Enhancement of the Improved Recursive Method for Multi-objective Integer Programming Problem

Masar Al-Rabeeah$^{1,3}$, Ali Al-Hasani$^{1,3}$, Santosh Kumar$^{1,2}$, Andrew Eberhard$^1$

$^1$ School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Australia.
$^2$ Department of Mathematics and Statistics, University of Melbourne, Melbourne, Australia.
$^3$ Department of Mathematics, College of Sciences, University of Basrah, Al- Basrah, Iraq.

E-mail: masar.alrabeeah@rmit.edu.au
E-mail: ali.alhasani@rmit.edu.au

Abstract.

In this paper, we developed a new algorithm to find the set of a non-dominated points for a multi-objective integer programming problem. The algorithm is an enhancement on the improved recursive method where the authors have used a lexicographic method for analysis. In this approach a sum of two objectives is considered as one weighted sum objective for each iteration. Computational results show that the proposed approach outperforms the currently available results obtained by the improved recursive method with respect to CPU time and the number of integer problems solved to identify all non-dominated points. Many problems such as assignment, knapsack and travelling salesman have been investigated on different sized problems. The benefit of this approach becomes more visible with the increase in the number of objective functions.

Keywords: Combinatorial optimization; Multi-objective integer programming; The improved recursive method; Non-dominated solution.

1. Introduction

Multi-objective optimization problems arise in different fields, including operations research, engineering, computer sciences and economics. Multi-objective optimization problems involve several objectives to be optimized simultaneously. These objectives usually conflict with each other which may lead to the situation where finding an optimal solution for all objectives is impossible. Therefore, another kind of solution, that is, a non-dominated solution becomes of interest to the decision maker. It is a feasible solution that cannot improve one objective without degrading some other objectives. The aim of this paper is to find the whole set of non-dominated points so that the decision maker can choose a suitable solution with regard to other considerations [6],[12],[20].

This paper develops an exact method for the multi-objective integer programming (MOIP) problem by reformulating the improved recursive method (IRM) [16] which becomes faster than the IRM especially when the number of objective functions is large for a given problem. In a multi-objective linear programming problem, the objective functions are linear and the variables are continuous, therefore, the non-dominated set contains only supported non-dominated points [5]. In the case of multi-objective integer programming and multi-objective mixed integer...
programming the non-dominated solution set contains supported and non-supported points which increases the difficulty associated with these problems [1],[2],[18]. In this paper, we deal with a multi-objective problem where variables are restricted to integer values only.

Many researchers have discussed problems with two and three objective functions and have obtained a set of non-dominated solutions. For the case of bi-objectives, we refer to [3],[4],[5],[9],[13] and for the tri-objective program (TOIP) an efficient algorithm discussed in [7], called the L-Shape Method (LSM) is an approach which combines ideas from [11] and [19] and finds the non-dominated points in the criterion space. The disadvantage of this algorithm is that the number of integer programs needing to be solved is more than the number of integer programs required by the algorithm in [11]. A new algorithm, the Quadrant Shrinking Method (QSM) to compute the non-dominated solution for a TOIP was developed in [8]. This algorithm is a variation of [11], and it uses scalarization techniques in two steps. The computational experiments show that QSM performs better compared to the other existing algorithms for TOIP [7],[11].

In this paper, an exact algorithm has been developed for a general MOIP problem with \( k \geq 2 \). One general method is discussed in [19], which generates all non-dominated points for a given MOIP by operating in the decision space. At each step, new constraints and binary variables are inserted along with each new non-dominated point found in the previous step, which increases the complexity and computational effort that is needed to find the whole set of non-dominated solutions. In [14] two exact new algorithms are proposed to compute the non-dominated frontier for a MOIP. The first algorithm is a modification of [19], where the authors minimize the number of constraints and the number of binary variables that were added at each iteration in [19]. The second algorithm in [14] is an improvement over the first, where they decreased the complexity of the first algorithm by utilizing the search steps after excluding the dominated region when searching a new non-dominated point and avoid adding extra binary variables. Their computational study showed that the second algorithm outperformed the first.

