A Fast and Robust Algorithm for Solving Biobjective Mixed Integer Programs

Diego Pecin
Econometric Institute
Erasmus University Rotterdam, Rotterdam

Ian Herszterg, Tyler Perini, Natashia Boland, Martin Savelsbergh
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta

We present a fast and robust algorithm for solving biobjective mixed integer programs. The algorithm extends and merges ideas from two existing methods: the Boxed Line Method and the ϵ-Tabu Method. We demonstrate its efficacy in an extensive computational study. 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.

Key words: 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. 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. We focus on problems over both continuous and integer variables having linear objective functions and constraints, i.e., multiobjective mixed integer linear programs, and on exact algorithms, which are guaranteed to 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) and Gandibleux (2006), and the more recent paper of Stidsen et al. (2014)). There exist highly effective algorithms for generating the NDF of pure integer linear programs with two and three objectives, e.g., Boland et al. (2015a) for biobjective and Boland et al. (2017) for triobjective integer linear programs. The situation is quite different for multiobjective *mixed* integer linear programs (MOMIPs). Only a few computationally effective algorithms exist for biobjective mixed

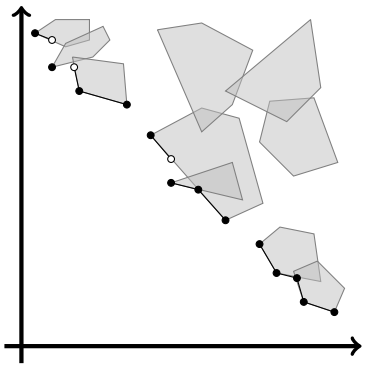


Figure 1 **The nondominated frontier of a BOMIP with minimization objectives. The shaded regions represent the images of feasible solutions with a common integer part. The nondominated frontier is darkened.**

integer programs (BOMIPs) – none for triobjective mixed integer linear programs – and these have been proposed only relatively recently, e.g., Vincent et al. (2013), Boland et al. (2015b), Soylu and Yıldız (2016), Perini et al. (2019), and Soylu (2018). The reason is, in part, 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 Figure 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.

The most recent and most practically effective algorithms for solving BOMIPs are *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 take advantage of the advances in single-objective solver software, since they repeatedly solve single-objective problems, both linear programs (LPs) and mixed integer linear programs (IPs).

In this paper, we extend and merge ideas from two of these algorithms, namely the ϵ-Tabu Method (Soylu and Yıldız 2016) and the Boxed Line Method (Perini et al. 2019). The result is a fast and, importantly, robust algorithm for generating the NDF of a BOMIP; it is robust in the sense that it works well across a wide range of instances with different characteristics. Furthermore, we demonstrate that the algorithm can produce a high-quality approximation of the NDF in a fraction of the time it takes to generate the complete NDF. This is the first exploration of approximating the NDF of a BOMIP. Here we propose metrics to assess the quality of such an approximation.

The remainder of the paper is organized as follows. In Section 2, we introduce notation, key definitions, and review the ϵ-Tabu Method and the Boxed Line Method. In Section

3, we propose enhancements to the ϵ-Tabu Method and the Boxed Line Method, which improve their performance and robustness. In Section 4, we present an algorithm that merges ideas from these two methods. In Section 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 Section 6, we present the results of an extensive computational study. In Section 7, we discuss the recently introduced Search-and-Remove Method and how it relates to our work. We conclude, in Section 8, with some final remarks.

2. Definitions and overview of existing 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 $\mathcal{X}_I := \{x_I \in \mathbb{Z}^{n_I} : (x_I, x_C) \in \mathcal{X},\ \exists x_C \in \mathbb{R}^{n_C}\}$ and $\mathcal{X}_C := \{x_C \in \mathbb{R}^{n_C} : (x_I, x_C) \in \mathcal{X},\ \exists x_I \in \mathbb{Z}^{n_I}\}$, 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 $\mathcal{Y} := \{y \in \mathbb{R}^2 : y = z(x),\ \exists x \in \mathcal{X}\}$, 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 nondominated points is the *nondominated frontier* (NDF), which we denote by $\mathcal{N}$.

