Developed Some Algorithms

Dr .Subhashish Biswas ,Rajoo

Abstract:
A reformulation, that was proposed for a knapsack problem by Munapo and Kumar (2016), has been extended to single and bi-objective linear integer programs. A further reformulation for a knapsack problem is also proposed and extended to the bi-objective case. These reformulations significantly reduce the number of branch and bound iterations with respect to single and bi-objective knapsack models and general linear integer programming models. Numerical illustrations have been presented and computational experiments have been carried out to compare the behaviour before and after the reformulation. For this purpose, Python software was used.

Keyword: Single, Bi-Objective Integer Linear Programming, MOILP, Knapsack Method, Python software.

Introductive review

Many applications in real-life problems require integer restricted variables in a linear program. Many approaches have been developed to reduce computational effort and time required to get to the optimal solution. Some of the well-known approaches for integer programming problems are by Mitten (1970); Kelley (1960) and Taha Hamdy (2003). One of the well-known integer linear programming problem in discrete optimization is a knapsack problem, which has also been attempted by several researchers such as Kellerer et al. (2004); Taha Hamdy (2003) and Della Croce et al. (2017), and real-life applications have been discussed by Winston and Goldberg (2004). A few well-known methods that have been used to solve the knapsack problem are based on dynamic algorithm by Chebil and Khemakhem (2015), greedy algorithms by Chu and Beasley (1998) and the branch and bound concept by Dudzi´nski and Walukiewicz (1987).

Padberg and Rinaldi (1991) has improved the standard branch and bound algorithm by using the concept branch and cut algorithm. It is a combination of the ideas from branch and bound and Gomory cutting

plane Kelley (1960). Branch and price by Barnhart et al. (1998) is another improvement in the branch and bound approach for solving the knapsack problem.

Instead of modifying these existing methods to solve knapsack problem, Munapo and Kumar (2016) suggested a reformulation of the knapsack model itself. This reformulation resulted in a significant reduction in the number of sub-problems used by the standard branch and bound approach to reach the optimal solution.

In this chapter, the knapsack reformulation by Munapo and Kumar (2016) has been applied to single and bi-objective general linear integer programs (LIP). Furthermore, the knapsack model is slightly modified resulting in further reduction in branch and bound iterations.
The proposed reformulation results in:

- Additional constraints and variables yet significantly reduce the number of branch and bound iterations.

- A new modification with respect to the knapsack model developed in this chapter adds only one extra constraint that is identified by a few simple steps discussed in the chapter.
- The computational experiments indicate significant savings in the number of sub problems required by branch and bound algorithm to verify the optimal solution.
- The efficiency of this new ideas has been investigated by implementing it to solve a few bi-objective models. The _-constraint method developed by Chankong and Haines (2008) has been used to solve the bi-objective models before and after reformulation. Numerical examples have been used to demonstrate the enhancement.

This chapter attempts to:

1. Extend the reformulation by Monaco and Kumar (2016) for the knapsack model to the general linear integer programming model.
2. Proposes a new modification for the knapsack model.
3. Extends the above two ideas for the bi-objective models

Prelaminar and concepts:

A mathematical model for multi-objective linear integer programming problem can be expressed as follow:

$$Z = \text{Min}\{ z_1(x), z_2(x)\} \quad \text{……..(1)}$$

subject to x ∈ X where $z_i(x)$ represents the i^{th} objective functions for i = 1…. k.

Here x is a vector of integer decision variables in X which represent the feasible set in the decision space. z(x) is an objective vector in Y which represent the feasible set in the criterion space. The integrability condition is represented by $x_j \in Z$ for all j = 1, 2, …. n, where n is the number of variables.

Each objective function above can be stated as $z_i(x) = \sum_{j=1}^{n} a_{ij}\, x_j$, where $a_{ij} \in Z$, i = 1….. k and j = 1 2 …. n. The optimization problem (1) is a multi-objective linear integer programming problem as all the objective functions and the constraints involved are linear. If there are one or more non-linear objective functions or non-linear constraints, then the problem become a multi-objective non-linear integer programming problem. Substantially, this study discusses only the multi-objective linear integer programming problem Chong and
Zak (2013); Ehrgott (2005); Antunes et al. (2016).