Another way to solve a MOIP is by improving the \( \epsilon \)-constraint method, which is recognized as the most commonly used method to solve MOIP. Based on the \( \epsilon \)-constraint method, the algorithm in [15] is developed. It identifies the efficiency with respect to bounds on objective function, by shrinking these bounds at each step according to the new non-dominated point found, all non-dominated solutions are computed. The approach discussed in [15] was further improved in [16] by reducing the number of integer programming problems solved to find the set of non-dominated solutions.

In Section 2, some important concepts for MOIP are presented. Existing methods are reviewed in Section 3. The proposed method is discussed in Section 4. Computational experiments are presented in Section 5 and finally, the paper concludes in Section 6.

2. Preliminaries

Some important definitions and notation essential to understand the proposed method are presented below. In order to maintain uniformity, these definitions and notations have been adopted from [3],[10],[12],[20].

MOIP can be expressed as follows:

\[
\min \left\{ z_1(x), z_2(x), \ldots, z_k(x) \right\},
\]

subject to \( x \in X \) where \( z_i(x) \) represents the \( i^{th} \) objective functions for \( i = 1, \ldots, k \).

Let \( X \) represent the feasible set in the decision space and \( Y \) represent the feasible set in the criterion space such that \( x_j \in \mathbb{Z} \) for all \( j = 1, 2, \ldots, n \), where \( n \) is the number of variables.

The objective function: \( z_i(x) = \sum_{j=1}^{n} a_{ij} x_j \), where \( a_{ij} \in \mathbb{Z}, i = 1, \ldots, k \) and \( j = 1, 2, \ldots, n \).
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 $z_i(x) \leq z_i(x^*)$ for each $i$ and $z_i(x) \prec z_i(x^*)$ for at least one $i$. The image $z(x^*)$ of efficient point $x^*$ 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 Let $x^* \in X_E$ an efficient solution, If there is some positive value $\lambda$ such that $x^* \in X_E$ is an optimal solution of $\min_{x^* \in X} \lambda^T z(x^*)$ then is termed as supported non-dominated solution otherwise it is called non-supported non-dominated solution.

Definition 3 An efficient solution $x^* \in X$ in the decision space is called strictly efficient if there is no $x \in X$ such that $z_i(x) \leq z_i(x^*)$ for each $i$, image of the strictly efficient solution in the criterion space called strictly non-dominated solution.

Definition 4 A feasible point in the criterion space is called weakly non-dominated point if and only if it is not strictly dominated by any other non-dominated point.

3. Review of existing methods

3.1. The $\epsilon$-constraint method [20]
This method is the most popular scalarization method for finding all non-dominated solutions of a MOIP. It converts the multi-objective problem into a single objective by optimizing one objective function and converting the other objective functions into inequality constraint restricted by $\epsilon_i$ value as a bound which can be set as the maximum value of each corresponding objective function for $i = 2, \ldots, k$. The method repeatedly computes all sets of the non-dominated points by iterating the $\epsilon_i$ value of the constraints strategically. This means the $\epsilon_i$ value starts in the first iteration as the upper bound then it changes in the second iteration according to the value of the new non-dominated point. When the $\epsilon_i$ value reaches to the lower bound the algorithm terminates.

$$
\min z_k(x), \\
\text{s.t. } x \in X, \quad z_i(x) \leq \epsilon_i, \text{ for all } i=1, \ldots, k-1.
$$

The main drawback of the $\epsilon$-constraint method is that it only enumerates the weakly non-dominated points. In order to compute the non-dominated points, the weighted sum method [5] has been combined with the $\epsilon$-constraint method, as in [6],[15]. In this combination, the weight assigned to the objective function to be optimized is 1 and a small weight $w$ is assigned to other objective functions, as shown below:

$$
\min \left\{ z_k(x) + w \sum_{i=1}^{k-1} z_i(x) \right\}, \\
\text{s.t. } x \in X, \quad z_i(x) \leq \epsilon_i, \text{ for all } i=1, \ldots, k-1.
$$

