Mugla Journal of Science and Technology

# COMPARISON OF CLASSIC AND GREEDY HEURISTIC ALGORITHM RESULTS IN INTEGER PROGRAMMING: KNAPSACK PROBLEMS

Burcu DURMUŞ*, Department of Statistics, Faculty of Science, Mugla Sitki Kocman University, Mugla, Turkey, burcudrmz@windowslive.com
(https://orcid.org/0000-0002-0298-0802)
Öznur İŞÇİ GÜNERİ, Department of Statistics, Faculty of Science, Mugla Sitki Kocman University, Mugla, Turkey, oznur.isci@mu.edu.tr
(https://orcid.org/0000-0003-3677-7121)
Aynur İNCEKIRIK, Department of Econometrics, Faculty of Economics and Administrative Sciences, Manisa Celal Bayar University, Manisa, Turkey, aynur.incekirik@bayar.edu.tr
(https://orcid.org/0000-0002-5029-6036)

Received: 11.10.2018, Accepted: 24.01.2019 Research Article
*Corresponding author DOI: 10.22531/muglajsci.469475

## Abstract

*This paper focus on comparing the differences and similarities between the results obtained from Greedy and classical algorithms for integer linear programming (ILP) problems. For this purpose, the solution of the problems related to different models with the purpose function and constraints has been provided by developing a software (Java Program) which solves the Knapsack problems (KP) with Greedy algorithm. Both the classical algorithm and the results obtained from Greedy algorithm are compared for the problems considered here. In this context, the results obtained from algorithms are found to be the same for small-sized pure and 0-1 binary Knapsack problems. Since packet programs are limited in dimension and number of constraints, it becomes difficult to obtain appropriate results from classical algorithms as the dimension of the problem grows. However, Greedy algorithm gives the appropriate results regardless of the dimension and the number of constraints.*

**Keywords: Integer Programming, Classic Algorithms, Greedy Algorithm, Knapsack Problems**

# TAMSAYILI PROGRAMLAMADA KLASİK VE GREEDY SEZGİSEL ALGORiTMALARININ KARŞILAŞTIRILMASI: SIRT ÇANTASI PROBLEMLERİ

## Özet

*Bu çalışmada, tamsayılı doğrusal programlama (TDP) problemleri için Greedy ve klasik algoritmalardan elde edilen sonuçlar arasındaki fark ve benzerlikler karşılaştırılmıştır. Bu amaçla, Sırt Çantası Problemlerini (SÇP) Greedy algoritmasıyla çözen bir yazılım (Java Program) geliştirerek, amaç fonksiyonu ve kısıtları verilmiş farklı modellere ilişkin problemlere çözüm sağlanmıştır. Dikkate alınan problemler için hem klasik algoritma hem de Greedy algoritmasından elde edilen sonuçlar karşılaştırılmıştır. Bu bağlamda, küçük boyutlu saf ve 0-1 binary sırt çantası problemleri için algoritmalardan elde edilen sonuçlar aynı bulunmuştur. Paket programlar boyut ve kısıt sayısı ile sınırlı olduğundan problemin boyutu büyüdükçe klasik algoritmalar için uygun sonuç elde etmek zorlaşmaktadır. Ancak, Greedy algoritması, boyut ve kısıt sayısını dikkate almaksızın uygun sonuç vermektedir.*

**Anahtar Kelimeler: Tam Sayılı Programlama, Klasik Algoritmalar, Greedy Algoritması, Sırt Çantası Problemleri**

**Cite**
**Durmuş, B., İşçi Güneri, Ö., İncekırık, A. (2019). "Comparison of classic and greedy heuristic algorithm results in integer programming: knapsack problems", Mugla Journal of Science and Technology, 5(1), 34-42**

## 1. Introduction

Although Integer Linear Programming seems to be easier and more understandable than Linear Programming (LP), it is often more time consuming and more complex. Problem solving in ILP requires a true investigation of all possible fields and the number of constraints increases when an integer is added into a problem finding a solution for an integer in ILP may require exponential computing time [1,2,3]. For this reason, various algorithms are needed to facilitate access for the optimal solutions in integer problems. Among these algorithms, the classic ones provide optimal solutions either manually or in a computer environment. However, problems that are too complicated and without algebraically expressible constraints and objective functions could not be solved easily with these algorithms, and solution sometimes cannot be achieved. Such situations need for algorithms called heuristic. Although heuristic algorithms do not always provide the optimal solution, they try to produce the proximal optimal solution. At the beginning of the current study, it was assumed that the classic algorithms give the optimal solution and the heuristic algorithms give the proximal optimal solution. Thus, this study was aimed to provide support for this assumption by comparing the solution results of the two algorithms.