A single NDP of (1) can be found by solving single-objective IPs[1] over $\mathcal{X}$ either by lexicographic optimization or by use of a scalarized objective function. 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

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

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

as the *scalarized IP with respect to* λ. If x^λ is an optimal solution to (3) with λ positive, so $\lambda_1, \lambda_2 > 0$, then $z(x^\lambda)$ is an NDP of (1). 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 $z_1^1 \leq z_1^2$. 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, then we call $L(z^1, z^2)$ a *nondominated line segment (NLS)*.

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 $z_1^k \leq y_1^{k+1}$ and $z_2^k \geq y_2^{k+1}$ for all $k = 1, \ldots, K-1$, and for which

$$y_1^1 < y_1^2 < \cdots < y_1^K \qquad \text{and} \qquad y_2^1 > y_2^2 > \cdots > y_2^K.$$

A gap may appear between second and first endpoints, respectively, of two consecutive NLSs in the NDF: for some k, it may be that $z_1^k < y_1^{k+1}$, which is a gap in the horizontal direction, and/or $z_2^k > y_2^{k+1}$, which implies a gap in the vertical direction. In the case that $z_1^k < y_1^{k+1}$, we define the interval $(z_1^k, y_1^{k+1}) \subset \mathbb{R}$ to be a *horizontal 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). The NDF of a slice problem consists of a (connected) set of (closed) line segments; we call this a *slice*[2]. The NDF of a slice problem

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

for x_I is called the *slice for* 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) generates the frontier from left to right (or vice-versa). 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 a given integer solution, and (2) check, sequentially – from left to right, if each line segment in the resulting slice is dominated or not using a (modified) lexicographic IP.

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 end of the line segment to the projected point is a NLS. ϵTM then processes the slice to which new NDP belongs. If, on the other hand, no such NDP is found, then the 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. For more details see Soylu and Yıldız (2016).

If an NDF contains many line segments from the same slice, as is the case in some benchmark instances, then it may be advantageous (more efficient) to check whether a *set* of consecutive line segments in a slice is dominated or not. This new enhancement is presented in more detail in Section 3.1.

2.2. Boxed Line Method

The Boxed Line Method (BLM) (Perini et al. 2019) is an extension of the Balanced Box Method (Boland et al. 2015a), which solves pure integer programs, to handle continuous variables. BLM has four main components: initialization, outer loop, inner loop, and line generation. 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 processes boxes in this queue until it is empty, at which time the algorithm terminates. The first step in processing a box is to search for the

leftmost NDP in the lower part 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. Two new boxes are added to the queue with these NDPs as corner points.

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 in 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 scalarized IPs; see Perini et al. (2019) for the details. Computational experiments have revealed that solve times for scalarized IPs are unpredictable and can sometimes be (too) long. To eliminate this unpredictability, a variant that does not solve scalarized IPs has been developed and will be presented in Section 3.2.

In Perini et al. (2019), two variants of BLM are presented. The *basic* variant, described above, has a simple inner loop. Additionally, a *recursive* variant is presented, which has a recursive inner loop: when a new NDP is found below the line segment, the inner loop recurses to discover the NLS containing the new NDP.

Regardless of the variant used, 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.

3. Enhancements

In this section, we propose enhancements to both ϵTM and BLM (the basic variant), which improve their performance and robustness.

3.1. An enhanced implementation of the ϵ-Tabu Method

We propose an enhanced implementation of ϵTM, which we denote by ϵTM-PWL, in which we solve 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 to determine the "left-most" segment of the slice that is nondominated. 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 for x_I^0 is generated and represented as a 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. 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 $L(z^i, z^{i+1})$.

