A two-stage approach for bi-objective integer linear programming

Rui Dai, Hadi Charkhgard*

Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, 33620 USA

Abstract

We present a new exact approach for solving bi-objective integer linear programs. The new approach efficiently employs two of the existing exact algorithms in the literature, including the balanced box and the $\epsilon$-constraint methods, in two stages. A computationally study shows that (1) the new approach solves less single-objective integer linear programs in comparison to the balanced box method, and (2) it significantly improves the solution time in comparison to both the balanced box and the $\epsilon$-constraint methods.

Keywords: two-stage approach, balanced box method, $\epsilon$-constraint method, bi-objective integer linear programming

1. Introduction

Many problems in different fields such as scheduling, transportation, and production planning can be formulated as an integer linear program. However, these problems often involve multiple objectives, and so due to conflict between them, finding a feasible solution that simultaneously optimizes all of the objectives is usually impossible. Consequently, in practice, decision makers want to understand the trade off between the objectives for these problems before choosing a suitable solution. Thus, generating many or all efficient solutions, i.e., solutions in which it is impossible to improve the value of one objective without a deterioration in the value of at least one other objective, is the primary goal in multi-objective integer linear programming.

This work focuses on developing an exact algorithm for Bi-Objective Integer Linear Programs (BOILPs). The main contribution of our research is efficiently combining two of the fastest (criterion

*Corresponding author

Email address: hcharkhgard@usf.edu (Hadi Charkhgard )
space search) algorithms, including the Balanced Box Method (BBM) developed by Boland et al. (2015) and the \( \epsilon \)-constraint method developed by Chankong and Haimes (1983), to take the main advantage of both of these algorithms, and hence solving BOILPs faster.

BBM is a recently developed algorithm which can be viewed as an extension of the box algorithm (Hamacher et al., 2007). Boland et al. (2015) have numerically shown that BBM can compute the nondominated frontier, i.e., the set of points in the criterion space corresponding to the efficient solutions, faster than many (if not all) of the existing methods such as the \( \epsilon \)-constraint method, the augmented weighted Tchebycheff method (Ralphs et al., 2006; Dächert et al., 2012), and the perpendicular search method (Chalmet et al., 1986). It is worth mentioning that if \( \mathcal{Y}_N \neq \emptyset \) denotes the set of nondominated points of a BOMILPs, then BBM solves \( 3|\mathcal{Y}_N| \) (feasible) Single-Objective Integer Linear Programs (SOILPs).

On the other hand, the \( \epsilon \)-constraint method is perhaps the most well-known algorithm for computing the (entire) nondominated frontier of BOILPs because of its simplicity and its long history. Boland et al. (2015) have shown that this algorithm does not outperform BBM in terms of solution time mainly because in BBM high-quality feasible solutions are always available to be initialized in SOILPs. However, the main advantage of \( \epsilon \)-constraint method is the fact that it solves only \( 2|\mathcal{Y}_N| + 1 \) (feasible) SOILPs.

In light of the above, the main goal of this paper is to develop a combined approach that (1) is better than BBM and \( \epsilon \)-constraint method in terms of solution time, and (2) can solve less SOILPs similar to the \( \epsilon \)-constraint method. To achieve these properties at the same time, the proposed approach starts by employing the BBM and at some point it switches to the \( \epsilon \)-constraint method. Of course the switching time is critical because if we switch too early the solution time would not probably be much different from the \( \epsilon \)-constraint method. Similarly, if it occurs too late, solving less SOILPs will not probably be achieved, and the solution time would not probably be much different from BBM. We develop a simple but effective mechanism for the switching that causes up to around 10% and 40% improvements in the solution time in comparison to the solution times of the original BBM and the \( \epsilon \)-constraint method, respectively.

The structure of the paper is organized as follows: In Section 2, the main concepts in bi-objective
integer linear programming are explained. In Section 3, both BBM and the \( \epsilon \)-constraint method are briefly explained. In Section 4, the proposed two-stage approach is introduced. In Section 5, a computational study is conducted. Finally, in Section 6, some concluding remarks are provided.