Approximately well-known methods for BOILP

A widely used approach for solving a MOIP is a scalarization technique where the multi-objective problem is transferred into a Bi-Objective problem and then solved using any method for single objective integer or mixed integer linear model.

1. The weighted-sum method

One of the common concepts of scalarization method is the weighted-sum method. This method transfers the multi-objectives into a combined

objective by multiplying each objective function with a suitable weight and then aggregate all of them into a single objective function. The problem becomes:

$$\text{Min } \left\{\sum_{i=1}^{k} w_i z_i\right\}, x_i \geq 0 \ \sum_{i=1}^{k} w_i = 1, w_i \geq 0 \quad \ldots\ldots \ . \quad (2)$$

One can choose the weight of an objective based on relative importance of that objective. It is a simple method and it can generate all supported non dominated points However, it cannot generate the non-supported non-dominated points which is considered as a disadvantage of this approach. In other word it does not give the complete set of nondominated points for MOIP Aneja and Nair (1979). Therefore, weighted -sum method, in this thesis, has been used as a part of some other methods to find the whole set of non-dominated points for MOIP.

2. The ϵ -constraint method

Haimes Vira and Haimes (1983), introduced the ϵ -constraint method which is one of the most popular methods for solving MOIP. It transfers the multi-objective into a single-objective. More details about this method are presented in Chapter 5.

3. The two phases method

The two phases method is a general approach to solve MOIP in discrete optimization Ulungu and Tighes (1995). It solves the multi-objective problem via two phases. In the first phase, supported non-dominated points are computed by using the weighted sum method. In the second phase, non-supported non-dominated points are found by searching other region in the criterion space by considering the result from the first phase. Authors modified the two-phase method for BOIP in Przybylski et al. (2008), then they have extended it to solve TOIP in Przybylski et al. (2010).

Some software tools used for solving integer problems

There are many optimizations software available for use by the operations research community. For example, Python, LINDO, LINGO, AMPL,

CPLEX and GAMS. Some of them are free and others can be purchased. Every software has advantages and disadvantages. However, we used software that were used by earlier authors as we wanted to study their work, compare their work with ours and establish if we have attained some improvement over the existing approaches. We have reported performance of these models in this thesis. This thesis is dealing with integer programming models in single and multi-objective programming. All multi-objective optimization studies have used CPLEX software and we also used the same to establish our claims. We also studied the Knapsack model were the previous authors used the Python, hence, once again, we used Python. In this thesis, Python and C++ are the two software packages that have been used to prove various claims made in this thesis. A short note
on characteristics of these two packages is discussed below:

1. Python software:

It a mathematical software to solve operations research problems. In context of mathematical programming, it is useful for dealing with single-objective programs Taha Hamdy (2003). In this thesis it has been used for solving the integer programming problems. One can easily enter the input data and get the output either in the form of optimal solution or some related information. There are many problems that can be solved using Python software such as:

- linear programming
- Integer and mixed-integer programming
- Special structured problems like assignment and transportation
- Network routing problems and CPM etc.

The above list is not intended to be exhaustive. However, many limitations surround this software, for example, number of variables and constraints are limited. It is a teaching software not suited to deal with models arising from real-life situations. It is not very useful for solving multi-objective mathematical programming models.

Bi-objective models with respect to General Integer Programming

A mathematical model of a bi-objective linear integer programming problem is given by:

$$\text{Max } \{z_1 x,\ z_2 x\}$$

$$\text{s.t. } x \in X$$

where $z_1(x)$ and $z_2(x)$ represent the two objective functions. Here X represents the feasible set in the decision space and Y represents the feasible set in the criterion space such that $x_j \in Z$ for all j = 1,2,…. n, where n is the number of variables.

Definition .1

A feasible solution $x \in X$ in the decision space is called efficient solution if there is no $x \in X$ such that $z_i(x) \geq z_i(\dot{x})$ for each i and $z_i(x) > z_i(\dot{x})$ for at least one *i*. The image $z(\dot{x})$ of efficient point $\dot{x}$ in the criterion space is called non-dominated point. Let X_E, Y_N denote the set of all efficient solutions and non-dominated solutions, respectively.