For ϵTM-PWL, we propose a new formulation that considers all line segments in the slice simultaneously and models the piecewise linear slice function using binary variables (see, for example, Wolsey and Nemhauser (2014)). We introduce K binary variables y_i ($i = 1, \ldots, K$), one for each segment, with y_i associated with $L(z^{i-1}, z^i)$, and $K+1$ continuous variables λ_i ($i = 0, \ldots, K$), one for each of the end points of the line segments. We define a minimization problem that seeks the "left-most" point from a different slice that dominates any of the line segments, as follows:

$$\hat{z}_1 = \quad \min z_1(x) \tag{5a}$$

$$\begin{aligned} \text{s.t.} \quad & z_1(x) = \sum_{i=0}^{K} \lambda_i z_1^i \\ & \sum_{i=0}^{K} \lambda_i = 1 \end{aligned} \tag{5b}$$

$$
\begin{aligned}
&\lambda_0 \le y_1 \\
&\lambda_i \le y_i + y_{i+1} \quad \forall i = 1, \dots, K-1 \\
&\lambda_K \le y_K \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(5c)} \\
&\sum_{i=1}^{K} y_i = 1 \\
&z_2(x) \le \sum_{i=0}^{K} \lambda_i z_2^i \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(5d)} \\
&x_I \ne x_I^0 \\
&y_i \in \{0,1\} \quad i = 1, \dots, K \\
&0 \le \lambda_i \le 1 \quad i = 0, \dots, K \\
&x \in \mathcal{X}.
\end{aligned}
$$

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 (5c), together with the requirement that the y_i variables are binary, ensure that at most two λ_i variables may be positive (the rest must be zero), and that any two that are positive must be consecutive: exactly one i has $y_i = 1$, 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 (5b) and (5d) thus ensure that the image of x in criterion space, $z(x)$, 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 has a solution and we continue lexicographic minimization,

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

to ensure that the result, $\hat{z}$, is a nondominated point. By construction, there is a unique $i \in \{1, \dots, K\}$ such that $z_1^{i-1} < \hat{z}_1 \le z_1^i$. Then $L(z^{j-1}, z^j)$ is a NLS for all $j = 1, \dots, i-1$ 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. 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.

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, Rand7500.A, Bent7500.A and C21 (to be described in more detail in Section 6.1), and for all lexicographic and scalarized IPs solved during the execution of the basic variant of BLM, the area of the active box when the IP is solved and 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 seconds. 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 scalarized IPs solved when refining the initial overestimate of the line segment with lexicographic IPs. 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:

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

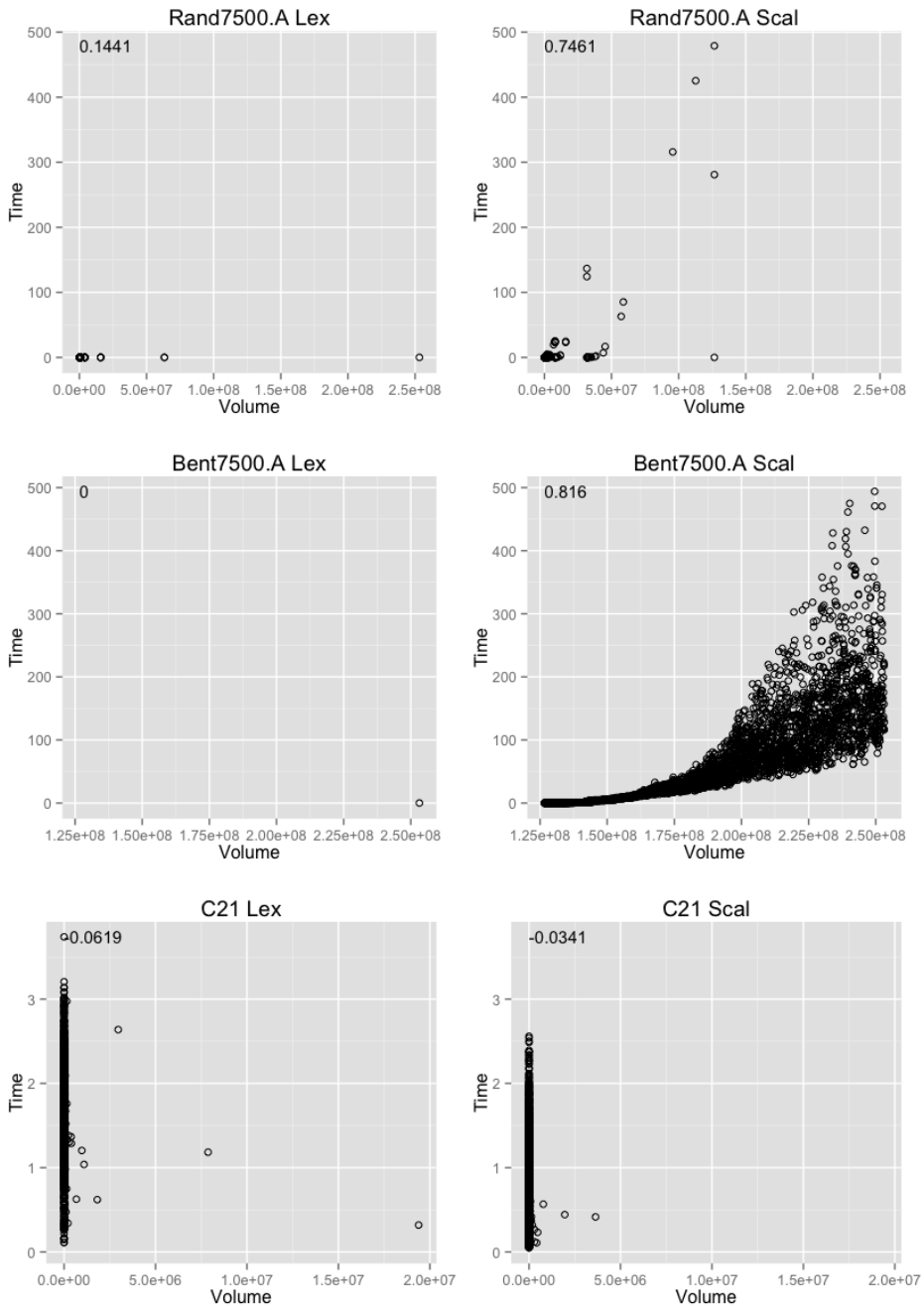


Figure 2 **For the basic variant of BLM, the area of the box and the solve time for the IPs, separated by instance and type of IP, are plotted as points. The resulting correlation coefficient is in the upper left of each graph. Note that only one lexicographic IP is solved on the bent instance, so the correlation is not well defined even though it is marked as zero. The correlation is at least 0.7 for scalarized IPs solved on the generated instances, i.e. Rand and Bent; everywhere else, the correlation is negligible.**

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 some $\epsilon > 0$) 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 constraintin (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.

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

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

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.

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 Figure 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 Section 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 to ϵTM to rapidly generate