2. Preliminaries

In this section, we introduce some necessary notation and concepts related to BOILPs to facilitate presentation and discussion of other sections. Let \( c^1 \) and \( c^2 \) be \( n \)-vectors, \( A \) be an \( m \times n \) matrix, and \( b \) be an \( m \)-vector, a BOILP can be stated as follows:

\[
\min_{x \in \mathcal{X}} \{ z_1(x), z_2(x) \},
\]

where \( \mathcal{X} := \{ x \in \mathbb{Z}^n_+ : Ax \leq b \} \) represents the feasible set in the decision space, and \( z_1(x) := c^1 x \) and \( z_2(x) := c^2 x \) are two linear objective function. Note that \( \mathbb{Z}^n_+ := \{ s \in \mathbb{Z}^n : s \geq 0 \} \). The image \( Y \) of \( X \) under vector-valued function \( z = (z_1, z_2) \) represents the feasible set in the objective/criterion space, i.e., \( Y := z(X) := \{ y \in \mathbb{R}^2 : y = z(x) \text{ for some } x \in \mathcal{X} \} \). It is assumed that \( \mathcal{X} \) is bounded, and all coefficients/parameters are integers, i.e., \( A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m \), and \( c^i \in \mathbb{Z}^n \) for \( i = 1, 2 \).

**Definition 1.** A feasible solution \( x \in \mathcal{X} \) is called efficient or Pareto optimal, if there is no other \( x' \in \mathcal{X} \) such that \( z_k(x') \leq z_k(x) \) for \( k = 1, 2 \) and \( z(x') \neq z(x) \). If \( x \) is efficient, then \( z(x) \) is called a nondominated point. The set of all efficient solutions \( x' \in \mathcal{X} \) is denoted by \( \mathcal{X}_E \). The set of all nondominated points \( z(x) \) for some \( x \in \mathcal{X}_E \) is denoted by \( \mathcal{Y}_N \) and referred to as the nondominated frontier.

Overall, multi-objective optimization is concerned with finding all nondominated points. Since by assumption \( \mathcal{X} \) is bounded, the set of nondominated points of a BOILP, i.e., \( \mathcal{Y}_N \), is finite. Next we introduce an operation, the so-called lexicographic operation, that is frequently used in both BBM and the \( \epsilon \)-constraint method to compute the entire nondominated frontier. It is worth mentioning that in the literature there are other operations (for instance the augmented operation) as well. However, Boland et al. (2015) have shown the lexicographic operation performs the best for BOILPs.
So, the following operation:

\[
\bar{z} := \text{lex min}_{x \in \mathcal{X}} \{ z_1(x), z_2(x) \},
\]

is defined as first solving the following optimization,

\[
\bar{z}_2 := \min \{ z_2(x) : x \in \mathcal{X} \}
\]

followed by

\[
\bar{z}_2 := \min \{ z_2(x) : x \in \mathcal{X}, z_1(x) \leq \bar{z}_1 \}.
\]

Note that \( \text{lex min}_{x \in \mathcal{X}} \{ z_2(x), z_1(x) \} \) can be defined similarly. Note too that if the first optimization problem is infeasible then the second one must also be infeasible, and so it should not be solved. Also, if the first optimization problem is feasible, then its optimal solution must be feasible for the second one, and so this optimal solution can be used for warm-starting the second optimization problem.

3. Review of BBM and the \( \epsilon \)-constraint method

A high-level description of BBM and the \( \epsilon \)-constraint method are explained in this section.

![Figure 1: The workings of BBM at each iteration.](image-url)
3.1. The Balanced Box Method

This algorithm maintains a priority queue of rectangles in non-increasing order of their areas. At the beginning, there is no rectangle in the priority queue. So, the algorithm first finds the endpoints of the nondominated frontier, denoted by $z^T$ and $z^B$, by using $\text{lex min}_{x \in X} \{ z_2(x), z_1(x) \}$ and $\text{lex min}_{x \in X} \{ z_1(x), z_2(x) \}$, respectively. These two points result in defining a rectangle, denoted by $R(z^T, z^B)$, containing all not yet found nondominated points. Next, we explain the workings of the algorithm in an arbitrary iteration.