3.2. Lexicographic method [17]
The lexicographic method requires information about the objective function to be arranged in terms of importance, either provided by the decision maker or taken in the order as given in the model i.e. importance associated with $z_1$ is $\geq$ the importance associated with $z_2 \geq \ldots \geq$ importance associated with $z_{k-1}$. Therefore, this method is also known as a priori method because it requires sufficient ranking information. It optimizes the objective with the highest
importance first and then the next and so on with additional equality constraints. It solves a single objective function as follow:

\[
\begin{align*}
\min & \{ z_1(x) \}, \text{ s.t. } x \in X, \\
\min & \{ z_2(x) \}, \text{ s.t. } x \in X, \quad z_1 = z_1^*, \\
\min & \{ z_k(x) \}, \text{ s.t. } x \in X, \quad z_1 = z_1^*, z_2 = z_2^*, \ldots, z_{k-1} = z_{k-1}^*
\end{align*}
\]  

Where \( z_1^*, z_2^*, \ldots, z_{k-1}^* \) are the optimal solution of \( z_1, z_2, \ldots, z_{k-1} \), respectively. Each point represents a non-dominated point but to compute the complete set of non-dominated points, this approach is considered in association with other approach [16]. Another drawback of this method is that the computational effort required is more than what is needed for the \( \epsilon \)-constraint method.

### 3.3. The Improved Recursive Method (IRM) [15,16]

The IRM is an improvement over the \( \epsilon \)-constraint method. The authors in [15],[16] used the recursive method to convert \( k \) problems into \( (k - 1) \) number of problems, then \( (k - 2) \) until the reduced number of problems is equal to 2. The solutions of each \( (k - 1) \) problem is also a non-dominated solution of the original \( k \) objective problem, where \( k \) represents the number of objectives.

Suppose there is a MOIP:

\[
\begin{align*}
\min & \{ z_1(x), z_2(x), \ldots, z_k(x) \} \\
\text{s.t. } x & \in X.
\end{align*}
\]

The first version in [15] considers:

\[
\begin{align*}
\min & \{ z_1(x) + w_k z_k(x) \}, \\
\min & \{ z_2(x) + w_k z_k(x) \}, \\
\vdots \\
\min & \{ z_{k-1}(x) + w_k z_k(x) \} \\
\text{s.t. } z_k(x) & \leq \epsilon_k, \quad x \in X
\end{align*}
\]  

Where \( \epsilon_k \) is the upper bound for the objective function \( k \) and 

\[
W_k = \frac{1}{(z_2^{GUP} - z_2^{GLP} + 1)(z_3^{GUP} - z_3^{GLP} + 1)\ldots(z_k^{GUP} - z_k^{GLP} + 1)}
\]

is a small quantity. Here \( z_i^{GUP}, z_i^{GLP} \) are the global upper bound and global lower bound for objective \( z_i(x) \).

The problem is solved repeatedly by reducing the value of \( \epsilon_k \) by 1 in each iteration. In the improved version [16], the authors used the lexicographic method [17] with the same procedure to find the non-dominated set. However, relaxation is used here to avoid repeated calculation by using the history of previously solved problems.

For example, if there are two problems \( p \) and \( q \) as below:

Problem \( p \):

\[
\begin{align*}
\min & \{ z_1(x), z_2(x), \ldots, z_k(x) \},
\end{align*}
\]  

(p)
subject to $x \in X$ where $z_k(x) \leq \epsilon_k$.

Problem $q$:

$$\min \left\{ z_1(x), z_2(x), \ldots, z_k(x) \right\}, \quad (q)$$

subject to $x \in X$ where $z_k(x) \leq \epsilon_k$.

If $\epsilon_k^\star \leq \epsilon_k$, then the feasible set of $p$ is included in $q$, so it means $q$ is a relaxation of $p$, hence there is no need to solve two problems rather only one needs to be solved. IRM is one of the recent efficient algorithms for a general MOIP.

4. Enhancement on the Improved Recursive Method (EIRM)

In the proposed approach we solve a problem with two objective functions at each stage. Note that in the equation (4) of the two objective functions, objective $z_k(x)$ is common to all other objectives $1, \ldots, k-1$. In the proposed approach, we combine objectives one and two as one problem and then combine objective three and four and so on, as shown in (5) and (6). This simple modification reduces the number of integer problems (IPs) from $(k-1)$ to $\binom{k}{2}$, i.e. if $k = 2K$, we are solving $K$ integer problems as given in (5) while if $(k = 2K + 1)$ we solve $(K+1)$ integer problems as given in (6). This idea enhances the performance compared to the IRM. It reduces the CPU time and also the number of IPs.