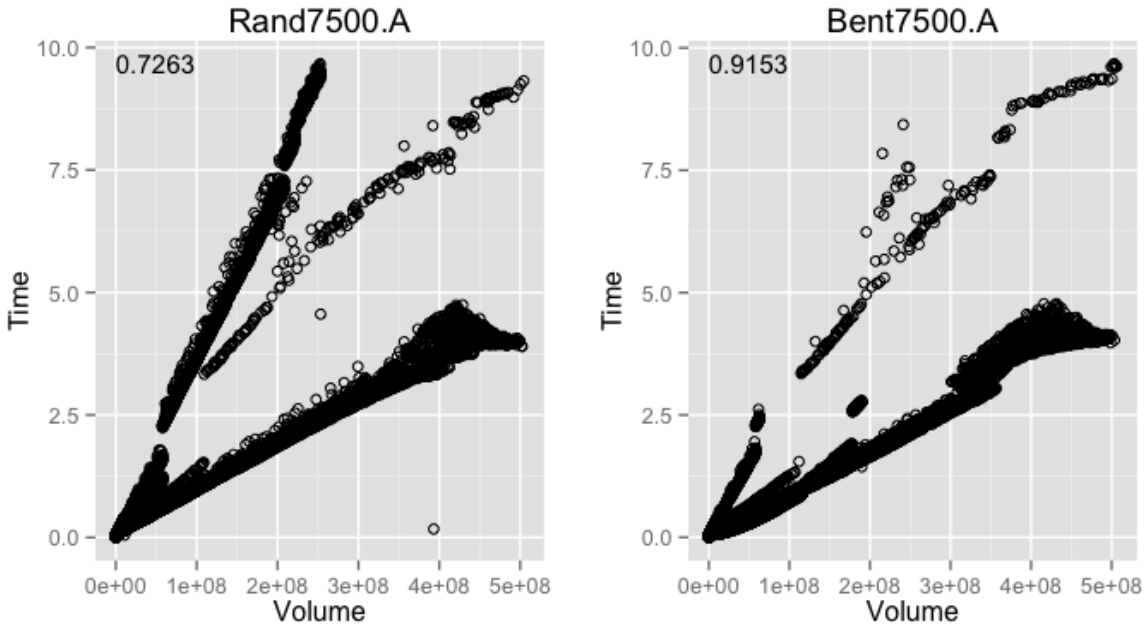


Figure 3 **The area of the box and the solve time for the lexicographic IPs solved by ϵTM, separated by instance, are plotted as points. The resulting correlation coefficient is in the upper left of each graph.**

the frontier in the remaining boxes (defining the as-yet unexplored area of the criterion space). We call SPureLex-PWL the variant of SPureLex using the ϵTM-PWL method. The choices of ρ are discussed in more detail in Section 6.

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 specifically designed to assess the quality of a subset of the NDF as an approximation to the full NDF of a biobjective mixed integer program:

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 naturally introduces diversification, in the sense that different parts of the criterion will be explored, and (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, we want to demonstrate the efficacy of ϵTM-PWL and SPureLex. Second, we want to investigate 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 2.3GHz with 50GB RAM, running Red Hat Enterprise Linux Server 7.4.

6.1. Test instances

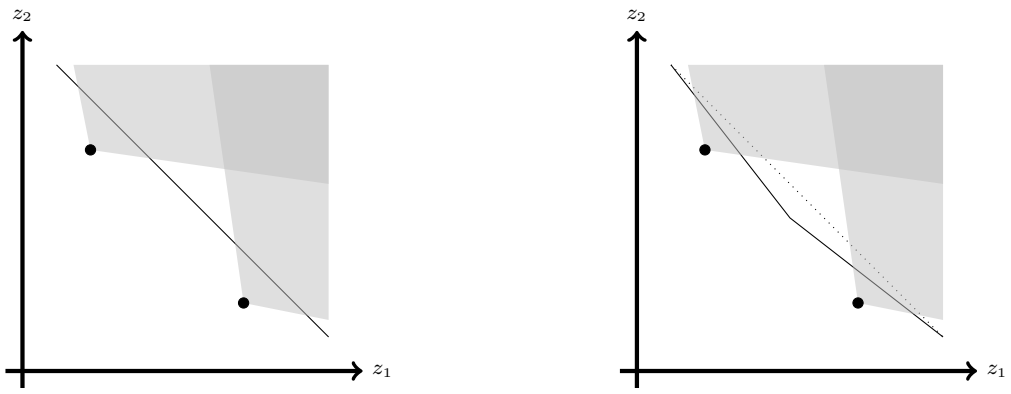
Three sets of instances were used: (1) the five largest instances (C320) from the benchmark instances proposed by Mavrotas and Diakoulaki (1998), which we refer to as "Historical", (2) five modified versions of these instances, which we refer to as "Relaxed Historical", (3) three large randomly generated 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), has half of the variables binary and half of them continuous, a randomly generated objective vector, and a set of knapsack constraints, in number the same as 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 include the largest instances in this set with $m = 320$. 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 to numerical rounding errors (see Fattahi and Turkay (2018) and Perini et al. (2019) for further discussion).

To increase diversity in the benchmark data set, we modify the Historical instances by relaxing their constraints, as follows. First, the constraints involving the continuous variables only 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 NDLs 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 NDLs 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 NDLs 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 Figure 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 the number of vertices or cones chosen to dominate the long line segment, which we simply call n. We include three



(a) The Rand instances include one slice in the form of a line segment (bold). In addition, for each of n additional integer solutions, its images of the 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).

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

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 Figure 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 1. 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 10000, 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 for ϵTM-PWL on the same instance, then in the row of Table 1 for the ϵTM method, we will report

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

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 value above zero indicates that the algorithm was not best for at least one instance in the set. A high value, such as 1.87, for example, indicates that faster algorithm gives a speed-up of a factor of one plus this value, on average; in the example, the speed-up is a factor of 2.87.