Definition .2

An efficient solution $x^* \in X$ is called non-supported efficient solution if it is dominated by infeasible convex combination of other efficient solutions in X_E. Non-supported nondominated point $z(x^*)$ is known as the image of non-supported efficient point x^*. However, other solutions are supported non-dominated.
In this chapter we will find the whole set of non-dominated points which include supported and
non-supported points.

Formulation of Integer Programming & Knapsack Problem:

A mathematical model of a general linear integer problem is given:

$$Z = \text{Max } \left\{\sum_{i=1}^{n} c_j x_j\right\}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad (i = 1,2,\ldots,m) \qquad \ldots\ldots \ldots (3)$$

$$a_{ij} x_j, b_i, c_j \geq 0, x_j \geq 0 \text{ and integer}$$

here $a_{ij}x_j, b_i, c_j$ are positive constraint and x_j is an integer restricted variable, j = 1,2,…,n.

The knapsack problem is a particular case of (3) with a single constraint as given in (4):

$$Z = \min \left\{\sum_{i=1}^{n} c_j x_j\right\}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \qquad \ldots\ldots\ldots(4)$$

Again a_j , b, c_j are positive constants and xj is an integer restricted variable, j = 1,2…… n.

In each LP model, since variables can be easily renamed, and constraints can be rearranged, for convenience of presentation and without any loss of generality, it is assumed that the 1st constraint in (1) satisfies the condition:

$$a_{11} < a_{12} < \ldots. < a_{1n} \qquad \ldots\ldots\ldots \quad (5)$$

A more general case of (5) can be as given by (6):

$$a_{11} \leq a_{12} \leq \ldots. \leq a_{1n} \qquad \ldots\ldots\ldots \quad (6)$$

Which is also considered in this chapter Similarly, for convenience of presentation, it has been assumed that the knapsack constraint in (4) also satisfies the condition.

$$a_1 < a_2 < \ldots. < a_n \qquad \ldots\ldots\ldots \quad (7)$$

Proposition.1 Let z^1 and z^2 be two points in the criterion space. If $(z^1 - z^2_{\geq}) \cap Y_N = \emptyset$ and $(z^1 - z^2_{\geq}) \cap Y_N = \emptyset$, then min x ∈ X $\{z_1(x), z_2(x) : z(x) \in R(z^1, z^2)\}$ returns a nondominated point if it has a solution.

Proof. It is trivial that the proposition is correct when the rectangle $R(z^1, z^2)$ contains no feasible solution. Now suppose that there exists a feasible solution in the rectangle but $\bar{z}$ = min x ∈ X $\{z_1(x), z_2(x) : z(x) \in R(z^1, z^2)\}$ ∈ R $(z^1, z^2) \notin Y_N$. Therefore, there should be a point $\hat{z} \in Y_N$ which dominates $\bar{z}$. According to the assumptions, i.e., $(z^1 - z^2_{\geq}) \cap Y_N = \emptyset$ and (z2−R2 >) ∩YN = /0 and $(z^1 - z^2_{\geq}) \cap Y_N = \emptyset$, $\hat{z}$ cannot be outside the

rectangle. So, $\hat{z}$ must be in R (z^1, z^2). However, referring to the definition of min x ∈ X {$z_1(x)$, $z_2(x)$: z(x) ∈ R(z^1, z^2)}, z_1 must be the optimal value of $z_1(x)$ in R(z^1, z^2). This implies that $\hat{z}_1 = \bar{z}_1$ and $\hat{z}_2 < \bar{z}_2$, but $\hat{z}_2 < \bar{z}_2$ is a contradiction, because, by definition, $\bar{z}_2$ is optimal value of $z_2(x)$ in R (z^1, z^2) when $z_1(x) \leq \bar{z}_1$.