The algorithm pops out an element of the priority queue, denoted by $R(z_1, z_2)$ in which $z_1$ and $z_2$ with $z_1^1 < z_2^1$ and $z_1^2 > z_2^2$ are two already found nondominated points of the problem. The algorithm then splits the rectangle horizontally into two equal parts. It first explores the bottom rectangle for the absent local endpoint, which is denoted by $\bar{z}_1$ in Figure 1a, by using

$$\bar{z}_1 := \text{lex min}_{x \in X, z_2(x) \leq \frac{z_1^1 + z_2^2}{2}} \{ z_1(x), z_2(x) \}.$$ 

Based on the position of $\bar{z}_1$, it then splits the rectangle vertically, and explores the left rectangle for the absent local endpoint, which is denoted by $\bar{z}_2$ in Figure 1b, by using

$$\bar{z}_2 := \text{lex min}_{x \in X, z_1(x) < \bar{z}_1^1} \{ z_2(x), z_1(x) \}.$$ 

Note that since all decision variables and coefficients are integers, $z_1(x) < \bar{z}_1^1$ can be replaced by $z_1(x) \leq \bar{z}_1^1 - 1$. It can be shown that by finding these two local endpoints, $R(z^1, z^2)$ can be split into (at most) two independent rectangles, i.e., $R(z^1, \bar{z}_2)$ and $R(\bar{z}_1, z^2)$ containing all not yet found nondominated points in $R(z^1, z^2)$ as shown in Figure 1c. So, these two new rectangles should be added to the priority queue, before starting the next iteration. Next we make a few comments about BBM.

- To compute $\bar{z}_1$, a solution corresponding to $z^2$ is feasible for the first optimization problem in the lexicographic operation, and so it should be used for warm-starting. Similarly, to compute $\bar{z}_2$, a solution corresponding to $z^1$ is feasible for the first optimization problem in
(a) Step 1. (b) Step 2. (c) Step 3.

Figure 2: The workings of BBM when $R(z^1, z^2)$ is empty.

the lexicographic operation, and so it should be used for warm-starting.

- In the process of computing $z^1$, if after solving the first optimization problem in the lexicographic operation, it turns out that $z_1^1 = z_1^2$ then since $z^2$ is a nondominated point, we must have that $z_2^1 = z_2^2$. So, there is no need to solve the second optimization problem, and also $R(z^1, z)$ should not be added into the priority queue. Similarly, in the process of computing $z^2$, if after solving the first optimization problem in the lexicographic operation, it turns out that $z_2^2 = z_2^1$ then we must have that $z_1^2 = z_1^1$. So, there is no need to solve the second optimization problem, and obviously $R(z^1, z^2)$ should not be added into the priority queue.

An illustration of this observation can be found in Figure 2.

- Since all decision variables and coefficients are integers, we do not need to explore $R(z^1, z^2)$ if $z_1^1 = z_1^2 - 1$ or $z_2^1 = z_2^2 + 1$, because in this case there cannot exist any nondominated point in the interior of the rectangle.

**Proposition 2.** BBM solves at most $3 \| Y_N \|$ SOILPs (Boland et al., 2015).

3.2. The $\epsilon$-constraint method

Similar to BBM, the $\epsilon$-constraint method first finds the endpoints of the nondominated frontier, i.e., $z^T$ and $z^B$. These two nondominated points define a rectangle $R(z^B, z^T)$ containing all not yet found nondominated points. The algorithm then starts to generate the nondominated points one by one from the direction starting at $z^T$ and ending at $z^B$. An illustration of the first three
iterations of the algorithm can be found in Figure 3. Let $z^{last}$ be the nondominated point found in the last iteration, the new nondominated point can be produced by solving:

$$z^{New} = \text{lex min}_{x \in X, z_2(x) < z_2^{last}} \{z_1(x), z_2(x)\}.$$ 

Again note that since all decision variables and coefficients are integers, $z_2(x) < z_2^{last}$ can be replaced by $z_2(x) \leq z_2^{last} - 1$. Furthermore, the algorithm can immediately terminate if $z_1^{last} = z_1^B - 1$ or $z_2^{last} = z_2^B + 1$, because in this case there cannot exist any other nondominated point.