	Average				Max			
Algorithm	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†

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

† - on one instance of the set, the runtime limit of 86400 seconds was exceeded.

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. 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 to 3.

Unfortunately, these impressive gains are not observed for the Rand and Bent instances, as the NDF of these instances is not determined by a few integer solutions contributing many line segments, but by many integer solutions contributing few line segments. 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 hours.

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.

	Average				Max			
Algorithm	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

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

§ - on two instances of the set, the runtime limit of 86400 seconds was exceeded.

‡ - on three instances of the set, the runtime limit of 86400 seconds was exceeded.

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 hour time limit for each, whereas PureLex can solve all of them in less than 3 hours. For the Historic 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 Historic 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

	Average				Max			
Algorithm	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-0.005-PWL	0.00	0.00	0.02	0.02	0.00	0.00	0.03	0.03

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.
† - one instance of the group exceeded the time limit of 86400 seconds.
§ - two instances of the group exceeded the time limit of 86400 seconds.

is that PureLex performs well on the Historic 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.

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 Figure 3), even though ϵTM solves significantly fewer IPs than PureLex, ϵTM takes more time as it solves several IPs in boxes with large areas. This happens because ϵTM generates the nondominated frontier from left-to-right, and solves a lexicografic IP with respect to as-yet

unprocessed portion of the line segment L (in case of the Rand instances) or $\hat{L}$ (in case of the Bent instances).

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. 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; **fLNLS**, 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 "-". The results can be found in Tables 4 – 7 in Appendix 2, and are summarized in Figure 5.

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 minutes, PureLex and SPureLex have explored more than 99.5% of the area of $B(z^L, z^U)$, whereas ϵTM has explored a little more than 5% for the historical instance, and 1% or less for rand and bent instances.

Interestingly, looking at the fraction of the resolved area the initial box $B(z^L, z^R)$ (Area), the fraction of the number of nondominated line segments found (NLS), the fraction of the total length of the nondominated line segments found (TNLS), and the fraction of slices that contribute to the frontier found (Slice) for rand and bent instances reveals that ϵTM advances very slowly at the beginning (from an approximation perspective) and 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

