18. Matsuura, M. (2014). Asymptotic Behaviour of the Maximum Curvature of Lame Curves. Journal for Geometry and Graphics, 18 (1), 45–59.
19. Dura, E., Bell, J., Lane, D. (2008). Superellipse Fitting for the Recovery and Classification of Mine-Like Shapes in Sidescan Sonar Images. IEEE Journal of Oceanic Engineering, 33 (4), 434–444. doi: https://doi.org/10.1109/joe.2008.2002962
20. Khoomwong, E., Phongcharoenpanich, C. (2016). Design of a Dual-Band Bidirectional Antenna Using Superellipse-Monopole-Fed Rectangular Ring for IEEE 802.11 a/b/g/n Applications. International Journal of Antennas and Propagation, 2016, 1–11. doi: https://doi.org/10.1155/2016/9368904
21. Dos Santos, R. A., Penchel, R. A., Rehder, G. P.,Spadoti, D. H. (2019). Omnidirectional Ultra-wideband Superellipse Patch Antenna for mm-Waves Applications. 2019 PhotonIcs & Electromagnetics Research Symposium - Spring (PIERS-Spring). doi: https://doi.org/10.1109/piers-spring46901.2019.9017517
22. Duchemin, M., Tugui, C., Collee, V. (2017). Optimization of Contact Profiles using Super-Ellipse. SAE International Journal of Materials and Manufacturing, 10 (2), 234–244. doi: https://doi.org/10.4271/2017-01-1349
23. Forsayt, Dzh., Mal'kol'm, M., Mouler, K. (1980). Mashinnye metody matematicheskih vychisleniy. Moscow: Mir, 279.

UDC 519
DOI: 10.15587/1729-4061.2020.198849

IMPROVEMENT OF THE BRANCH AND BOUND ALGORITHM FOR SOLVING THE KNAPSACK LINEAR INTEGER PROBLEM

Elias Munapo
PhD, Professor of Operations Research
Department of Statistics and Operations Research
School of Economics and Decision Sciences
North West University
Mmabatho Unit 5, Mahikeng, 2790, Mafikeng, South Africa
E-mail: emunapo@gmail.com

У статті представлений новий підхід до переформулювання, що дозволяє зменшити складність алгоритму розгалуження і меж для вирішення лінійної цілочисельної задачі про рюкзак. Алгоритм розгалуження і обмеження в цілому спирається на звичайну стратегію, яка полягає в першому ослабленні цілочисельного завдання в моделі лінійного програмування (ЛП). Якщо оптимальне рішення лінійного програмування є цілочисельним, то є оптимальне рішення цілочисельного завдання. Якщо оптимальне рішення лінійного програмування не є цілочисельним, то обирається змінна з дробовим значенням для створення двох підзадач, так що частина допустимої області відкидається без усунення будь-якого з можливих цілочисельних рішень. Процес повторюється для всіх змінних з дробовими значеннями, поки не буде знайдено цілочисельне рішення. У цьому підході змінна сума і додаткові обмеження генеруються і додаються до вихідної задачі перед її рішенням. Для цього швидко визначається об'єктивна межа задачі про рюкзак. Потім межа використовується для генерації набору меж змінної суми і чотирьох додаткових обмежень. Виходячи за межі змінної суми, вихідні підзадачі будуються і вирішуються. Оптимальне рішення потім виходить як краще рішення з усіх підзадач з точки зору об'єктивного значення. Пропонована процедура призводить до підзадач, які мають меншу складність і легше вирішуються, ніж вихідна задача, з точки зору кількості гілок і пов'язаних ітерацій або підзадач.

Задача про рюкзак – це особлива форма загальної лінійної цілочисельної задачі. Є багато видів задач про рюкзак. Вони включають в себе задачі «нуль-один», «множинного вибору», «обмежену», «необмежену», «квадратичну», «багатоцільову», «багатовимірну», «колапсу нуль-один» та задачу про об'єднання рюкзаків. Задачі про рюкзаки «нуль-один» – ті, в яких змінні приймають тільки 0 і 1. Причина в тому, що предмет може бути обрано або не обрано. Іншими словами, немає можливості отримати дробові суми або предмети. Це найпростіший клас завдань про рюкзаки, і він єдиний, який може бути вирішений в поліномі за допомогою алгоритмів внутрішніх точок і в псевдополіноміальному часі за допомогою методів динамічного програмування. Задачі з множинним вибором рюкзаків – це узагальнення звичайної задачі про рюкзаки, коли набір предметів розбивається на класи. Нульовий варіант вибору предмета замінюється вибором рівно одного предмета з кожного класу предметів

Ключові слова: цілочисельна задача про рюкзаки, переформулювання, алгоритм гілок і меж, унімодулярний, обчислювальна складність

Received date 19.02.2020
Accepted date 20.04.2020
Published date 30.04.2020

1. Introduction

In general the linear integer programming problem has very important real life applications. The general linear integer problem comes in the form of capital budgeting, transportation, traveling salesman, facility location, scheduling, knapsack etc. This model even though it is very easy to model mathematically, has proved to be very difficult to solve. See [1–5] for more on linear integer models.

The paper presents a new reformulation approach to reduce the complexity of a branch and bound algorithm for solving the knapsack linear integer problem. The branch and bound algorithm [6, 7] in general relies on the usual strategy of first relaxing the integer problem into a linear programing (LP) model. If the linear programming optimal solution is integer then, the optimal solution to the integer problem is available. If the linear programming optimal solution is not integer, then a variable with a fractional value is selected to create two sub-problems such that part of the feasible region is discarded without eliminating any of the feasible integer solutions. The process is repeated on all variables with fractional values until an integer solution is found. In this approach variable sum and additional constraints are generated and added to the original problem before solving. In order to do this the objective bound of knapsack problem is quickly determined. The bound is then used to generate a set of variable sum limits and four additional constraints. From the variable sum limits, initial sub-problems are constructed and solved. The optimal solution is then obtained as the best solution from all the sub-problems in terms of the objective value. The proposed procedure results in sub-problems that have reduced complexity and easier to solve than the original problem in terms of numbers of branch and bound iterations or sub-problems.

Knapsack problem reformulation is not a new idea. The reformulation approaches were once used to solve some knapsack and other problems [8, 9].

The knapsack problem is a special form of the general linear integer problem. There are so many types of knapsack problems. These include the zero-one, multiple, multiple-choice, bounded, unbounded, quadratic, multi-objective, multi-dimensional, collapsing zero-one and set union knapsack problems. The zero-one knapsack [10, 11] problem is one in which the variables assume 0 s and 1 s only. The reason being that an item can be chosen or not chosen. In other words there is no way it is possible to have fractional amounts or items. This is the easiest class of the knapsack problems and is the only one that can be solved in polynomial by interior point algorithms and in pseudo-polynomial time by dynamic programming approaches. The multiple-choice knapsack problem is a generalization of the ordinary knapsack problem, where the set of items is partitioned into classes. The zero-one choice of taking an item is replaced by the selection of exactly one item out of each class of items.