Finally, to compute $z^{New}$, a solution corresponding to $z^B$ is feasible for the first optimization problem in the lexicographic operation, and so it should be used for warm-starting. So, as soon as it turns out that $z_1^{new} = z_1^B$, the algorithm can be terminated and there is no need to solve the second optimization problem in the lexicographic operation in this case.

**Proposition 3.** The $\epsilon$-constraint method solves at most $2|Y_N| + 1$ SOILPs. (Boland et al., 2015)

4. A two-stage approach

We first make an observation about BBM to show the main motivation of our research. From workings of BBM in Figure 2, we observe that when a rectangle is empty, two SOILPs have to be solved to prove that it is empty. Now suppose that whenever a rectangle is empty, we immediately switch to the $\epsilon$-constraint method as shown in Figure 4. In this case, for each empty rectangle, only one SOILP has to be solved. So, we can conclude that if a given rectangle $R(z^1, z^2)$ is expected to
be empty, then by switching to the \( \epsilon \)-constraint method, hopefully one SOILP can be saved. Note that, in this case, for the \( \epsilon \)-constraint method, a solution corresponding to \( z^2 \) should be used for warm-starting (rather than a solution corresponding to \( z^B \)). Note too that since both versions of BBM and the \( \epsilon \)-constraint method use the idea of warm-starting and we basically provide an optimal solution for warm-starting when a rectangle is empty, we can expect that solving two SOILPs (in BBM) to be computationally more expensive than solving one SOILP (in the \( \epsilon \)-constraint method).

In light of the above, our proposed method solves a BOILP in two stages. In the first stage, it employs BBM in order to generate some nondominated points from different parts of the nondominated frontier, and so it quickly splits the search region into small rectangles (that are hopefully empty). In the second stage, the algorithm switches to the \( \epsilon \)-constraint method to conduct the searching in the not yet explored rectangles.

Obviously, the threshold for triggering the switch significantly influences the performance of the proposed method. If we trigger the switch too early, rectangles in the queue are probably not empty, and so the solution time may even increase. Similarly, if we trigger the switch too late, opportunities to save SOILPs while exploring empty rectangles will be eliminated, and so the solution time may not improve at all. The following definition and proposition are informative.

**Definition 4.** The ideal switch is defined as computing all nondominated points of a BOILP exactly once by BBM (and so not exploring any empty rectangle using BBM), and then switching to the \( \epsilon \)-constraint method for exploring empty rectangles.
Proposition 5. The two-stage method with ideal switch solves at most \(2.5\lceil|Y_N|\rceil\) SOILPs.

Proof. For computing a nondominated point for the first, two SOILPs have to be solved. Since in BBM, the rectangles are disjoint, after computing all nondominated points there are at most \(0.5\lceil|Y_N|\rceil\) empty rectangles in the priority queue. For each empty rectangle, one SOILP has to be solved by the \(\epsilon\)-constraint method to prove that it is empty. Therefore, the result follows. \(\Box\)

We note that in practice it is highly unlikely to realize the ideal switch and to push the proposed two-stage method to its theoretical maximum efficiency. This is because real-world BOILPs rarely have well-distributed nondominated frontiers, i.e., the distance between two adjacent nondominated points may vary significantly. In consequence, avoiding exploring empty rectangles in the first stage is likely to be impossible. However, since BBM explores rectangles in non-decreasing order of their areas, it is expected that empty rectangles to arise more often at the end of the search. This is because intuitively smaller rectangles are more likely to be empty. So, by considering this observation, we next introduce a simple switching technique that can potentially meet two goals including (1) reducing the solution time, and (2) saving SOILPs.

Let \(\eta \in [0, 1]\) and \(\theta \geq 1\) be two user-defined parameters. Furthermore, let \(\beta_1\) denote the number of the first optimization problem in the lexicographic operation in the last \(\theta\) SOILPs that are solved by BBM. Similarly, let \(\beta_2\) denote the number of the second optimization problem in the lexicographic operation in the last \(\theta\) SOILPs that are solved by BBM. So, \(\beta_1 - \beta_2\) is the number of times that BBM was not able to find a new nondominated point in its last \(\theta\) SOILPs it has solved. We impose our proposed method to switch as soon as \(\beta_1 - \beta_2 \geq \lceil\eta\theta\rceil\) is satisfied at the same time. Intuitively, this condition measures the likelihood of facing empty rectangles in the subsequent iterations of BBM.