Suppose that we have a multi-objective integer problem:

$$\min \left\{ z_1(x), z_2(x), \ldots, z_k(x) \right\}$$

s.t. $x \in X$. In the EIRM, the constrained weighted multi-objective is as follows:

Case 1 when $k$ is even:

$$\min \left\{ z_1(x) + w_2z_2(x) \right\}, \quad \min \left\{ z_3(x) + w_4z_4(x) \right\},$$

$$\vdots$$

$$\min \left\{ z_{k-1}(x) + w_Kz_k(x) \right\}$$

s.t. $x \in X$

$z_2(x) \leq \epsilon_2, \ldots, z_k(x) \leq \epsilon_k$. \hspace{1cm} (5)

Case 2 when $k$ is odd:

$$\min \left\{ z_1(x) + w_2z_2(x) \right\}, \quad \min \left\{ z_3(x) + w_4z_4(x) \right\},$$

$$\vdots$$

$$\min \left\{ z_k(x) \right\}$$

s.t. $x \in X$

$z_2(x) \leq \epsilon_2, \ldots, z_k(x) \leq \epsilon_k$. \hspace{1cm} (6)
The above formulation is a combination of the $\epsilon$-constraint method and the weighted sum method using the weighted sum objective as shown in (5) and (6). First, we find the global upper bound and global lower bound for each objective and denote them as $z_{i}^{GUP}, z_{i}^{GLP}$ for $i = 1, \ldots, k$. These bounds are calculated by solving single-objective integer programming problems for each objective. We use the upper bound $z_{i}^{GUP}$ to initialize the value of $\epsilon_{i}$ for $i = 2, \ldots, k$, then we solve the optimization problem as in (5) or (6). Since (5) and (6) are similar processes, so for further explanation let us consider the weighted sum objective (5).

In (5) we solve first $\min \left\{ z_{1}(x) + w_{2}z_{2}(x) \right\}$ and check if the solution is feasible and then we save this solution, and solve $\min \left\{ z_{3}(x) + w_{4}z_{4}(x) \right\}$ and so on. If the solution is infeasible at any stage, the algorithm switches to step 6 to change the value of $\epsilon_{i}$ as shown in EIRM algorithm 1. During this process, the algorithm skips over the repeated problems using relaxation to avoid repeated calculation as discussed in section 3.3. The values of $\epsilon_{i}$ are updated according to the last non-dominated point found. If the value of $\epsilon_{i}$ reaches the global lower bound, the process will terminate. This process finds all the non-dominated points and does not miss any. We also attempted the weighted sum of all objective functions as one problem. However, we note that the weighted sum for two objectives is more efficient than three or more objectives when using CPLEX as a solver. Further, a combination of two objectives is more efficient than one as used in the lexicographic method because it minimizes the computational effort and the CPU time.

**Theorem 4.1.** For $x^* \in X$ if $y^* = z(x^*)$ is a non-dominated solution for the proposed weighted sum-objective (5) then it is a non-dominated solution for the general MOIP (1).

**Proof.** Suppose $y^* = z(x^*)$ is a non-dominated solution for (5) but it is dominated for (1), so there exist another point $x \in X, y = z(x)$ such that for each $i = 1, \ldots, k$

$$z_{i}(x) \leq z_{i}(x^*) \quad (7)$$

As $w$ is a small positive value [20], from (7) for $i = 2$ we have

$$w_{2}z_{2}(x) \leq w_{2}z_{2}(x^*) \quad (8)$$

Combining equation (7) when $i = 1$ with equation (8), we have

$$z_{1}(x) + w_{2}z_{2}(x) \leq z_{1}(x^*) + w_{2}z_{2}(x^*)$$

Similarly, one can write

$$z_{j}(x) + w_{j+1}z_{j+1}(x) \leq z_{j}(x^*) + w_{j+1}z_{j+1}(x^*)$$

for $j = 3, 5, \ldots, k - 1$.