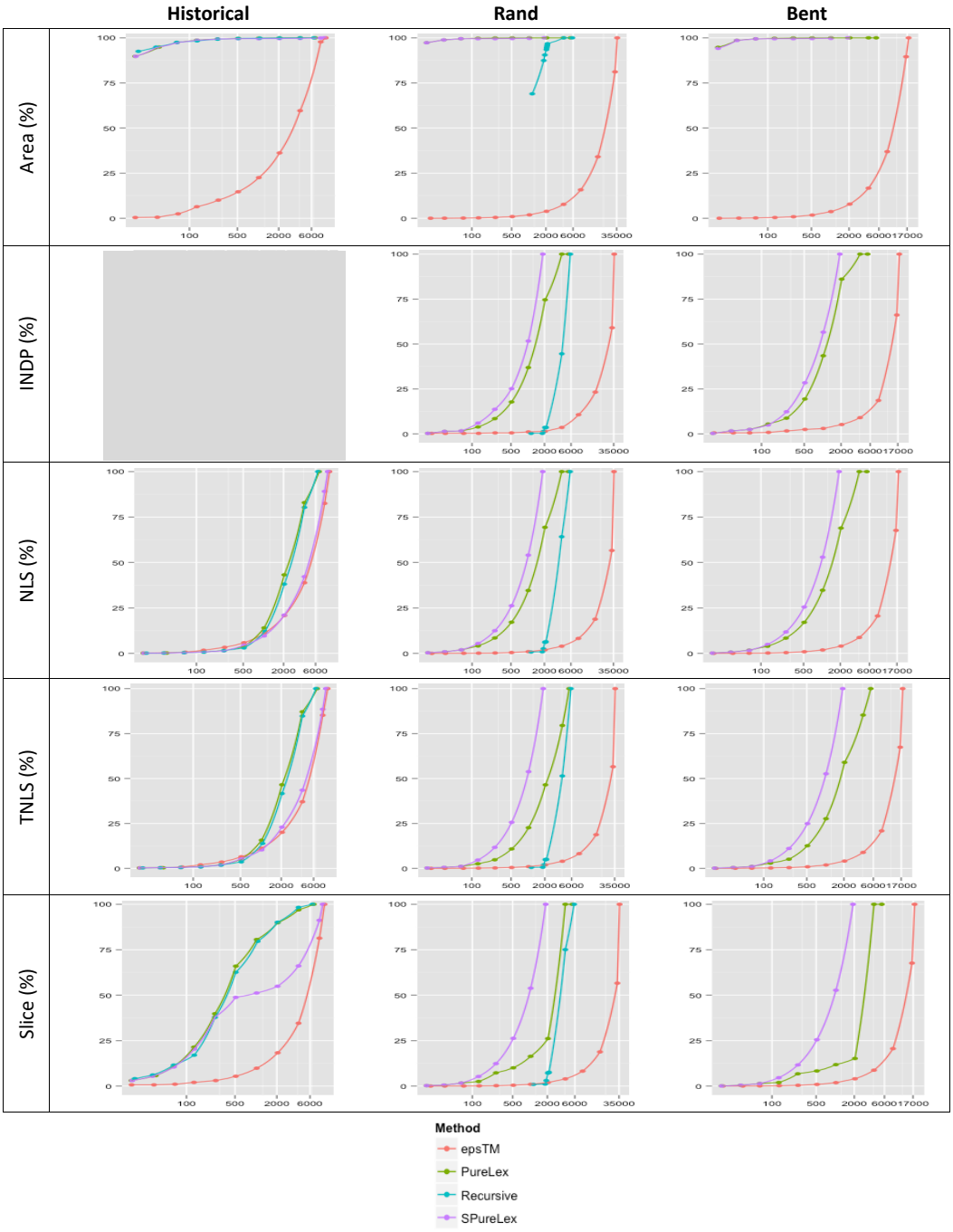


Figure 5 **Approximation results for ϵTM, the recursive variant of BLM, PureLex, and SPureLex with $\rho = 0.005$. There are no isolated NDPs in the NDF of the historical instance, so this space is blank. (No approximation metrics are reported for the recursive variant of BLM on the bent instance.)**

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 (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), but overall the switching makes for a more *robust* algorithm.

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. Search-and-Remove

Recently, Soylu (2018) introduced the Search-and-Remove Method (SRM). SRM generates slices intersecting (or close to) the nondominated frontier, and, in a post-processing step, SRM extracts the true nondominated frontier from these slices. More specifically, in each

iteration of SRM the following steps are performed: (1) use dichotomic search to identify all extreme supported NDPs, (2) for each integer solution identified, solve the associated slice problem and add the slice to the slice list[3], (3) add no-good constraints to ensure that the integer solutions cannot be found in subsequent iterations, and (4) evaluate the stopping criteria to determine whether the NDF can be extracted from the slices in the slice list. Note that after adding no-good constraints, the extreme supported NDPs found may not really be dominated, which will be determined in the post-processing step.

Soylu (2018) establishes six stopping criteria to recognize that the list of slices contains all the slices that can contribute to the NDF. An example of a simple stopping criterion is that all extreme supported NDPs found by the dichotomic search belong to the same slice. More sophisticated stopping criteria rely on bound sets, both lower and upper bounds sets. The search for slices can be terminated when a lower bound set has an empty intersection with an upper bound set. Different bound set constructions lead to different stopping criteria.

The post-processing step determines which segments of the slices in the slice list constitute the NDF. Slices may intersect and portions of slices may be dominated by portions of other slices. In order to extract the NDF, horizontal and vertical line segments emanating from the end points of each slice are created (in criterion space) as well as vertical lines passing through intersection points. These horizontal and vertical line segments divide the criterion space into regions with each region being either empty or with its upper line segments belonging to the NDF (assuming maximization objectives).