Corollary .2 Let z^1 and z^2 be two points in the criterion space. If ($z^1 - z^2_{\leq}$) ∩ Y_N = ∅, ($z^1 - z^2_{\leq}$)∩Y_N = ∅, and z^2 ∈ Y_N then $\bar{z}$ = min x ∈ X {$z_1(x)$, $z_2(x)$: z(x) ∈ R(z^1, z^2)} ∈ Y_N furthermore, if $\bar{z} = z^2$, z is the only nondominated point in R (z^1, z^2).

Because it may be too time-consuming to compute the exact efficient frontier, we will also consider computation of an approximation of the efficient frontier or an approximate efficient frontier. To compare different approximate efficient frontiers, we use the hypervolume indicator.

Let $\bar{y}_N$ be an approximate efficient frontier, and H($\bar{y}_N$) denotes its hypervolume. If $\bar{y}_N \subseteq \bar{y}_N$, i.e., if the approximate efficient frontier consists of a subset of the points of the true efficient frontier, the definition of hypervolume can be adjusted to obtain a measure $\bar{H}$ ($\bar{y}_N$) with the property $\bar{H}$ ($\bar{y}_N$) ≥ $\bar{H}$ (y_N), i.e., the adjusted hypervolume of an approximate efficient frontier is larger than the adjusted hypervolume of the true efficient frontier. More specifically, $\bar{H}$ ($\bar{y}_N$) = H(y_N) ∪ $\bigcup_{i=1}^{k-1} a$ (z^i, z^{i+1}). Furthermore, if the only nondominated points in R(z^i, z^{i+1}) are z^i and z^{i+1}, then a (R(z^i, z^{i+1})) can be eliminated from the calculation of the adjusted hypervolume to provide a tighter upper bound on the hypervolume of the true efficient frontier. This implies that $\bar{H}$ (y_N) = H(y_N), and $\bar{H}$($\bar{y}_N$) provides an upper bound on H(YN). **Figure 3.1** shows the hypervolume and the adjusted hypervolume for an approximate efficient frontier {z^1, z^2, . . . , z^6} with $z^1 = z^T$, $z^6 = z^B$, and for which it has been proved that the only nondominated points in R(z^4, z^5) are z^4 and z^5. The difference $\bar{H}(\bar{y}_N)$ - $H(\bar{y}_N)$ provides a quantitative assessment of an approximate efficient frontier. As far as we have been able to establish, we are the first to introduce an adjusted hypervolume to assess the quality of an approximate efficient frontier independently, i.e., independent of other approximate frontiers.

Bi-Objective Integer Existing of Criterion Space Search Algorithms

Popular scalarization techniques for solving BOIPs are the perpendicular search method, the augmented weighted Tchebycheff method, and the ε-constraint method. All methods have

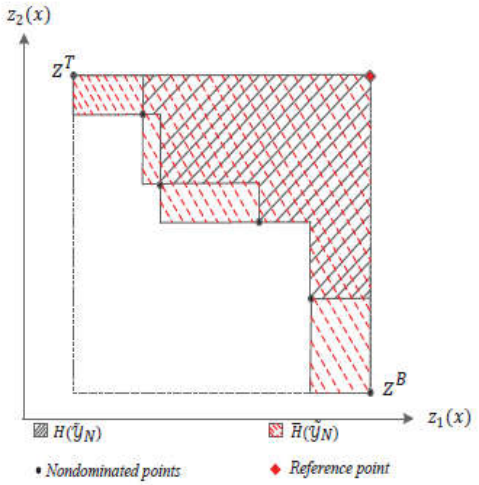


Fig. Lower bound and upper bound for the optimal hypervolume

essentially the same common structure. They maintain a list of nondominated points and a priority queue with rectangles that still have to be explored with the root element being the rectangle with largest area. Initially, the priority queue only contains rectangle $R(z^T, z^B)$. Exploring a rectangle $R(z^1, z^2)$ is done by means of an optimisation problem, which identifies a yet unknown nondominated point z if it exists. If so, the nondominated point z is used to create two new rectangles $R(z^1, z)$ and $R(z, z^2)$. These newly created rectangles are added to the priority queue unless they cannot contain any new nondominated points. The process terminates when all rectangles have been explored, i.e., when the priority queue is empty. See Algorithm 1 for details. The methods differ in the optimization problem used to search for an as yet unknown nondominated point in a rectangle.