That means $y^* = z(x^*)$ is also a dominated solution for (5) and that contradicts the assumption. Hence $y^* = z(x^*)$ must be a non-dominated solution for (1). ■

5. Computational analysis

To investigate the performance of the proposed algorithm, it was implemented by using Cplex Callable Library, which is a C programming language linked with CPLEX 12.5 as a solver for an integer programming problem. All experiments were conducted on a Dell Inc.OptiPlex 9020 with processor Intel (R) Core (TM) i7-4770 CPU@ 3.40 GH and RAM 4.00 GB, Lubuntu operating system 16.04.01.

The performance of the EIRM algorithm was compared with the IRM by considering the same instances as those used previously by IRM [16] for the assignment problems in the case
Algorithm 1 EIRM

Step 0. Find $z_{i}^{GUP}$, $z_{i}^{GLP}$ for $i = 1, \ldots, k$.

Step 1. Set up the initial values.

Step 1.1. Set $\epsilon_{k} = z_{k}^{GUP}$

Step 1.2. Set $\epsilon_{k-1} = z_{k-1}^{GUP}$

$\vdots$

Step 1.(k-1). Set $\epsilon_{2} = z_{k}^{GUP}$

Step 2. If $\exists (z_{1}^{*}, z_{2}^{*}, \ldots, z_{k}^{*}, \epsilon_{1}^{*}, \epsilon_{2}^{*}, \ldots, \epsilon_{k}^{*}) \in Y_{N}$ such that $\epsilon_{2}^{*} \geq \epsilon_{2}, \epsilon_{3}^{*} \geq \epsilon_{3}, \ldots, \epsilon_{k}^{*} \geq \epsilon_{k}$ and $z_{2}^{*} \leq z_{2} \leq \epsilon_{2}, z_{3}^{*} \leq \epsilon_{3}, \ldots, z_{k}^{*} \leq \epsilon_{k}$ then $\epsilon_{2} = \epsilon_{2} - 1$, repeat step 2.

Step 3. If $k = 2K$ solve (5) with $\epsilon_{2}, \epsilon_{3}, \ldots, \epsilon_{k}$ else if $(k = 2K + 1)$ solve (6) with $\epsilon_{2}, \epsilon_{3}, \ldots, \epsilon_{k-1}$.

Step 4. If the solution is infeasible, then go to step 6.

Step 5. Let the solution $(z_{1}, z_{2}, \ldots, z_{k})$ add it to $Y_{N}$ and $\epsilon_{2} = \epsilon_{2} - 1$ go to step 2.

Step 6. Determination of the bounds.

Step 6.1. If $\epsilon_{2} \geq z_{2}^{GLP}$ go to step 2

Step 6.2. If $\epsilon_{3} \geq z_{3}^{GLP}$, then $\epsilon_{3} = \max(z_{3}^{*}) - 1$ such that $\epsilon_{4} = \epsilon_{4}^{*}, \ldots, \epsilon_{k} = \epsilon_{k}^{*}$, go to step 1.(k-1).

$\vdots$

Step 6.(k-2). If $\epsilon_{k-1} \geq z_{k-1}^{GLP}$ then $\epsilon_{k-1} = \max(z_{k-1}^{*}) - 1$ such that $\epsilon_{k} = \epsilon_{k}^{*}$, go to step 1.3.

Step 6.(k-1). If $\epsilon_{k} \geq z_{k}^{GLP}$ then $\epsilon_{k} = \max(z_{k}^{*}) - 1$ go to step 1.2. otherwise go to step 7.

Step 7. Stop.

of the three objective in Table 1. In Table 2 only two instances i.e. AP10 * 10 and AP20 * 20 marked with asterisk symbol are the same as those used in [16]. All other instances are different in Tables 1 and 2. These instances are indicated by: Assignment problem (AP), Knapsack Problem (KP) and the Travelling Salesman Problem (TSP) as shown in Tables 1 and 2.

Solutions* indicates the number of non-dominated points. The percentage of improvement for large instances of TOIP from Table 1, for instances AP50 * 50 is 7%, KP80 is 13% and for TSP − 012 is 15%.