No public-domain implementation of SRM is available (yet), which is why we have not included it in our computational study. The computational results reported in Soylu (2018) indicate that it is substantially faster than ϵTM on the historical instances. However, SRM struggles with Rand and Bent instances (especially in the post-processing step)[4]

8. Final remarks

The algorithms presented and analyzed in this paper show that great progress has been made in solving biobjective mixed integer programs. Consequently, the next major challenge is the development of algorithms for triobjective mixed integer linear programs, of which, at the time of writing of this paper, there are none.

[3] The slice problem can be solved by either dichotomic search or by parametric simplex; Soylu (2018) uses parametric simplex.

[4] Private communication with Banu Soylu.

Acknowledgments

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

References

P. Belotti, B. Soylu, and M. M. Wiecek. A branch-and-bound algorithm for biobjective mixed-integer programs. Optimization Online, 2013.

N. Boland, H. Charkhgard, and M. Savelsbergh. A criterion space search algorithm for biobjective integer programming: The Balanced Box Method. INFORMS Journal on Computing, 27(4):735–754, 2015a.

N. Boland, H. Charkhgard, and M. Savelsbergh. A criterion space search algorithm for biobjective mixed integer programming: The Triangle Splitting Bethod. INFORMS Journal on Computing, 27(4):597–618, 2015b.

N. Boland, H. Charkhgard, and M. Savelsbergh. The Quadrant Shrinking Method: A simple and efficient algorithm for solving tri-objective integer programs. European Journal of Operations Research, 260: 873–885, 2017.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spectrum, 22(4):425–460, 2000.

A. Fattahi and M. Turkay. A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems. European Journal of Operational Research, 266(2):415–425, 2018.

X. Gandibleux. Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, volume 52. Springer Science & Business Media, 2006.

G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. European Journal of Operational Research, 107:530–541, 1998.

T. Perini, N. Boland, D. Pecin, and M. Savelsbergh. A criterion space method for biobjective mixed integer programming: The Boxed Line Method. INFORMS Journal on Computing, 32:16–39, 2019.

B. Soylu. The Search-and-Remove algorithm for biobjective mixed-integer linear programming problems. European Journal of Operational Research, 268(1):281–299, 2018.

B. Soylu and G. B. Yıldız. An exact algorithm for biobjective mixed integer linear programming problems. Computers & Operations Research, 72:204–213, 2016.

T. Stidsen, K. A. Andersen, and B. Dammann. A branch and bound algorithm for a class of biobjective mixed integer programs. Management Science, 60(April):1009–1032, 2014.

T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for biobjective case. Computers and Operations Research, 40:498–509, 2013.

L. A. Wolsey and G. L. Nemhauser. Section I.1.4. Modeling with binary variables III: Nonlinear functions and disjunctive constraints. In Integer and Combinatorial Optimization. John Wiley & Sons, 2014.

Appendix 1. 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 1 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

Given corner point (a, b), a cone in objective space is generally defined by $\{z \in \mathbb{R}^2 : \theta_1 z_1 + (1 - \theta_1) z_2 \geq \theta_1 a + (1 - \theta_1) b, \theta_2 z_1 + (1 - \theta_2) z_2 \geq \theta_2 a + (1 - \theta_2) b\}$, 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 $\theta_2 = 0$); we use $\pi = 0.05$.

Algorithm 2 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

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 $\{(a_i, b_i)\}_{i=1,2,...,n}$ and their cones be defined by $\{(\theta_1^i, \theta_2^i)\}_{i=1,2,...,n}$. We then have the following BOMILP for the bent instance:

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

$$\text{s.t.} \quad \theta_1^i x_1 + (1 - \theta_1^i) x_2 \geq \theta_1^i a_i + (1 - \theta_1^i) b_i - 2k(1 - y_i) \quad \forall i = 0, 1, 2, ..., n \tag{9}$$

$$\theta_2^i x_1 + (1 - \theta_2^i) x_2 \geq \theta_2^i a_i + (1 - \theta_2^i) b_i - 2k(1 - y_i) \quad \forall i = 0, 1, 2, ..., n \tag{10}$$

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

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

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

Appendix 2. Approximation results

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%
	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 4 Approximation results for ϵTM.

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%
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 5 Approximation results for PureLex.

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%
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 6 Approximation results for SPureLex with $\rho = 0.005$.

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%

Table 7 Approximation results for the recursive variant of BLM.