We will discuss each of these methods in the remainder of this section, but we start by reviewing the weighted sum method for finding extreme supported nondominated points, because it is sometimes used as a component of other criterion space search algorithms.

Algorithm 1: Generating Nondominated Points

List. create(L); List. Add (L, z^T); List. Add (L, z^B)

PQ. Create (P); PQ. Add (P, R (z^T , z^B))

while not PQ. empty (P) **do**

PQ. Pop (P, R (z^1, z^2))

Search Rectangle (R(z^1, z^2))

if z ≠ null then

List. Add (L, z)

if R (z^1,z) can still contain no dominated points then PQ. add (P,R(z^1, z))

if R (z, z^2) can still contain non-dominated points then PQ.add(P,R(z, z2))

return L

Bi-Objective Integer using Weighted Sum Method

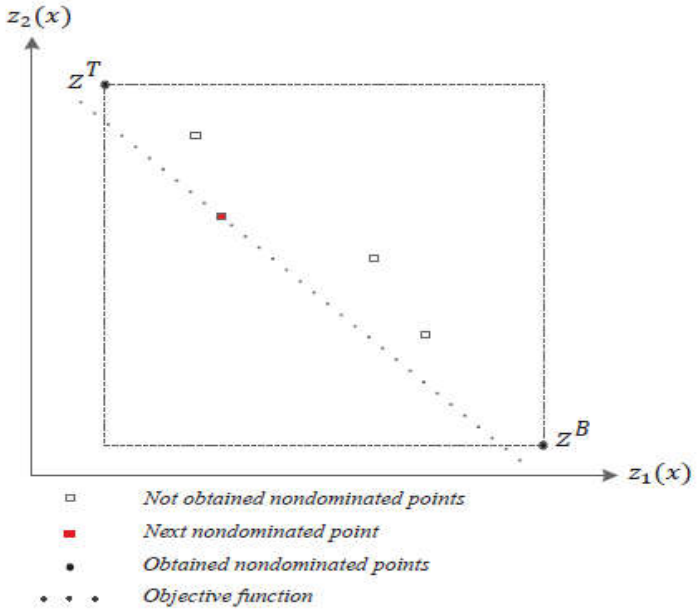
The weighted sum method finds all extreme supported nondominated points. It uses the following optimisation problem to search a rectangle defined by points z^1 and z^2

$$\min_{x \epsilon X} \lambda_1 z_1(x) + \lambda_2 z_2(x)$$

subject to z(x) ∈ R (z^1, z^2)

with $\lambda_1 = z_2^1 - z_2^2$ and $\lambda_2 = z_1^2 - z_1^1$, i.e.the objective function is parallel to the line that connects z^1 and z^2 in the criterion space. **Figure 3.2** shows an example with $z^1 = z^T$ and $z^2 = z^B$.

It is easy to see that the optimum point z^{new} can be a yet unknown extreme supported nondominated point if $\lambda_1 z_1{}^{new} + \lambda_2 z_2{}^{new} < \lambda_1 z_1^1 + \lambda_2 z_2^1$. That is, the optimisation either returns a new (and possibly extreme) supported nondominated point, one of z^1 and z^2, or a convex combination of z^1 and z^2. Note that when a yet unknown supported nondominated point is returned, it is not necessarily an extreme supported nondominated point. However, the set of supported nondominated points returned by the weighted sum method is guaranteed to include all extreme supported nondominated points, but may also include supported nondominated points that are not extreme.



The weighted sum method

The ε- Constraint Method

The ε-constraint method [24] is probably the most popular scalarization method (for generating all nondominated points). It always optimises one of the objective functions while using the value of the other objective function to restrict the search. More specifically, starting from z^T, the

method iteratively finds the nondominated point that is closest to the last found nondominated point zℓ by solving.

$$\text{Min } x \in X \{z_1(x), z_2(x) : z(x) \in R(z^{\ell} - \begin{pmatrix} 0 \\ \varepsilon \end{pmatrix} z^B)\}$$

With ε > 0 a small constant, until z^B is reached. Fig….. illustrates how the closest non-dominated point to z^T is found.