A significant enhancement appears in Table 2 when the number of objectives are four and five, for example the improvement for instance AP20 × 20 is 44% and for the instance TSP − 012 is 47%. It is clear that the improvement in four objectives is much more than the improvement in three objectives. In terms of five objectives for example, the improvement for the instance AP10 * 10 is 74%. The improved percentages are shown in bold in Tables 1 and 2.

As a consequence, many algorithms can solve TOIP efficiently, however, the proposed algorithm is a general algorithm and it outperformed the IRM when solving four, five and possibly a greater number of objectives see (Figure1).
Table 1: Comparison between the EIRM and IRM in terms of CPU time and IPs when $k = 3$

| Problem | Solutions* | EIRM | IRM |
|---------|------------|------|-----|
|         |            | CPU time | IPs | CPU time | IPs |
| AP$10 \times 10$ | 221 | 12.7474 | 800 | 13.2190 | 1158 |
| AP$20 \times 20$ | **1942** | **278.0612** | 6101 | **296.8623** | 9055 |
| AP$30 \times 30$ | 5195 | 1116.9050 | 15045 | 1308.3590 | 22410 |
| AP$40 \times 40$ | 14733 | 4743.4770 | 37431 | 4962.4776 | 55935 |
| AP$50 \times 50$ | 29193 | 12747.0341 | 72972 | 13621.2936 | 109142 |
| KP20 | 20 | 0.3120 | 85 | 0.4168 | 119 |
| KP30 | 35 | 0.9777 | 148 | 1.0932 | 208 |
| KP40 | 117 | 4.8460 | 454 | 5.4247 | 639 |
| KP60 | 578 | 38.155 | 2112 | 44.940 | 3068 |
| KP80 | **1082** | **168.8867** | 3709 | **193.2458** | 5430 |
| TSP − 008 | 72 | 6.7406 | 336 | 8.3445 | 469 |
| TSP − 012 | **346** | **166.8654** | 1654 | **195.2864** | 2324 |

Figure 1: CPU time comparison between EIRM and IRM for assignment instances
Table 2: Comparison between EIRM and IRM in terms of CPU time and IPs when $k = 4, 5$

| problem       | $k$ | Solution | EIRM     | IRM     |
|---------------|-----|----------|----------|---------|
|               |     |          | CPU time | IPs solved | CPU time | IPs solved |
| $AP05 \times 5$ | 4  | 427      | 42.3838  | 5813     | 43.5925  | 8856       |
| $*AP10 \times 10$ | 4 | 756      | 247.5760 | 11102    | 296.3428 | 16268      |
| $AP15 \times 15$ | 4 | 22130    | 6636.5128| 167631   | 10030.5623| 300348     |
| $*AP20 \times 20$ | 4 | 22837    | 12716.4237| 180514   | 19067.0620| 323703     |
| $TSP - 008$    | 4  | 121      | 124.9964 | 5604     | 148.3992 | 6623       |
| $TSP - 012$    | 4  | 1794     | 34948.1765| 245870   | 53454.6736| 264507     |
| $AP05 \times 5$ | 5  | 1931     | 2281.4647| 210169   | 5942.5408| 243872     |
| $AP10 \times 10$ | 5 | 2949     | 12259.0946| 502490   | 45535.0820| 560004     |
| $AP15 \times 15$ | 5 | 5013     | 15727.1645| 405297   | 30650.5105| 469604     |

6. Conclusions
In this paper, the performance of the IRM [16] is enhanced by a simple modification. This enhancement is based on re-arranging the weight sum objective functions which reduces the CPU time and also the number of IPs significantly. From the computational experiments, the improvements became more significant when more objective functions were considered. Future work will involve applying this algorithm to practical problems in scheduling, logistics and supply chain management. More future work can be by utilizing this work to solve the bi-objective generalizing assignment problem as was done in [2].

Acknowledgements
Authors are grateful to the ICoMPAC 2019 conference organizers for their cooperation and constructive suggestions.