5. A Computational Study

In this section we conduct a computational study to determine a reasonable value for \(\theta\) and \(\eta\). We implement \(\epsilon\)-constraint method, BBM, and the proposed two-stage approach in C++ and employ CPLEX 12.7 to solve SOILPs. All computational experiments are carried out on a Dell
PowerEdge R630 with two Intel Xeon E5-2650 2.2 GHz 12-Core Processors (30MB), 128GB RAM, and the RedHat Enterprise Linux 6.8 operating system, and using a single thread.

Two large sets of instances are downloaded from http://hdl.handle.net/1959.13/1036183. These instances used by Boland et al. (2015). The first set contains 20 instances of (bi-objective) 2-Dimensional Knapsack Problem (2DKP) with 375, 500, 625, and 750 binary variables. The second set contains 20 instances of (bi-objective) Assignment Problem (AP) with $200 \times 200$, and $300 \times 300$ binary variables.

We assume that $\theta \in \{7, 15, 31\}$ and $\eta \in \{0.1, 0.2, 0.4, 0.6\}$ and consider all possible combinations to find the best value for $\theta$ and $\eta$. It is worth mentioning that, regardless of value of $\eta$, we have computationally observed that (in practice) the proposed trigger will be activated naturally when at least $\theta$ rectangles are explored by BBM. Note that, we know that in BBM (1) after exploring a rectangle, the area of the two new rectangles are at most half of the original rectangle (Boland et al., 2015), and (2) the rectangles will be explored in non-decreasing order of their areas. So, exploring $\theta$ rectangles by BBM, implies that the sum of areas of not yet explored rectangles is at most $a\left(\frac{a(R(z^T, z^B))}{\theta+1}\right)$, where $a(R(z^T, z^B))$ is the area of $R(z^T, z^B)$. This can be an indication that the proposed two-stage method will perform well even if the goal is to compute high-quality approximations for the nondominated frontier when a limited computational time is available.

Our proposed values for $\theta$ and $\eta$ are 15 and 0.2, respectively. This can be easily observed from Figure 5a, in which the ratio of the solution time of each algorithm to the solution time of BBM are reported for each instance. The first 20 instances are 2DKP instance, and the others are AP instances. In this figure, the “best performance” and “worst performance” shows the performance of the best and worst combinations of $\theta \in \{7, 15, 31\}$ and $\eta \in \{0.1, 0.2, 0.4, 0.6\}$ for each particular instance that resulted the minimum and maximum solution time, respectively. So, the performance of any combination lies within the best and worse performances. Observe that even the worse performance is better than both BBM and $\epsilon$-constraint method in most instances. Observe too that the proposed combination performance is as good as the best performance, and the solution time of the proposed combination is always better than both BBM and the $\epsilon$-constraint method. In fact the solution time of BBM is up to around 10% less than BBM and up to around
40% better than $\epsilon$-constraint method. It is worth mentioning that, from Figure 5b, we see that the average number of SOILPs per each nondominated point when using the proposed combination is significantly smaller than BBM and it is close to the $\epsilon$-constraint method.

Figure 5: performance of the proposed threshold.

(a) Solution time.  
(b) Number of SOILPs.

Figure 6: Performance of the two-stage approach on just two instances.

(a) An instance of 2DKP.  
(b) An instance of AP.

To highlight the performance of our proposed combination better, we also conduct a different experiment on one of the instances of 2DKP and one of the instances of AP. In this experiment, we compute the (total) solution time of the proposed method if we switch to the $\epsilon$-constraint method
after exploring $k$ rectangles (by using BBM). We have reported the results for different values of $k$ in Figure 6. The solution time of the best and worst performance and the proposed combination are also reported for each instance. Observe that the solution time of the proposed combination is almost equal to the global minimum that can be ideally reached by the proposed two-stage method.

6. Conclusion

We presented a simple but effective two-stage approach for solving BOILPs. This method combines BBM and the $\epsilon$-constraint method to remedy their weakness. The proposed method is faster, and solves less SOILPs. We hope that its simplicity, versatility, and performance encourages (more) researchers to consider studying combined methods for BOILPs.

References