The multiple knapsack problem [12–14] is a generalization of the standard knapsack problem formed by combining single knapsacks into a group of knapsacks having different capacities. In this case the objective is to assign each item to at most one of the knapsacks in such a way that all capacity constraints are satisfied and that the total profit of all the items put into knapsacks is made maximum. In the bounded knapsack problem [15] there is a knapsack capacity and a set of items, each having a positive integer value, a positive integer weight, and a positive integer limit or bound on its availability. With the bounded knapsack problem the main objective is to select the number of each item type to add to the knapsack in such a way that the total weight is not violated and that the total value is a maximum.

2. Literature review and problem statement

In this case of the unbounded knapsack problem, types of items of different values and volumes are given, then it is required to find the most valuable set of items that fit in a knapsack of fixed volume. The main difference with the bounded knapsack problem is that the number of items of each type is unbounded. A quadratic knapsack [16–19] is a knapsack problem whereby the objective is expressed as a quadratic function subject to a set of linear constraints. The variables in this knapsack problem can either be zero-one or general integers. With the multi-objective knapsack, the objective changes from a single objective into many objectives within the same problem. For example in agriculture, there is an objective to maximize profit and at the same time minimizing transportation costs and maximizing the number of employees. In multi-objective [20, 21] knapsack problems there is the dilemma of dealing with environmental, social, political and or economic concerns. In the multidimensional knapsack problem, several dimensions are considered in the formulation of the problem. The multidimensional knapsack [22–25] problem basically consists of finding a subset of objects that maximizes the total profit while observing some capacity restrictions.

The collapsing zero-one knapsack problem is a type of non-linear knapsack problem in which the knapsack size is a non-increasing function of the number of items included. The set-union knapsack [26] problem is a variation of the zero-one knapsack problem in which each item is a set of elements, each item has a nonnegative value, and each element has a nonnegative weight. The weight of one item is given by the total weight of the elements in the union of the items' sets.

The branch and bound was the first algorithm to be developed in 1960 [6] for these linear integer models. This method was further modified in 1965 to solve the mixed linear integer problem [7]. So many improvements have been done on the branch and bound algorithm in terms of addition of cuts to get the branch and cut algorithm [19, 27–29]. Pricing was introduced within the context of branch and bound to get the branch and price algorithm [30, 31]. The improved versions, branch and cut and branch and price were also combined to get the branch, cut and price [32–34]. In addition to using cuts and pricing within the context of a branch and bound algorithm, preprocessing can reduce the number of sub-problems needed to verify optimality. Even with all these efforts the general linear integer is still very difficult to solve. In fact the general linear integer problem including the knapsack problem is NP hard [10, 18, 19, 26, 35, 36] and there are not aware of any consistent efficient algorithm for these problems. These difficult problems and include the knapsack problem which is a special case with only one constraint.

The proposed algorithm has the advantage that it is parallelizable and independent processors can be used. The knapsack problem has so many real life applications. These include home energy management, cognitive radio networks, mining operations use, relay selection in secure cooperative wireless communication, electrical power allocation management, production planning, in selection of renovation actions, waste management, formulation and solution method for tour conducting and optimization of content delivery networks. Network of electricity that intelligently integrates the users'.

The knapsack problem has so many real life applications. These include home energy management [37], cognitive radio networks [38], mining operation use [39], relay selection in secure cooperative wireless communication [40], electrical power allocation management [41], production planning [42], in selection of renovation actions [43], waste management [44], formulation and solution method for tour conducting and optimization of content delivery networks [45].

Nowadays electricity network grid which incorporates the user's input or actions is now being used and is known as a smart grid. This smart grid is very important for sustainable, economical and secure supply of electrical power to the people. Knapsack optimization is used in the management and distribution of power.

Knapsack problem formulation is used in channel and power allocation for cognitive radio (CR) networks. In this formulation it is assumed that the total available spectrum is divided into several bands, each consisting of a group of channels. A centralized base station, enabled by spectrum sensing, is assumed to have the knowledge of all vacant channels, which will be assigned to various CRs according to their requests. In this case the objective of resource allocation is to maximize the sum data rate of all CRs.

An extension of the precedence constrained knapsack problem where the knapsack can be filled in multiple periods has applications in the mining operations. This problem formulation is known in the mining industry as the open-pit mine production scheduling problem. Both exact and heuristics are used in solving the LP relaxation of this problem.

Knapsack formulation is used in cooperative jamming. Cooperative jamming schemes support secure wireless communication in the presence of more eavesdroppers. Large numbers of cooperative relays provide better secrecy rate while increasing the communication ad synchronization needs associated with cooperative beam forming.

When renovating a building structure there is a need for the construction manager to select the most feasible renovation activities and the order in which they must be done. The main challenge of renovating a building structure is to determine whether to renovate the existing structure or start building a new building. Such a decision requires use of decision tools such as knapsack modeling.

Waste is another challenge that may cause serious environmental damage if not properly managed. Plastic and paper waste is a serious issue in most developing countries. The many problem of waste in developing countries is that there is very low level recycling in these countries. In other words there is a small amount of plastic and paper waste that is recycled and the rest is sent to the landfills or dumbed on the streets. For recycling to be financially profitable there is need to use effective and efficient ways in selecting items to be produced from a lot of items given a limited amount of money and other resources. This is where knapsack modeling is applied to minimize waste management costs.

Land conservation projects require proper managing and planning for the benefits to be seen. For these land projects there is always the dilemma of how to select the most profitable land projects subject to financial constraints. A multiple knapsack formulation is employed in making such decisions and it outperforms other decision making tools such as benefit targeting, cost-effectiveness analysis, and sequential binary integer programming.

The fast growing populations and introduction of healthcare systems have resulted in increased both inpatients and outpatients to public hospitals, particularly those hospitals that provide special and comprehensive health services in large countries such as China and India. The hospitals in these countries have huge numbers of both inpatients and outpatients. The huge numbers of patients result in overcrowding and these overcrowding conditions are a concern for the hospital managers. The obvious question is how to manage these huge numbers of patients effectively given the fact that some patients require less attention than the others, the lengths of some patients are predictable than the others. In other words those who require less clinical care are less likely to stay longer at the hospital. In order to alleviate the challenge of overcrowding, a multi-criteria knapsack model is used for disease selection in the reception or observation ward of the public hospitals.

So many studies have been done in the field of optimization methods. Unfortunately, sometimes it is not possible to directly apply these optimization methods to practical problems. As an example, a tourist deciding on a traveling schedule within a traveling time limit needs to select travel tourist spots from lists so as to be satisfied as far as possible. This is a problem that can't be solved by the available conventional methods. This problem is formulated as a tour conducting knapsack problem. The formulation and solution method of tour conducting knapsack problem are based on those of traveling salesman problem and knapsack problem.