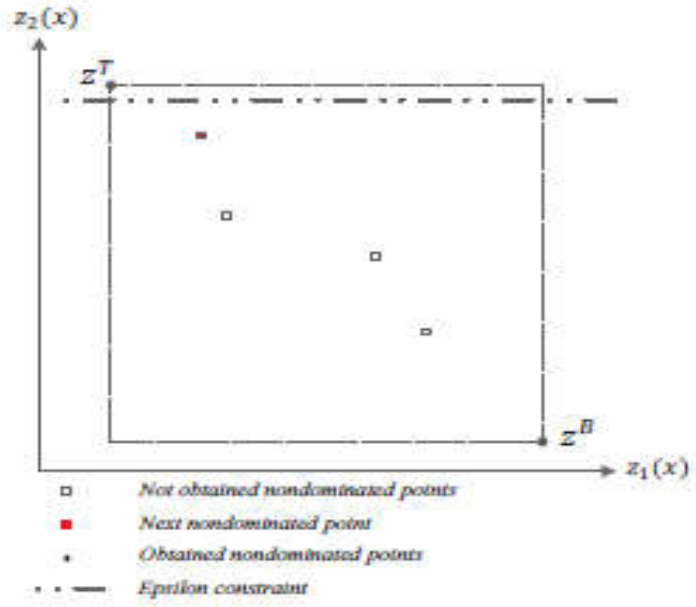


Fig. The ϵ- Constraint method

Using Balance Box method

The existing criterion space search methods each have their own "weakness". The ε-constraint method is not suited for obtaining high-quality approximate efficient frontiers quickly. The perpendicular search method may not be efficient because half of the optimisation problems solved are infeasible and integer programming solvers tend to struggle when proving infeasibility. The augmented Tchebycheff method may not be efficient because integer programming solvers tend to toil when faced with min-max objectives. The balanced box method seeks to remedy all these weaknesses.

We present the method by discussing how the initial rectangle $R(z^T, z^B)$ is explored in great detail; subsequent, smaller rectangles are handled similarly. Therefore, consider rectangle $R(z^T, z^B)$. To find a yet unknown nondominated point, the rectangle is split into two rectangles R^T and R^B defined by points z^T and z^t and z^b and z^B, respectively, with z^t =

$(z_1^B, \frac{z_2^B + z_2^B}{2})$ and $z^b = (z_1^T, \frac{z_2^T + z_2^B}{2})$ That is, the original rectangle is split horizontally along the $z_2(x)$ axis. First, we search rectangle R^B for a nondominated point z^1 minimising $z_1(x)$

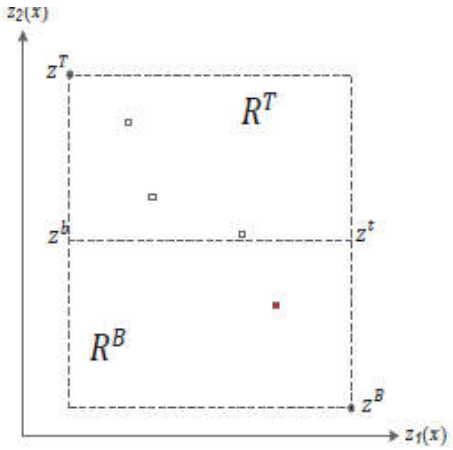


Fig. 3.5 Investigating R^B.