Knapsack problems (KP) is one of the most commonly used problems in integer programming. In this study, KP was used and Greedy algorithm was chosen for the heuristic solutions of KP. A literature search could show that is examined, there are many studies on the solutions of KP with Greedy algorithm (or algorithms based on Greedy terminology). Akçay et al. [4] proposed a new Greedy algorithm for heuristic solution of multi-dimensional KP. Their calculations showed that the solution was reliable. Liu [5] conducted a simulation study on 0-1 KP using Greedy terminology and dynamic programming. Lv et al. [6] designed a degree model by inspiring from Greedy algorithm. Their algorithm has been found to be more effective than the other algorithms. Zhou et al. [7] proposed a new version for Greedy algorithm and compared the results of this new version with the results of five different heuristic algorithms. Their comparisons showed that this new version 0-1 can be an effective alternative to solve KP problem. Bulut et al. [8] tried to optimize the recursive model by adding Greedy approach to the branch and bound algorithm. They found that such an approach can shorten the calculation time.

As seen from the literature, both former and new versions of Greedy algorithm work effectively. However, the number of programs that solve Greedy algorithm are limited. Ready software programs developed for solving ILP problems are limited by the number of variables, constraints and iterations. For this reason, new software development for solving problems is necessary for solving larger-scale problems, since the number of variables and constraints in real life problems is excessive.

Given that Greedy algorithm is used in finding solutions to the problems confronted in daily life or to the problems that cannot be solved, the importance of this algorithm's giving good results could better have understood. For this purpose, a program with Java codes was developed in the application part of this study to compare the solutions of problems with multiple variables. With the developed program, multidimensional problems could not only be solved more economically in terms of both time and calculation cost using Greedy algorithm, but the program also allows the comparison of classic and Greedy heuristic results of multidimensional problems.

It should be noted that integer programming (IP) with integer linear programming (ILP) means the same in the literature.

## 2. Integer Linear Programming (ILP)

It may be desirable to obtain integer results in some problems solved with LP. For example, selection or not selection of the course added to curriculum can be determined by decision variables that take values of 1 and 0, or the number of products to be produced in an enterprise is expressed by integer values. An integer programming method has been developed for such problems where values such as 3.6 computer are not meaningful [1,2]. A general IP model can be expressed as in the Equation (1) below. In this model the constraints can be $\leq$, $\geq$ or $=$ [9].

Objective function:

$$Max\ Z = \sum_{j=1}^{n} c_j\, x_j \quad \text{or} \quad \text{Min}\ Z = \sum_{j=1}^{n} c_j\, x_j \tag{1}$$

Constraints:

$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_j \qquad i = 1,2,\dots,m \ \ and\ \ x_j \geq 0\ and\ integer \tag{2}$$

### 2.1. Types of Integer Programming Problems

The IP problems are divided into three main classes according to the values that their variables will have: pure, 0-1 and mixed integer problems. These models are briefly explained below.

#### 2.1.1. Pure integer problems

Programming problems where all of the variables can only take integer values are called pure integer problems. A general model of pure IP problems can be expressed as follows [10].

Objective function:

$$Max\ Z = \sum_{j=1}^{n} c_j\, x_j \tag{3}$$

Constraints:

$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_j \qquad i = 1,2,\dots,m \ \ and\ \ x_j \geq 0\ and\ integer \tag{4}$$

#### 2.1.2. Mixed integer problems

Programming problems where some of the variables are integers and the others are real values are called mixed integer problems. A general mixed integer problem model is given below.

Objective function:

$$Max\ Z = \sum_{j=1}^{n} c_j\, x_j \qquad (5)$$

Constraints:

$$\sum_{j=1}^{n} a_{ij}\, x_j \le b_j$$
$$i = 1,2,\dots,m \qquad (6)$$
$$x_j \ge 0 \text{ and integer}, \quad j = 1,2,\dots,p$$
$$x_j \ge 0, \quad j = p,\dots,n\ (p \le n)$$

### 2.1.3. Binary (0-1) integer problems

Programming problems where all variables must be 0 or 1 are called 0-1 integer problems.

$$x_j = \begin{cases} 1, \text{if } j.\text{product is selected} \\ 0, \text{if } j.\text{product is not selected} \end{cases} \qquad (7)$$

In the Equation (7), the situations where the product is selected or not selected are shown (j=1, 2,..., n) [11]. In such situations, $x_j$ variables can only take 1 or 0 values. A general 0-1 IP problem model can be expressed as follows.

