	2071.65	2071.84	1.256458	3.55%	6.19	4.91	94.50	7.28
	2096.62	2096.83	1.061497	3.55%	6.26	4.94	95.34	7.37
	2119.79	2120.00	1.006409	3.55%	6.34	4.98	96.05	7.49
	2132.32	2132.56	0.773759	3.55%	6.43	5.02	96.74	7.61
	4096.03	4098.84	0.000016	44.54%	64.17	51.40	99.99	75.06
	5821.14	5826.61	0.000000	100.00%	100.00	100.00	100.00	100.00

References

Aneja YP, Nair KP (1979) Bicriteria transportation problem. Manag Sci 25:73–78

Belotti P, Soylu B, Wiecek MM (2013) A branch-and-bound algorithm for biobjective mixed-integer programs. Optim Online

Boland N, Charkhgard H, Savelsbergh M (2015a) A criterion space search algorithm for biobjective integer programming: the balanced box method. INFORMS J Comput 27:735–754

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

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

Boland N, Charkhgard H, Savelsbergh M (2017) The quadrant shrinking method: a simple and efficient algorithm for solving tri-objective integer programs. Eur J Oper Res 260:873–885

Cabrera-Guerrero G, Ehrgott M, Mason AJ, Raith A (2021) Bi-objective optimisation over a set of convex sub-problems. Ann Oper Res, 1–26

Ceyhan G, Köksalan M, Lokman B (2023) Finding the nondominated set and efficient integer vectors for a class of three-objective mixed-integer linear programs. Manag Sci

Chankong V, Haimes YY (2008) Multiobjective decision making: theory and methodology. Courier Dover Publications, Mineola

Cohon JL (1978) Multiobjective programming and planning. Academic Press, Cambridge

Dai R, Charkhgard H (2018) A two-stage approach for bi-objective integer linear programming. Oper Res Lett 46:81–87

Dächert K, Klamroth K (2015) A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. J Glob Optim 61:643–676

Diessel E (2022) An adaptive patch approximation algorithm for bicriteria convex mixed-integer problems. Optimization 71:4321–4366

Ehrgott M (2005) Multicriteria optimization, 2nd edn. Springer, Berlin

Ehrgott M, Gandibleux X (2000) A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spectr 22:425–460

Eichfelder G, Gerlach T, Warnow L (2023a) A test instance generator for multiobjective mixed-integer optimization

Eichfelder G, Stein O, Warnow L (2023b) A solver for multiobjective mixed-integer convex and nonconvex optimization. J Optim Theory Appl 1–31

Emre D (2020) Exact solution algorithms for biobjective mixed integer programming problems. Ph.D. Thesis, Bilkent University

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 266:415–425

Gandibleux X (2006) Multiple criteria optimization: state of the art annotated bibliographic surveys, vol 52. Springer, Berlin

Geoffrion AM (1968) Proper efficiency and the theory of vector maximization. J Math Anal Appl 22:618–630

Haimes Y (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans Syst Man Cybern 1:296–297

Halffmann P, Schäfer LE, Dächert K, Klamroth K, Ruzika S (2022) Exact algorithms for multiobjective linear optimization problems with integer variables: a state of the art survey. J Multi-Criteria Decis Anal 29:341–363

Herszterg I (2020) Efficient algorithms for solving multi-objective optimization and large-scale transportation problems. Ph.D. Thesis, Georgia Institute of Technology

Kirlik G, Sayın S (2014) A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. Eur J Oper Res 232:479–488

Klamroth K, Lacour R, Vanderpooten D (2015) On the representation of the search region in multi-objective optimization. Eur J Oper Res 245:767–778

Mavrotas G, Diakoulaki D (1998) A branch and bound algorithm for mixed zero-one multiple objective linear programming. Eur J Oper Res 107:530–541

Pareto V (1996) Cours d'Economie Politique Professé a l'Université de Lausanne

Pecin D, Herszterg I, Perini T, Boland N, Savelsbergh M (2022) BOMIP GitHub project. https://github.com/dppecin/BOMIP

Perini T, Boland N, Pecin D, Savelsbergh M (2019) A criterion space method for biobjective mixed integer programming: the boxed line method. INFORMS J Comput 32:16–39
Pettersson W, Ozlen M (2019) Multi-objective mixed integer programming: an objective space algorithm. In: AIP conference proceedings, vol 2070. AIP Publishing
Przybylski A, Gandibleux X, Ehrgott M (2010a) A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. INFORMS J Comput 22:371–386
Przybylski A, Gandibleux X, Ehrgott M (2010b) A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. Discret Optim 7:149–165
Ralphs TK, Saltzman MJ, Wiecek MM (2004) An improved algorithm for biobjective integer programming and its application to network routing problems. Ann Oper Res 73:253–280
Rasmi SAB, Türkay M (2019) Gondef: an exact method to generate all non-dominated points of multi-objective mixed-integer linear programs. Optim Eng 20:89–117
Rasmi SAB, Fattahi A, Türkay M (2017) An exact algorithm to find non-dominated facets of tri-objective milps. In: The 12th international conference on multiple objective programming and goal programming (MOPGP), pp 30–31
Soylu B (2018) The search-and-remove algorithm for biobjective mixed-integer linear programming problems. Eur J Oper Res 268:281–299
Soylu B, Yıldız GB (2016) An exact algorithm for biobjective mixed integer linear programming problems. Comput Oper Res 72:204–213
Stidsen T, Andersen KA, Dammann B (2014) A branch and bound algorithm for a class of biobjective mixed integer programs. Manag Sci 60:1009–1032
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 158:46–55
Tamby S, Vanderpooten D (2020) Enumeration of the nondominated set of multiobjective discrete optimization problems. INFORMS J Comput 33:72–85
Wolsey LA, Nemhauser GL (2014) Section I.1.4. Modeling with binary variables III: nonlinear functions and disjunctive constraints. In: Integer and combinatorial optimization. Wiley, New York
Yu P-L, Zeleny M (1975) The set of all nondominated solutions in linear cases and a multicriteria simplex method. J Math Anal Appl 49:430–468
Zadeh L (1963) Optimality and non-scalar-valued performance criteria. IEEE Trans Autom Control 8:59–60