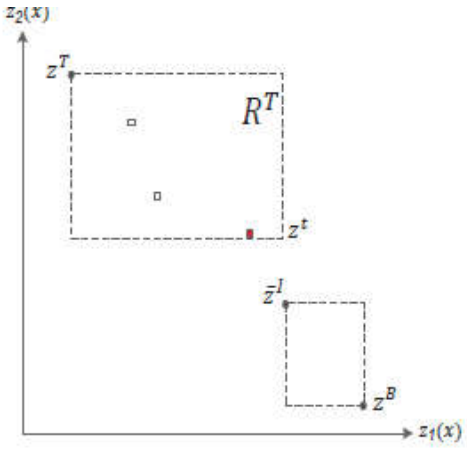
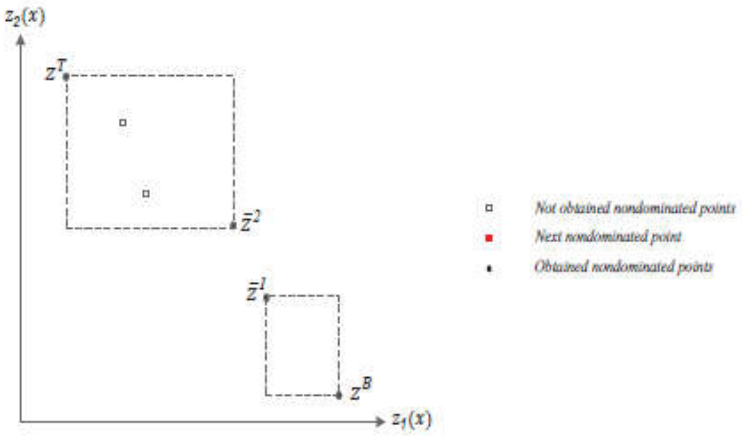


Fig. 3.6 Investigating R^T.



3.7 The new rectangular

by solving min $x \in X$ $\{z_1(x), z_2(x): z(x) \in R^B\}$ (see Figure 3.5). If a new nondominated point is found in rectangle R^B, it identifies a portion of rectangle R^T that is now dominated. Therefore, R^T is modified by setting

$z^t = (\bar{z}_1^1, \frac{z_2^T + z_1^B}{2})$). Next, we search rectangle R^T for a nondominated point $\bar{z}^2$ minimising $z_2(x)$ by solving min $x \in X$ $\{z_2(x), z_1(x) : z(x) \in R^T\}$ (see Figure 3.6). Observe that the search of rectangle R^B returns z^B if and only if z^B is the only nondominated point in rectangle R^B, the search of rectangle R^T returns z^T if and only if z^T is the only nondominated point in rectangle R^B. Moreover, by construction and by the optimality of $\bar{z}^1$ and $\bar{z}^2$, there do not exist nondominated points in $R(\bar{z}^2, \bar{z}^1)$, i.e., any as yet unknown nondominated point has to be in either $R(z^T, \bar{z}^2)$ or in $R(\bar{z}^1 z^B)$ (see Figure 3.7). Therefore, the search for additional nondominated points can be restricted to (and performed independently on) rectangles $R(z^T, \bar{z}^2)$ and $R(\bar{z}^1, z^B)$. See Algorithm 2 for more details.

Algorithm 2: The Balanced Box Method

List. create(L); List. add (L, z^T); List. add (L, z^B)

PQ. create (P); PQ. add (P, R (z^T , z^B))

While not PQ. empty (P) **do**

PQ. pop (P,R (z^1, $\frac{z_2^1 + z_2^2}{2}$))

$R^B \leftarrow R((z_1^1, z_1^1 + z22\ \ 2), z2)$

$\bar{z}^1 \leftarrow$ min $x \in X$ $\{z_1(x), z_2(x) : z(x) \in R^B\}$

if $\bar{z}^1 \neq z^2$ then

List. add (L, $\bar{z}^1$)

PQ. add (P,R($\bar{z}^1$, z2))

$R^T \leftarrow R(z^1, (\bar{z}_1^1 - \varepsilon, \frac{z_2^1 + z_2^2}{2}))$

$\bar{z}^2 \leftarrow$ min $x \in X$ $\{z_2(x), z_1(x) : z(x) \in R^T\}$

if $\bar{z}^2 \neq$ z1 then

List. add (L, $\bar{z}^2$)

PQ.add (P,R (z^1, $\bar{z}^2$))

return L

To reduce the hypervolume indicator of the approximate efficient frontier defined by the known nondominated points at any point during the execution as quickly as possible, the rectangles are processed in nonincreasing order of their area.
The balanced box method has several important features:

- The area of the rectangles constructed during the exploration of a rectangle R (z^1, z^2), i.e., the rectangles defining portions of the criterion space in which there may still be yet unknown nondominated points, is less than 50 percent of the area of the rectangle

R (z^1, z^2). This implies that after exploring v ≥ 1 elements (rectangles) of the priority queue

$$\bar{H}(\bar{y})_N - H(\bar{y}) \leq \frac{1}{2[log2(v+1)]} a\,(R(z^T, z^B)), \quad \ldots\ldots$$

where Y˜N is the set of nondominated points obtained after exploring v elements.