Objective function:

$$Max\ Z = \sum_{j=1}^{n} c_j\, x_j \qquad (8)$$

Constraints:

$$\sum_{j=1}^{n} a_{ij}\, x_j \le b_j \qquad (9)$$
$$i = 1,2,\dots,m \text{ and } x_j = 0 \text{ or } 1$$

## 3. Solution Methods in Integer Programming

The classic algorithms developed for LP after adding the property of being an integer are insufficient to solve IP problems. For this reason, new methods of solution are needed. Among these methods, heuristic algorithms are divided into two in the literature:

- Classic Heuristic Algorithms
- Meta (Modern) Heuristic Algorithms

Classic heuristic algorithms are examined separately as they perform narrower (local) searches than meta heuristic algorithms.

These solution methods developed for IP can be generally grouped as in Figure 1.

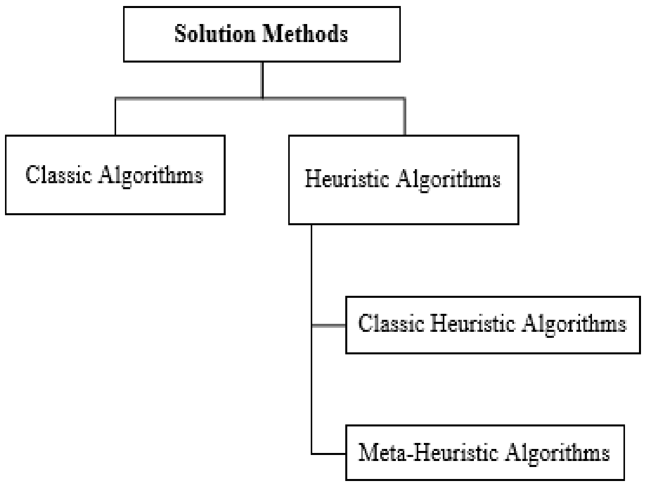


Figure 1. Solution algorithms for IP.

### 3.1. Classic Algorithms

When methods like Simplex developed to solve LP problems are used to solve the problems modeled with business data, they make it possible to reach optimal solutions with modern computers within a few hours. Therefore, LP is widely used [12].

However, this does not hold true for IP problems. IP problems are created by adding integer constraints to LP problems. They therefore contain more constraints. As a consequence of increasing of the constraints, IP problems show exponential growth according to LP problems. Sometimes proper formulations cannot be formed. In this respect, it is not possible to solve the IP problems with the methods developed for LP. As a result of such fundamental differences, it became necessary to develop new algorithms for IP problems.

These algorithms are also known in the literature as 'exact solution algorithms' [2,13]. The classic algorithms most commonly used by researchers are listed in Figure 2.

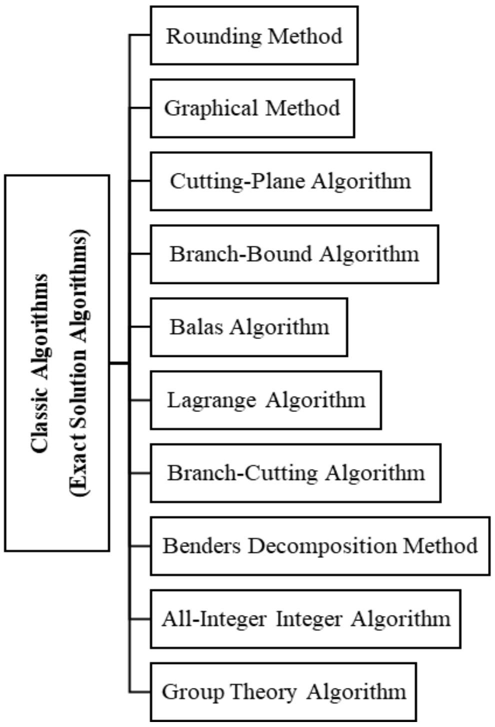


Figure 2. IP classic solution algorithms.

### 3.2. Heuristic Algorithms

Heuristic algorithms are methods that provide optimal solutions close to the optimization problem. Its name is derived from the Greek word "heuristic" meaning "discovering a new method", "solving problem". It entered into our language with the words "sezgisel', 'buluşsal', 'bulgusal'. They became popular as some problems could not be solved with classic algorithms, solutions to these problems were complex, took a long time and were expensive [14]. Heuristic algorithms are also used for quadratic and nonlinear problems. Some of these algorithms are given below;

- Greedy algorithms
- Nearest neighbor algorithm
- Dantzig and Ramser's method
- Saving algorithm
- Sorting algorithm
- Tabu search
- Genetic algorithm
- Simulated annealing
- Ant colony
- Artificial bee colony
- Particle swarm optimization
- Artificial neural networks