References
[1] Al-Hasani, Ali, Al-Rabeeah, Masar, Kumar, Santosh, and Eberhard, Endrew. “An Improved CPU Time in Triangle Splitting Method for Solving a Biobjective Mixed Integer Program”. In: International Journal of Mathematical, Engineering and Management Sciences 3.4 (2018), pp. 351–364.
[2] Al-Hasani, Ali, Al-Rabeeah, Masar, Kumar, Santosh, and Eberhard, Andrew. “Finding all non-dominated points for a bi-objective generalized assignment problem”. In: Journal of Physics: Conference Series. Vol. 1218. 1. IOP Publishing, 2019, p. 012004.
[3] Al-Rabeeah, Masar, Kumar, Santosh, Al-Hasani, Ali, Munapo, Elias, and Eberhard, Andrew. “Bi-objective integer programming analysis based on the characteristic equation”. In: International Journal of System Assurance Engineering and Management 10.5 (2019), pp. 937–944.
[4] Al-Rabeeah, Masar, Munapo, Elias, Al-Hasani, Ali, Kumar, Santosh, and Eberhard, Andrew. “Computational Enhancement in the Application of the Branch and Bound Method for Linear Integer Programs and Related Models”. In: International Journal of Mathematical, Engineering and Management Sciences 4.5 (2019), pp. 1140–1153.
[5] Aneja, Yash P and Nair, Kunhiraman PK. “Bicriteria transportation problem”. In: Management Science 25.1 (1979), pp. 73–78.
[6] Antunes, Carlos Henggeler, Alves, Maria João, and Clímaco, João. Multiobjective linear and integer programming. Springer, 2016.
[7] Boland, Natasha, Charkhgard, Hadi, and Savelsbergh, Martin. “The L-shape search method for triobjective integer programming”. In: Mathematical Programming Computation 8.2 (2016), pp. 217–251.
[8] Boland, Natashia, Charkhgard, Hadi, and Savelsbergh, Martin. “The Quadrant Shrinking Method: A simple and efficient algorithm for solving tri-objective integer programs”. In: European Journal of Operational Research 260.3 (2017), pp. 873–885.

[9] Chahmet, LG, Lemonidis, L, and Elzinga, DJ. “An algorithm for the bi-criterion integer programming problem”. In: European Journal of Operational Research 25.2 (1986), pp. 292–300.

[10] Chankong, Vira and Haines, Yacov Y. Multiobjective decision making: theory and methodology. Courier Dover Publications, 2008.

[11] Dächert, Kerstin and Klamroth, Kathrin. “A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems”. In: Journal of Global Optimization 61.4 (2015), pp. 643–676.

[12] Ehrhoff, Matthias. Multicriteria optimization. Springer Science & Business Media, 2006.

[13] Hamacher, Horst W, Pedersen, Christian Roed, and Ruzika, Stefan. “Finding representative systems for discrete bicriterion optimization problems”. In: Operations Research Letters 35.3 (2007), pp. 336–344.

[14] Lokman, Banu and Köksalan, Murat. “Finding all nondominated points of multi-objective integer programs”. In: Journal of Global Optimization 57.2 (2013), pp. 347–365.

[15] Özlen, Melih and Azízoğlu, Meral. “Multi-objective integer programming: a general approach for generating all non-dominated solutions”. In: European Journal of Operational Research 199.1 (2009), pp. 25–35.

[16] Özlen, Melih, Burton, Benjamin A, and MacRae, Cameron AG. “Multi-objective integer programming: An improved recursive algorithm”. In: Journal of Optimization Theory and Applications 160.2 (2014), pp. 470–482.

[17] Stanimirovic, Ivan P. “Compendious lexicographic method for multi-objective optimization”. In: Ser. Math. Inform 27.1 (2012), pp. 55–66.

[18] Stidsen, Thomas, Andersen, Kim Allan, and Dammann, Bernd. “A branch and bound algorithm for a class of biobjective mixed integer programs”. In: Management Science 60.4 (2014), pp. 1009–1032.

[19] Sylva, John and Crema, Alejandro. “A method for finding the set of non-dominated vectors for multiple objective integer linear programs”. In: European Journal of Operational Research 158.1 (2004), pp. 46–55.

[20] Vira, Chankong and Haines, Yacov Y. “Multiobjective decision making: theory and methodology”. In: North Holland series in system science and engineering. 8. North-Holland, 1983.