The knapsack problem has many very important applications in so many areas of business and engineering and it is certainly very necessary to develop efficient solution algorithms for it. Most of the algorithms for the knapsack problem are branch and bound based and in this paper the branch and bound algorithm is improved for the knapsack problem.

The knapsack problem has so many applications [37–45] and there is definitely a need for efficient and consistent algorithms for this problem. Some of the applications are home energy management [37], cognitive radio networks [38], mining operation use [39], relay selection in secure cooperative wireless communication [40], electrical power allocation management [41], production planning [42], in selection of renovation actions [43], waste management [44], formulation and solution method for tour conducting and optimization of content delivery networks [45].

Even though there is a lot of effort from researchers to develop an efficient and consistent solution such a method does not exists. The knapsack problem is NP hard [10, 18, 19, 26, 35, 36] and an optimal solution is very difficult to obtain. For example in [10] a parallel algorithm for solving the NP-complete Knapsack Problem was proposed. NP complete is the most difficult subset of the NP hard problems. In [18] it is pointed out that the knapsack problem is an NP-hard optimization problem with so many diverse applications in industrial and management engineering, however, computational complexities associated with this problem still remain in the knapsack problem. In [19] it is also made very clear that the knapsack problem is a well-known NP-hard combinatorial optimisation problem, with many practical applications. Even up to now, approximation methods are still being developed [13, 25, 26, 35, 45] for this problem. The reason for using heuristics is that there are no efficient consistent exact methods for the knapsack problem. In [35] it is clarified that because of the high computational complexity of knapsack problem, three heuristic approaches are proposed. The paper [26] is a recent heuristic which shows that exact efficient approaches for this knapsack problem are not available.

The approximated solution for the knapsack problem is easy to obtain and good for quick decisions but the difference between the approximated solution and the exact solution may be in millions of dollars for large projects such the UN humanitarian projects and the US military operations. There is a need for exact methods for the knapsack problem.

3. The aim and objectives of the study

The aim of the study is to reformulate the knapsack problem given in so that it is easier to solve by branch and bound algorithm. To achieve the set aim the following tasks have been solved:

– to determine the objective bound Z_0^B;

– to use the objective bound to generate the variable sum limits $\ell_1, \ell_2, ..., \ell_k$ and additional constraints;

– to construct the *k* initial parallel sub-problems;

– to illustrate by an example how to reformulate a knapsack;

– to give classes ad examples of difficult knapsack problems.

4. The knapsack linear integer problem

4. 1. General form of knapsack problem

The knapsack linear integer problem is a special case of the general integer problem. Even though this integer problem has only one constraint, it is believed to be NP complete and very difficult to solve.

Minimize $Z = c_1x_1 + c_2x_2 + ... + c_nx_n$. Such that:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \geq b, \qquad (1)$$

where x_j is integer.

4. 2. Totally unimodular transportation matrix

The constraints of any linear integer problem can be expressed as (2).

$$AX = B, \qquad (2)$$

where A is the transportation coefficient matrix.

Theorem 1: Matrix *A* is totally unimodular if the determinant of each square submatrix of is 0, –1 or +1.

Theorem 2: If matrix A is totally unimodular, then every vertex solution of (2) is integral

Proof of 1&2. Note that every column of A has exactly two 1's, thus any column of A_k has either:

1) two 1's;

2) only one 1;

3) exactly No. 1.

If A_k contains a column that has No. 1, then clearly $Det[A_k] = 0$ and done for (*i*). Thus now assume that every column of A_k contains at least one 1. There are two cases that must be considered here. The first case is where every column of A_k contains two 1's. Then one of the 1's must come from the source rows and the other one must come from the destination rows. Hence subtracting the sum of all source rows from the sum of all destination rows in A_k will give the zero vector. Thus the row vectors of A_k are linearly dependent. Hence $Det[A_k] = 0$. What is now left is to consider the case where at least one column of A_k contains exactly one 1. By expanding A_k with respect to this column, let's have $Det[A_k] = \pm Det[A_{k-1}]$ where the sign depends on the indices of that particular 1. Now the theorem is proved by repeating the argument to matrix A_{k-1}. Therefore the matrix A_k is totally unimodular. More on unimodular matrices can be found in [46]. The variable sum inequalities constructed in this chapter have zeros and ones as the only coefficients. Making the coefficient of every linear integer problem unimodular is a very difficult task. In this paper let's rely on the strategy of introducing new constraints to the knapsack problem with only zeros (0s) and ones (1s) as coefficients. This does not make the knapsack problem unimodular but makes the problem easier to solve than the original form.

4. 3. Branch and Bound Algorithm

The branch and bound algorithm in general relies on the usual strategy of first relaxing the integer problem into a linear programing (LP) model. If the linear programming optimal solution is integer then, the optimal solution to the integer problem is available. If the linear programming optimal solution is not integer, then a variable with a fractional value is selected to create two sub-problems such that part of the feasible region is discarded without eliminating any of the feasible integer solutions. The process is repeated on all variables with fractional values until an integer solution is found. The worst case complexity of the branch and bound algorithm on knapsack linear integer models is NP Complete. The number of sub-problems can easily reach levels that are not manageable.

4. 4. Variable sum equality

A constraint of the form $x_1 + x_2 + ... + x_n = \ell$, where ℓ is an integer, is called a variable sum equality. Let's note the coefficients are only ones and this equality is not new and has been used as clique inequality in the general integer programming. Variable sum equalities can be generated for (1). Let $x_j^0 = SINT \geq \left(b / c_j\right)$ where *SINT* stands for the smallest integer. The objective bound Z_0^B can be found as (3) and can be expressed as (4).

$$Z_0^B = \min\left[c_1x_1^0, c_2x_2^0, ..., c_nx_n^0\right]. \qquad (3)$$

$$c_1x_1 + c_2x_2 + ... + c_nx_n \leq Z_0^B. \qquad (4)$$

The variable sum bounds $\left(\ell_1 \,\&\, \ell_k\right)$ which are integers can now be determined once the objective bound is known. These two integral bounds satisfy (5).

$$\ell_1 \leq x_1 + x_2 + ... + x_n \leq \ell_k. \qquad (5)$$

The two variable sum bounds may be found by solving the following two linear programming models (6), (7).

Maximize $\ell_k = x_1 + x_2 + ... + x_n$. Such that:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \geq b,$$

$$c_1x_1 + c_2x_2 + ... + c_nx_n \leq Z_0^B, \qquad (6)$$

where x_j is integer.

Minimize $\ell_1 = x_1 + x_2 + ... + x_n$. Such that:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \geq b,$$

$$c_1x_1 + c_2x_2 + ... + c_nx_n \leq Z_0^B, \qquad (7)$$

where x_j is integer.