The rectangles R^T and R^B contain at least one nondominated point, so the optimisation problems solved always have a feasible solution.

If the optimization problem returns the known nondominated point for a rectangle, then the rectangle can be discarded from further consideration as it has been proven that it contains no additional nondominated points.

Conclusion: We have introduced a new criterion space search method, the balanced box method, for general bi-objective integer programs and have demonstrated that an intelligent implementation which exploits standard features of modern integer programming solvers outperforms other criterion space search methods for Bi-objective exact model.

References:

Antunes, C. H., Alves, M. J., and Climaco, J. (2016). Multi-objective linear and integer programming. Springer.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M.W., and Vance, P. H. (1998).

Branch-and-price: Column generation for solving huge integer programs. Operations research, 46(3):316–329.

Basirzadeh, H. (2012). One's assignment method for solving assignment problems. Applied Mathematical Sciences, 6(47):2345–2355.
Beckmann, M., K¨unzi, H., Fandel, G., and Truckle, W. (2007). Lecture notes in economics and mathematical systems 586.

Benson, H. and Sun, E. (2000). Outcome space partition of the weight set in multi-objective linear programming. Journal of Optimization Theory and Applications, 105(1):17–36.

Benson, H. P. (1998). An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem. Journal of Global Optimization, 13(1):1–24.

Benson, H. P. and Sun, E. (2002). A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program. European Journal of Operational Research, 139(1):26–41.

Berman, P. and Karpinski, M. (2006). Approximation algorithm for tsp. In Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pages 641–648. Society for Industrial and Applied Mathematics.

Berube, J.-F., Gendreau, M., and Potvin, J.-Y. (2009). An exact-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. European Journal of Operational Research, 194(1):39–50.

Boland, N., Charkhgard, H., and Savelsbergh, M. (2017a). A new method for optimizing a linear function over the efficient set of a multi-objective integer program. European Journal of Operational Research, 260(3):904–919.

Burke, E. K., Kendall, G., et al. (2005). Search methodologies. Springer.

Chalmet, L., Lemonidis, L., and Elzinga, D. (1986). An algorithm for the bi-criterion integer programming problem. European Journal of Operational Research, 25(2):292–300.

Chandy, K. M. and Lo, T. (1973). The capacitated minimum spanning tree. Networks, 3(2):173–181.

Chankong, V. and Haimes, Y. Y. (1983). Optimization-based methods for multi-objective decision-making-an overview. Large Scale Systems In Information And Decision Technologies, 5(1):1–33.

Chebil, K. and Khemakhem, M. (2015). A dynamic programming algorithm for the knapsack problem with setup. Computers & Operations Research, 64:40–50.
Chinchuluun, A. and Pardalos, P. M. (2007). A survey of recent developments in multi-objective optimization. Annals of Operations Research, 154(1):29–50.

Chong, E. K. and Zak, S. H. (2013). An introduction to optimization, volume 76. John Wiley & Sons.

Chu, P.C. and Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. Journal of heuristics, 4(1):63–86.

Dantzig, G. B. (1998). Linear programming and extensions. Princeton university press.

Dauer, J. P. and Liu, Y.-H. (1990). Solving multiple objective linear programs in objective space. European Journal of Operational Research, 46(3):350–357.

De Weck, O. L. (2004). Multi-objective optimization: History and promise. In Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan, volume 2, page 34.

Munapo, E., Kumar, S., Lesaoana, M., and Nyamugure, P. (2016). A minimum spanning tree with index _ 2. ASOR Bullet, 34(1):1–14.

Nemhauser, G. L. and Wolsey, L. A. (1989). Integer and combinatorial optimization, Wiley inter-science series in discrete mathematics and optimization.