They are often inspired by nature, disciplines of science or human behaviors [15]. The accuracy of heuristic algorithms cannot be proved. The main reasons for their frequent preference by researchers are as follows:

- They do not give the optimal solution exactly, but they give near-optimal solutions with small deviations.
- They reach the result with short and simple solutions.
- They are conceptually simple. In small problems, they can yield optimum results.
- Their calculation cost is less [16].
- They can be used for problems that do not have a formulization or an optimal solution.
- They can solve quadratic and nonlinear problems.
- They can be flexed and adapted according to the problem [15].
- They can produce solutions for some problems that package programs do not solve.
- The verifiability of algorithms is ignored [3].

## 4. Greedy Algorithm

Consciously or unconsciously, Greedy algorithm is the most commonly used algorithm in everyday life. The aim is always to keep the utility high. It addresses the problem whose objective function is given with the assumption that one with the highest coefficient would yield the highest profit. Its priority thus is to assign the highest value to this variable. In the case of single constrained problems, it calculates the utility per unit. It tries to fill its capacity by preferring to select the most from the variable yielding the highest benefit [17].

This algorithm uses less computation and fewer algorithms than the other heuristics. For this reason, researchers have remained in dilemma many times about the results. Despite the fact that Greedy heuristic has been disputed on the grounds it yields local or immediate results, its use is still quite common. The main reason for this is that it is quite easy to apply, the computational costs are very low, and it can be applied to all kinds of problems (problems without solution or formulation, complex problems, irreducible problems, problems without a certain model, etc.) [1,3].

A general flow diagram for Greedy is given in Figure 3 [17,18].

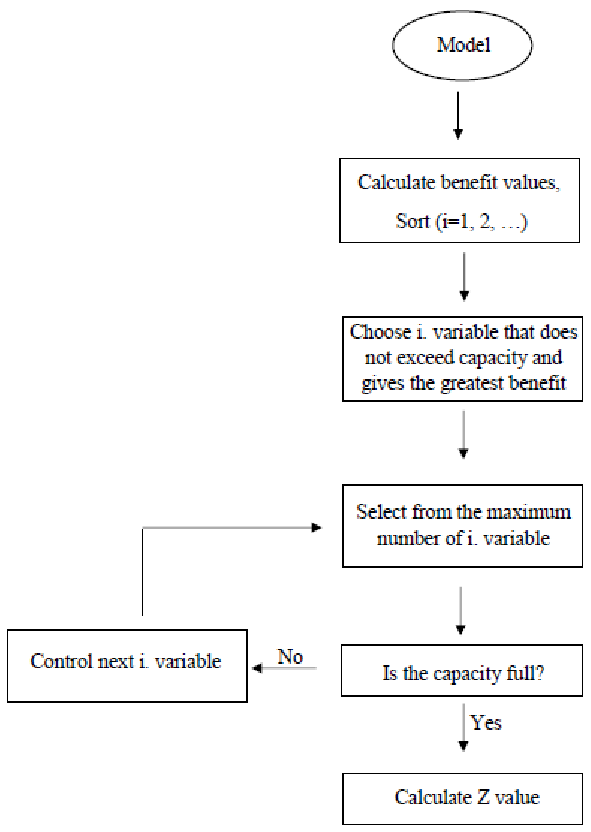

Model
Calculate benefit values, Sort (i=1, 2, …)
Choose i. variable that does not exceed capacity and gives the greatest benefit
Select from the maximum number of i. variable
Is the capacity full?
No
Control next i. variable
Yes
Calculate Z value

Figure 3. A general Greed flow diagram.

## 5. Application

In this part of the study, one-dimensional 0-1 and pure KP, two-dimensional 0-1 and pure KP were solved with Greedy algorithm. A software code has been developed to solve the KP with Greedy algorithm and multivariate problems have been solved with the help of this code. In addition, the branch-bound method was used from the classic algorithms and the results were compared with the results obtained from the pom-QM package.

### 5.1. Introduction of the Developed Program

When we look at the recent studies on IP, it can be said that it is preferred to write a program for the solutions of the problem considered here. With the widespread use of information technologies, it has been seen that code writing has yielded many benefits in terms of time, labor, calculation costs etc. therefore for KP solutions of Greedy algorithm on which we are working, a software program was developed [17].

The developed program was written in Java language and presented as desktop program in order to provide ease of use. The program solves one-dimensional (0-1 and pure) KP and two-dimensional (0-1 and pure) KP addressed in the current study with Greedy algorithm. In order to solve a desired problem with the developed program, it is necessary to enter the purpose and constraint function values, the capacity limit and to select the problem dimension. Figure 4 shows the main window of the program.