The variable sum equality was used recently in [17] to improve the optimality verification process. If there are parallel processors then these can be solved at the same time, otherwise these can be solved as a combined problem given in (8).

Let's maximize $\ell_2 - \ell_1$. Such that:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \geq b,$$

$$c_1x_1 + c_2x_2 + ... + c_nx_n \leq Z_0^B,$$

$$\ell_k = x_1 + x_2 + ... + x_n,$$

$$a_1y_1 + a_2y_2 + ... + a_ny_n \geq b,$$

$$c_1y_1 + c_2y_2 + ... + c_ny_n \leq Z_0^B,$$

$$\ell_1 = y_1 + y_2 + ... + y_n, \qquad (8)$$

where $x_j, y_j \geq 0$ are the unknown variables.

In this case the *y* variables are used for the second problem.

4. 5. Initial branches

Once the variable sum bounds have been determined then the variable sum constraints can now be constructed as given in (9). From $\ell_1 \leq x_1 + x_2 + \ldots + x_n \leq \ell_k$, let's have k equality constraints, i. e.

$$x_1 + x_2 + x_3 + \ldots + x_n = \ell_1,$$

$$x_1 + x_2 + x_3 + \ldots + x_n = \ell_2,$$

$$x_1 + x_2 + x_3 + \ldots + x_n = \ell_3,$$

$$\ldots$$

$$x_1 + x_2 + x_3 + \ldots + x_n = \ell_k. \qquad (9)$$

Each variable sum equality is an initial branch for the branch and bound procedure which imply that the knapsack problem has k initial branches to be explored. The branches are shown in Fig. 1. he k initial branches of the proposed in and illustrated in Fig. 1, can be explored independently thus allowing the use of the much needed parallel processors.

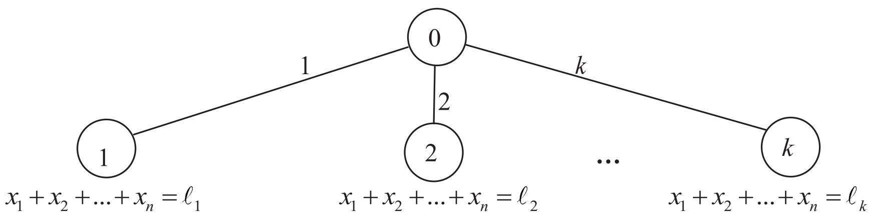
0
1
2
k
1
2
...
k
$x_1 + x_2 + \ldots + x_n = \ell_1$
$x_1 + x_2 + \ldots + x_n = \ell_2$
$x_1 + x_2 + \ldots + x_n = \ell_k$

Fig. 1. Initial branches of a knapsack problem

4. 6. Two additional constraints

Two additional binding constraints can be constructed and added to the original knapsack problem so that the complexity is reduced further. If the variable giving the objective bound is x_j then an additional variable x_{n+1} can be introduced such that.

$$x_j + x_{n+1} = \ell_i, \; x_{n+1} = x_1 + x_2 + \ldots + x_n, \qquad (10)$$

i. e.

$$x_1 + x_2 + \ldots + x_n - x_{n+1} = 0. \qquad (11)$$

The variable x_j is excluded in the sum of variables (11). Let's note that (11) is obtained by rearranging the variables and that the two constraints (10) and (11) are made up of only (0 s) and (±1 s) as the coefficients. The addition of these two constraints to each branch will significantly reduce the complexity of the problem.

5. Reformulation procedure for the knapsack linear integer problem

5. 1. Numerical illustration

Let's minimize:

$$Z = 162x_1 + 38x_2 + 26x_3 + 301x_4 + 87x_5 + 5x_6 + 137x_7.$$

Such that:

$$165x_1 + 45x_2 + 33x_3 + 279x_4 + 69x_5 + \\ + 6x_6 + 122x_7 \geq 18773, \qquad (12)$$

where $x_j \geq 0$ and integer $\forall j$.

The branch and bound algorithm takes 1351 sub-problems to verify the optimal solution: $x_3 = 568$, $x_6 = 5$, $x_6 = 5$, $x_1 = x_2 = x_4 = x_5 = x_7 = 0 \,\&\, Z = 14{,}793$. This is a very small problem and the 1,351 sub-problems used to verify the optimal solution is too much.

There is definitely a need to preprocess the knapsack linear problem before solving it by the branch and bound method. In this paper there are variable sum constraints and additional constraints and add them to the original problem and then solve,

5. 2. Reformulation procedure

Given any knapsack linear integer problem of the form.

Let's minimize $Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$.

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b,$$

where x_j is integer.

An objective bound (Z_0^B), variable sum limits ($\ell_1, \ell_2, \ldots, \ell_k$) and the two additional constraints as $x_j + x_{n+1} = \ell_i$ and $x_{n+1} = x_1 + x_2 + \ldots + x_n$ can be determined. The initial k sub-problems generated are:

– *Initial sub-problem 1.*

Let's minimize:

$$Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n.$$

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b,$$

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq Z_0^B,$$

$$x_1 + x_2 + \ldots + x_n = \ell_1,$$

$$x_j + x_{n+1} = \ell_1,$$

$$x_1 + x_2 + \ldots + x_n - x_{n+1} = 0.$$

– *Initial sub-problem 2.*

Let's minimize $Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$. Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b,$$

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq Z_0^B,$$

$$x_1 + x_2 + \ldots + x_n = \ell_2,$$

$$x_j + x_{n+1} = \ell_2,$$

$$x_1 + x_2 + \ldots + x_n - x_{n+1} = 0.$$

– *Initial sub-problem k.*

Let's minimize $Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$. Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b,$$

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq Z_0^B,$$

$$x_1 + x_2 + \ldots + x_n = \ell_k,$$

$$x_j + x_{n+1} = \ell_k,$$

$$x_1 + x_2 + \ldots + x_n - x_{n+1} = 0.$$

The *k* initial sub-problems can be solved independently and the optimal solution is the best solution (in terms of objective value) from the *k* sub-problems.

5. 3. Algorithm

In other words the knapsack linear integer problem is solved using the following steps.

Step 1: Determine the objective bound Z_0^B.

Step 2: Use the objective bound to generate the variable sum limits $\ell_1, \ell_2, ..., \ell_k$ and additional constraints.

Step 3: Construct the *k* initial sub-problems.

Step 4: Solve the *k* sub-problems to obtain the optimal solution as the best solution from the *k* sub-problems in terms of the objective value.

5. 4. Using the numerical illustration from 5. 1

Let's minimize:

$$Z = 162x_1 + 38x_2 + 26x_3 + \\ +301x_4 + 87x_5 + 5x_6 + 137x_7.$$

Such that:

$$165x_1 + 45x_2 + 33x_3 + 279x_4 \\ + + 69x_5 + 6x_6 + 122x_7 \geq 18{,}773,$$

where $x_j \geq 0$ and integer $\forall j$.

Step 1.

$$x_j^0 = 289\ 418\ 569\ 68\ 273\ 3129\ 154.$$

It should be a solid formula:

$$Z_0^B = \min\begin{bmatrix} 46{,}818, 15{,}884, 14{,}794, \\ 20{,}468, 23{,}751, 15{,}645, 21{,}098 \end{bmatrix} = 14{,}794. \quad (13)$$

Step 2.

Let's maximize $\ell_{10} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$.

Such that:

$$165x_1 + 45x_2 + 33x_3 + 279x_4 + \\ +69x_5 + 6x_6 + 122x_7 \geq 18{,}773,$$

$$162x_1 + 38x_2 + 26x_3 + 301x_4 + \\ +87x_5 + 5x_6 + 137x_7 \leq 14{,}794,$$

$$x_j \geq 0.$$

$$\ell_{10} \leq 578.33 \therefore \ell_{10} = 578.$$

Let's minimize $\ell_1 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$.

Such that:

$$165x_1 + 45x_2 + 33x_3 + 279x_4 + \\ +69x_5 + 6x_6 + 122x_7 \geq 18{,}773,$$

$$162x_1 + 38x_2 + 26x_3 + 301x_4 + \\ +87x_5 + 5x_6 + 137x_7 \leq 14{,}794,$$

$$x_j \geq 0.$$

$$\ell_1 \leq 568.43 \therefore \ell_1 = 569.$$

i. e.

$$569 \leq x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 578, \quad (14)$$

i. e.

$$\ell_1 = 569,\ \ \ell_2 = 570,\ \ \ell_3 = 571,$$

$$\ell_4 = 572,\ \ \ell_5 = 573,\ \ \ell_6 = 574,$$

$$\ell_7 = 575,\ \ \ell_8 = 576,\ \ \ell_9 = 577\ \&\ \ell_{10} = 578. \quad (15)$$

The general two additional constraints are:

$$x_3 + x_8 = \ell_i, \forall i = 1,2,..9. \quad (16)$$

$$x_8 = x_1 + x_2 + x_4 + x_5 + x_6 + x_7. \quad (17)$$

Step 3.

The 9 initial sub-problems are:

1) Sub-problem 1.

Let's minimize:

$$Z = 162x_1 + 38x_2 + 26x_3 + 301x_4 + 87x_5 + 5x_6 + 137x_7.$$

Such that:

$$165x_1 + 45x_2 + 33x_3 + 279x_4 + \\ +69x_5 + 6x_6 + 122x_7 \geq 18{,}773,$$

$$162x_1 + 38x_2 + 26x_3 + 301x_4 + \\ +87x_5 + 5x_6 + 137x_7 \leq 14{,}794, \quad (18)$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 569,$$

$$x_3 + x_8 = 569,$$

$$x_8 = x_1 + x_2 + x_4 + x_5 + x_6 + x_7.$$

2) Sub-problem 2.

Let's minimize:

$$Z = 162x_1 + 38x_2 + 26x_3 + 301x_4 + 87x_5 + 5x_6 + 137x_7.$$

Such that:

$$165x_1 + 45x_2 + 33x_3 + 279x_4 + \\ +69x_5 + 6x_6 + 122x_7 \geq 18{,}773,$$

$$162x_1 + 38x_2 + 26x_3 + 301x_4 + \\ +87x_5 + 5x_6 + 137x_7 \leq 14{,}794, \quad (19)$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 570,$$

$$x_3 + x_8 = 570,$$

$$x_8 = x_1 + x_2 + x_4 + x_5 + x_6 + x_7.$$

3) Sub-problem 3.

Let's minimize:

$$Z = 162x_1 + 38x_2 + 26x_3 + 301x_4 + 87x_5 + 5x_6 + 137x_7.$$

Such that:

$$\begin{aligned}&165x_1+45x_2+33x_3+279x_4+\\&+69x_5+6x_6+122x_7\geq 18{,}773,\\&162x_1+38x_2+26x_3+301x_4+\\&+87x_5+5x_6+137x_7\leq 14{,}794,\\&x_1+x_2+x_3+x_4+x_5+x_6+x_7=571,\\&x_3+x_8=571,\\&x_8=x_1+x_2+x_4+x_5+x_6+x_7.\end{aligned} \qquad (20)$$

4) Sub-problem 4.
Let's minimize:

$$Z=162x_1+38x_2+26x_3+301x_4+87x_5+5x_6+137x_7.$$

Such that:

$$\begin{aligned}&165x_1+45x_2+33x_3+279x_4+\\&+69x_5+6x_6+122x_7\geq 18{,}773,\\&162x_1+38x_2+26x_3+301x_4+\\&+87x_5+5x_6+137x_7\leq 14{,}794,\\&x_1+x_2+x_3+x_4+x_5+x_6+x_7=572,\\&x_3+x_8=572,\\&x_8=x_1+x_2+x_4+x_5+x_6+x_7.\end{aligned} \qquad (21)$$

5) Sub-problem 5.
Let's minimize:

$$Z=162x_1+38x_2+26x_3+301x_4+87x_5+5x_6+137x_7.$$

Such that:

$$\begin{aligned}&165x_1+45x_2+33x_3+279x_4+\\&+69x_5+6x_6+122x_7\geq 18{,}773,\\&162x_1+38x_2+26x_3+301x_4+\\&+87x_5+5x_6+137x_7\leq 14{,}794,\\&x_1+x_2+x_3+x_4+x_5+x_6+x_7=573,\\&x_3+x_8=573,\\&x_8=x_1+x_2+x_4+x_5+x_6+x_7.\end{aligned} \qquad (22)$$

6) Sub-problem 6.
Let's minimize:

$$Z=162x_1+38x_2+26x_3+301x_4+87x_5+5x_6+137x_7.$$

Such that:

$$\begin{aligned}&165x_1+45x_2+33x_3+279x_4+\\&+69x_5+6x_6+122x_7\geq 18{,}773,\\&162x_1+38x_2+26x_3+301x_4+\\&+87x_5+5x_6+137x_7\leq 14{,}794,\\&x_1+x_2+x_3+x_4+x_5+x_6+x_7=575,\\&x_3+x_8=575,\\&x_8=x_1+x_2+x_4+x_5+x_6+x_7.\end{aligned} \qquad (23)$$

7) Sub-problem 7.
Let's minimize:

$$Z=162x_1+38x_2+26x_3+301x_4+87x_5+5x_6+137x_7.$$

Such that:

$$\begin{aligned}&165x_1+45x_2+33x_3+279x_4+\\&+69x_5+6x_6+122x_7\geq 18{,}773,\\&162x_1+38x_2+26x_3+301x_4+\\&+87x_5+5x_6+137x_7\leq 14{,}794,\\&x_1+x_2+x_3+x_4+x_5+x_6+x_7=576,\\&x_3+x_8=576,\\&x_8=x_1+x_2+x_4+x_5+x_6+x_7.\end{aligned} \qquad (24)$$

8) Sub-problem 8.
Let's minimize:

$$Z=162x_1+38x_2+26x_3+301x_4+87x_5+5x_6+137x_7.$$

Such that:

$$\begin{aligned}&165x_1+45x_2+33x_3+279x_4+\\&+69x_5+6x_6+122x_7\geq 18{,}773,\\&162x_1+38x_2+26x_3+301x_4+\\&+87x_5+5x_6+137x_7\leq 14{,}794,\\&x_1+x_2+x_3+x_4+x_5+x_6+x_7=577,\\&x_3+x_8=577,\\&x_8=x_1+x_2+x_4+x_5+x_6+x_7.\end{aligned} \qquad (25)$$

9) Sub-problem 9.
Let's minimize:

$$Z=162x_1+38x_2+26x_3+301x_4+87x_5+5x_6+137x_7.$$

Such that:

$$165x_1+45x_2+33x_3+279x_4+$$
$$+69x_5+6x_6+122x_7\geq 18{,}773,$$

$$162x_1+38x_2+26x_3+301x_4+$$
$$+87x_5+5x_6+137x_7\leq 14{,}794, \tag{26}$$

$$x_1+x_2+x_3+x_4+x_5+x_6+x_7=578,$$

$$x_3+x_8=578,$$

$$x_8=x_1+x_2+x_4+x_5+x_6+x_7.$$

The initial branches and their corresponding number of sub-problems are given in Table 1.

Table 1

No. of sub-problems for each branch

Sub-problem	Number of sub-problems	Solution	Optimal solution
1	7	$Z_o=14794,\ \ x_3=569,$ $x_1=x_2=x_4=x_5=x_6=x_7=0$	
2	7	Infeasible	
3	7	Infeasible	
4	11	Infeasible	
5	5	$Z_o=14793,$ $x_3=568,\ \ x_6=5,$ $x_1=x_2=x_4=x_5=x_7=0$	Best & optimal
6	5	Infeasible	
7	5	Infeasible	
8	5	Infeasible	
9	7	Infeasible	
10	3	Infeasible	

The automated branch and bound algorithm takes only 5 sub-problems to verify the optimal solution: $x_3=568$, $x_6=5$, $x_1=x_2=x_4=x_5=x_7=0\ \&\ Z=14793$, in the initial parallel problem 5.

6. Some difficult classes of knapsack problems

6. 1. Knapsack binary linear problems with bizarre behaviour

Let's maximize:

$$Z=\sum_{i=1}^{n-1}x_i$$

or Minimize x_n.

Such that:

$$2\sum_{i=1}^{n-1}x_i\pm x_n=n-1, \tag{20}$$

where $x_j=0$ or $1\ \forall j$ and n is even.

The behaviour of the standard branch and bound method for n=4, 6, 8, 16, ..., is given in Table 2. The number of sub-problems increases exponentially as n increases.

Table 2

Complexity of the problem as n increases

Value of n in model	Number of sub-problems created by the branch and bound approach to reach the optimum solution
4	11
6	39
8	139
16	25,739
32	Number of sub-problems exceeds 30,000
...	...

This shows that the branch and bound on its own is not a very good approach.

6. 2. Second class of bizarre knapsack problems

This is a modification of Class 7. 1. Class 7. 1 and the general form is given in (21).

Let's maximize:

$$Z=\sum_{j=1}^{n-1}x_j$$

or Minimize x_n.

Such that:

$$2\sum_{j=1}^{n-1}x_j\pm\kappa x_n=n-1, \tag{21}$$

where $x_j=0$ or $1\ \forall j,\ 3\leq\kappa\leq n-1,\ \kappa$ is odd and n is even.

The bizarre behaviour of the branch and bound method is given in Table 3.

Table 3

Complexity of the problem as n increases

Value of n in model	Number of sub-problems created by the branch and bound approach to reach the optimum solution
4	$\kappa=3$, sub-problems=17
6	$\kappa=3$, sub-problems=59
	$\kappa=5$, sub-problems=59
8	$\kappa=3$, sub-problems=209
	$\kappa=5$, sub-problems=209
	$\kappa=7$, sub-problems=209
16	Sub-problems exceeded 30,000

These are small problems and the branch is not expected to struggle to solve these problems.

6. 3. Third class of knapsack problem pure integer case

Changing of variables from binary to pure integer in any difficult knapsack problem automatically increases the complexity of the problem.

Let's maximize:

$$Z=\sum_{j=1}^{n-1}x_j$$

or Minimize x_n.

Such that:

$$2\sum_{j=1}^{n-1} x_j + \kappa x_n = n-1, \tag{22}$$

where $x_j \geq 0,$ integer $\forall j,$ $1 \leq \kappa \leq n-1,$ κ is odd and n is even.

The behaviour of the branch and bound method for n=4, 6, 8 and 16 for this class of difficult problems is given in Table 4.

Table 4

Complexity of the problem as n increases

Value of n in model	Number of sub-problems created by the branch and bound approach to reach the optimum solution
4	$\kappa = 3,$ sub-problems=23
6	$\kappa = 3,$ sub-problems=129
	$\kappa = 5,$ sub-problems=129
8	$\kappa = 3,$ sub-problems=755
	$\kappa = 5,$ sub-problems=755
	$\kappa = 7,$ sub-problems=755
16	Number of the sub-problems exceed 30,000

Changing from binary to general integer means expanding the problem. The number of sub-problems increases and this is expected.

6. 4. Fourth class of hard knapsack problems

Let's maximize:

$$Z = \sum_{j=1}^{n-1} x_j$$

or Minimize x_n.

Such that:

$$2\sum_{j=1}^{n-1} x_j + \kappa x_n = \lambda, \tag{23}$$

where $x_j \geq 0,$ integer $\forall j,$ $2(n-1)+k = \lambda,$ $\kappa, \lambda \geq 0$ are odd and n is even.

The standard branch and bound method can't solve most of these for large values of κ. For example a knapsack problem with the parameters, $n=4,$ $k=91,$ $\lambda = 97,$ explodes to an unmanageable number of sub-problems.

Let's minimize $Z = x_n.$ Such that:

$$2x_1 + 2x_2 + 2x_3 + 91x_4 = 97, \tag{24}$$

where $x_j \geq 0,$ integer $\forall j.$

The branch and bound method requires 7449 sub-problems to verify the optimal solution. For large values of λ the knapsack problems are very difficult to solve by the standard branch and bound algorithm on its own. These numerical illustrations and more on complexity of knapsack problems and other linear integer models are given in [9].

7. Discussion of experimental results

The knapsack problem has been reformulated and a numerical illustration is used to show the reformulation process. The numerical illustration is given in Section 5. The branch and bound on its own took 1351 sub-problems to verify optimality. The same knapsack problem is reformulated into 10 parallel problems which can be solved independently as given in Section 5. 4. The reformulated same knapsack problem is solved by the branch and bound algorithm. The numbers of iterations required to verify optimality ranges from 3 to 11 for the parallel problems as given in Table 1. Reducing complexity from 1,351 to the worst case of 11 is a very significant improvement. The branch and bound algorithm is a general purpose algorithm for solving the general linear integer problem. Unfortunately this approach on its own has serious weaknesses as presented in Section 6 from Tables 2 to 4.

The reformulated knapsack can be identified by the following features. The new knapsack problem is split into several independent parallel problems. The number of constraints increases from 1 to 5 for each parallel. The 4 new constraints for each parallel problem include an objective bound. Splitting the problem into many parallel problems, increasing the number of constraints from only 1 to 4 and increasing the number of variables by 2 for each split problem are the weaknesses of the proposed approach. The most important feature of the new problem is that it is easier to solve by the branch and bound algorithm than the original single constraint knapsack form.

There is a need to compare the proposed approach with other methods. Again this is a limitation and a shortcoming for this study. What seems to be an obvious weakness is that the reformulation splits the single problem into many but easier problems to solve. The challenge of splitting the problem into parallel independent sub-problem can be alleviated by use of parallel computer processors. There is a need to further reduce the numbers of branch and branch bound iterations needed to solve each sub-problem. The main challenge with this is that the complexity of the general integer problem increases with an increase in the number of variables.

8. Conclusions

1. Determining an objective bound $\left(Z_0^B\right)$ to the knapsack problem. An objective bound which is the initial upper limit to the objective value of the problem. In this study all the n given variables in the original knapsack problem were used in determining the objective bound.

2. Once the objective bound was determined it became easy to generate the k variable sum limits $\ell_1, \ell_2, ..., \ell_k$ and the 2 additional constraints. To do this let's only calculate ℓ_1 and ℓ_k and the rest generated as all the integers between ℓ_1 and ℓ_k. The first additional constraint was easily generated from the objective bound and objective row and the other two constraints were generated from the variable sum limits.

3. The k parallel initial sub-problems where constructed from the k variable limits $\ell_1, \ell_2, ..., \ell_k$ and 3 additional constraints. Even though the original knapsack problem looked simpler than the each of the k parallel sub-problems the truth is that the reformulated parallel problems were easier to solve than the original problem as attested to by the numerical illustration. The available computing power which is in terms parallel processing can be taken advantage of.

4. The numerical illustration was shown in Section 5. Reformulation is the way for a knapsack problem given I this paper.

5. Classes of difficult problems were presented in this study. There is need for more research on knapsack problems as shown from the various applications.

References

1. Al-Rabeeah, M., Munapo, E., Al-Hasani, A., Kumar, S., Eberhard, A. (2019). Computational Enhancement in the Application of the Branch and Bound Method for Linear Integer Programs and Related Models. International Journal of Mathematical, Engineering and Management Sciences, 4 (5), 1140–1153. doi: https://doi.org/10.33889/ijmems.2019.4.5-090
2. Beasley, J. E. (1996). Advances in Linear and Integer Programming. Oxford University Press.
3. Kumar, S., Munapo, E., Jones, B. C. (2007). An Integer Equation Controlled Descending Path to a Protean Pure Integer Program. Indian Journal of Mathematics, 49 (2), 211–237.
4. Taha, H. A. (2016). Operations Research: An Introduction. Pearson Education Limited, 848.
5. Winston, W. L. (2003). Operations Research Applications and Algorithms. Duxbury Press, 1440.
6. Land, A. H., Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. Econometrica, 28 (3), 497. doi: https://doi.org/10.2307/1910129
7. Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. The Computer Journal, 8 (3), 250–255. doi: https://doi.org/10.1093/comjnl/8.3.250
8. Munapo, E. (2020). Improving the Optimality Verification and the Parallel Processing of the General Knapsack Linear Integer Problem. Research Advancements in Smart Technology. Optimization, and Renewable Energy. Chap. 3.
9. Munapo, E., Kumar, S. (2016). Knapsack constraint reformulation: A new approach that significantly reduces the number of sub-problems in the branch and bound algorithm. Cogent Mathematics, 3 (1). doi: https://doi.org/10.1080/23311835.2016.1162372
10. Vasilchikov, V. V. (2018). On a Recursive-Parallel Algorithm for Solving the Knapsack Problem. Automatic Control and Computer Sciences, 52 (7), 810–816. doi: https://doi.org/10.3103/s014641161807026x
11. Bhattacharjee, K. K., Sarmah, S. P. (2014). Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. Applied Soft Computing, 19, 252–263. doi: https://doi.org/10.1016/j.asoc.2014.02.010
12. Simon, J., Apte, A., Regnier, E. (2017). An application of the multiple knapsack problem: The self-sufficient marine. European Journal of Operational Research, 256 (3), 868–876. doi: https://doi.org/10.1016/j.ejor.2016.06.049
13. Lahyani, R., Chebil, K., Khemakhem, M., Coelho, L. C. (2019). Matheuristics for solving the Multiple Knapsack Problem with Setup. Computers & Industrial Engineering, 129, 76–89. doi: https://doi.org/10.1016/j.cie.2019.01.010
14. Martello, S., Monaci, M. (2020). Algorithmic approaches to the multiple knapsack assignment problem. Omega, 90, 102004. doi: https://doi.org/10.1016/j.omega.2018.11.013
15. Bienstock, D., Faenza, Y., Malinović, I., Mastrolilli, M., Svensson, O., Zuckerberg, M. (2020). On inequalities with bounded coefficients and pitch for the min knapsack polytope. Discrete Optimization, 100567. doi: https://doi.org/10.1016/j.disopt.2020.100567
16. Fampa, M., Lubke, D., Wang, F., Wolkowicz, H. (2020). Parametric convex quadratic relaxation of the quadratic knapsack problem. European Journal of Operational Research, 281 (1), 36–49. doi: https://doi.org/10.1016/j.ejor.2019.08.027
17. Dahmani, I., Hifi, M., Saadi, T., Yousef, L. (2020). A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict Graphs. Expert Systems with Applications, 148, 113224. doi: https://doi.org/10.1016/j.eswa.2020.113224
18. Wu, Z., Jiang, B., Karimi, H. R. (2020). A logarithmic descent direction algorithm for the quadratic knapsack problem. Applied Mathematics and Computation, 369, 124854. doi: https://doi.org/10.1016/j.amc.2019.124854
19. Djeumou Fomeni, F., Kaparis, K., Letchford, A. N. (2020). A cut-and-branch algorithm for the Quadratic Knapsack Problem. Discrete Optimization, 100579. doi: https://doi.org/10.1016/j.disopt.2020.100579
20. Bandyopadhyay, S., Maulik, U., Chakraborty, R. (2013). Incorporating ϵ-dominance in AMOSA: Application to multiobjective 0/1 knapsack problem and clustering gene expression data. Applied Soft Computing, 13 (5), 2405–2411. doi: https://doi.org/10.1016/j.asoc.2012.11.050
21. Zouache, D., Moussaoui, A., Ben Abdelaziz, F. (2018). A cooperative swarm intelligence algorithm for multi-objective discrete optimization with application to the knapsack problem. European Journal of Operational Research, 264 (1), 74–88. doi: https://doi.org/10.1016/j.ejor.2017.06.058
22. Abdel-Basset, M., El-Shahat, D., Faris, H., Mirjalili, S. (2019). A binary multi-verse optimizer for 0-1 multidimensional knapsack problems with application in interactive multimedia systems. Computers & Industrial Engineering, 132, 187–206. doi: https://doi.org/10.1016/j.cie.2019.04.025
23. Arin, A., Rabadi, G. (2016). Local search versus Path Relinking in metaheuristics: Redesigning Meta-RaPS with application to the multidimensional knapsack problem. Applied Soft Computing, 46, 317–327. doi: https://doi.org/10.1016/j.asoc.2016.05.016
24. Wang, L., Yang, R., Ni, H., Ye, W., Fei, M., Pardalos, P. M. (2015). A human learning optimization algorithm and its application to multi-dimensional knapsack problems. Applied Soft Computing, 34, 736–743. doi: https://doi.org/10.1016/j.asoc.2015.06.004
25. Lai, X., Hao, J.-K., Fu, Z.-H., Yue, D. (2020). Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem. Expert Systems with Applications, 149, 113310. doi: https://doi.org/10.1016/j.eswa.2020.113310
26. Wei, Z., Hao, J.-K. (2019). Iterated two-phase local search for the Set-Union Knapsack Problem. Future Generation Computer Systems, 101, 1005–1017. doi: https://doi.org/10.1016/j.future.2019.07.062
27. Brunetta, L., Conforti, M., Rinaldi, G. (1997). A branch-and-cut algorithm for the equicut problem. Mathematical Programming, 78 (2), 243–263. doi: https://doi.org/10.1007/bf02614373
28. Mitchell, J. E. (2001). Branch and cut algorithms for integer programming. Encyclopedia of Optimization. Kluwer Academic Publishers.

29. Padberg, M., Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. SIAM Review, 33 (1), 60–100. doi: https://doi.org/10.1137/1033004
30. Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research, 46 (3), 316–329. doi: https://doi.org/10.1287/opre.46.3.316
31. Savelsbergh, M. (1997). A Branch-and-Price Algorithm for the Generalized Assignment Problem. Operations Research, 45 (6), 831–841. doi: https://doi.org/10.1287/opre.45.6.831
32. Barnhart, C., Hane, C. A., Vance, P. H. (2000). Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. Operations Research, 48 (2), 318–326. doi: https://doi.org/10.1287/opre.48.2.318.12378
33. Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. de, Reis, M., Uchoa, E., Werneck, R. F. (2005). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. Mathematical Programming, 106 (3), 491–511. doi: https://doi.org/10.1007/s10107-005-0644-x
34. Ladányi, L., Ralphs, T. K., Trotter, L. E. (2001). Branch, Cut, and Price: Sequential and Parallel. Computational Combinatorial Optimization, 223–260. doi: https://doi.org/10.1007/3-540-45586-8_6
35. Meng, F., Chu, D., Li, K., Zhou, X. (2019). Multiple-class multidimensional knapsack optimisation problem and its solution approaches. Knowledge-Based Systems, 166, 1–17. doi: https://doi.org/10.1016/j.knosys.2018.11.006
36. Gurski, F., Rehs, C., Rethmann, J. (2019). Knapsack problems: A parameterized point of view. Theoretical Computer Science, 775, 93–108. doi: https://doi.org/10.1016/j.tcs.2018.12.019
37. Khonji, M., Karapetyan, A., Elbassioni, K., Chau, S. C.-K. (2019). Complex-demand scheduling problem with application in smart grid. Theoretical Computer Science, 761, 34–50. doi: https://doi.org/10.1016/j.tcs.2018.08.023
38. Chakraborty, T., Misra, I. S. (2020). A novel three-phase target channel allocation scheme for multi-user Cognitive Radio Networks. Computer Communications, 154, 18–39. doi: https://doi.org/10.1016/j.comcom.2020.02.026
39. Samavati, M., Essam, D., Nehring, M., Sarker, R. (2017). A methodology for the large-scale multi-period precedence-constrained knapsack problem: an application in the mining industry. International Journal of Production Economics, 193, 12–20. doi: https://doi.org/10.1016/j.ijpe.2017.06.025
40. Hassanzadeh, A., Xu, Z., Stoleru, R., Gu, G., Polychronakis, M. (2016). PRIDE: A practical intrusion detection system for resource constrained wireless mesh networks. Computers & Security, 62, 114–132. doi: https://doi.org/10.1016/j.cose.2016.06.007
41. Oprea, S. V., Bâra, A., Ifrim, G. A., Coroianu, L. (2019). Day-ahead electricity consumption optimization algorithms for smart homes. Computers & Industrial Engineering, 135, 382–401. doi: https://doi.org/10.1016/j.cie.2019.06.023
42. Amiri, A. (2020). A Lagrangean based solution algorithm for the knapsack problem with setups. Expert Systems with Applications, 143, 113077. doi: https://doi.org/10.1016/j.eswa.2019.113077
43. Alanne, K. (2004). Selection of renovation actions using multi-criteria «knapsack» model. Automation in Construction, 13 (3), 377–391. doi: https://doi.org/10.1016/j.autcon.2003.12.004
44. Delgado-Antequera, L., Caballero, R., Sánchez-Oro, J., Colmenar, J. M., Martí, R. (2020). Iterated greedy with variable neighborhood search for a multiobjective waste collection problem. Expert Systems with Applications, 145, 113101. doi: https://doi.org/10.1016/j.eswa.2019.113101
45. El Yafrani, M., Ahiod, B. (2017). A local search based approach for solving the Travelling Thief Problem: The pros and cons. Applied Soft Computing, 52, 795–804. doi: https://doi.org/10.1016/j.asoc.2016.09.047
46. Micheli, G., Weger, V. (2019). On Rectangular Unimodular Matrices over the Algebraic Integers. SIAM Journal on Discrete Mathematics, 33 (1), 425–437. doi: https://doi.org/10.1137/18m1177093