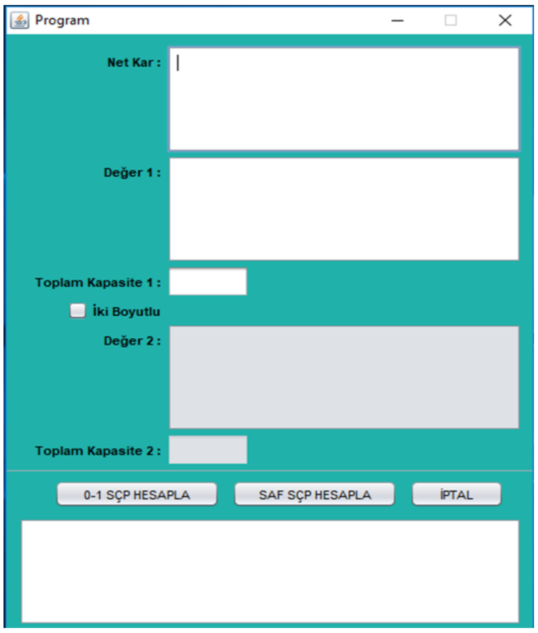


Figure 4. The main window of the developed program.

With Greedy algorithm, although the KP can be solved in a short time manually, this program facilitates the solution of multivariate problems and two-dimensional KP problems to a large extent. Also, while program packages such as pom-QM have limited iterations and variable numbers, this program can perform as many operations as requested and does not impose any restriction.

### 5.2. Problem: One-dimensional 0-1 KP

4-variable and single constraint sample problem is given below;

Objective function:

$$Max\ Z = 20x_1 + 108x_2 + 27x_3 + 12x_4 \qquad (10)$$

Constraints:

$$\begin{aligned} 4x_1 + 16x_2 + 9x_3 + 5x_4 &\leq 24 \\ x_i &= 0 \text{ or } 1 \end{aligned} \qquad (11)$$

In the solution of the problem with branch-boundary method, the results in each branch are calculated separately. For this example, the results were found to be $x_1 = x_2 = 1, x_3 = x_4 = 0$ and $Z = 128$. As we have already mentioned that various program packages have been developed, but calculation of results for each branch of the large scale problems could be problematical.

The solution of the problem with the pom-QM package was found to be the same as the branch-bound algorithm solution. Heuristic algorithms can be used to save time in solving the problem. The following steps can be defined to solve the problem with Greedy Heuristic.

**Step 1.** All the benefit $r_j = (c_j/a_j)$ values are calculated.

**Step 2.** Benefit values ($r_j$) are ranked. If there are same benefit values, their rank is randomly determined.

**Step 3.** The item offering the most benefit is put into the knapsack. The following benefit value is examined. If it does not exceed the capacity, it is put into the knapsack, if it exceeds, then the following item is examined.

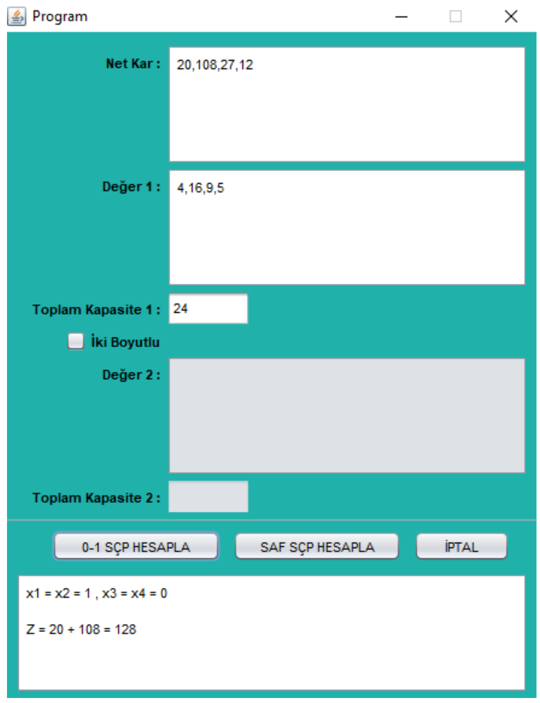
**Step 4.** If all the items have been checked, then the operation is concluded. Otherwise, it is returned to step 3.

According to these steps, Greedy algorithm solution of KP is given in Table 1.

Table 1. One-dimensional 0-1 KP benefit values.

| $x_j$ | $c_j$ | $a_j$ | $r_j = (c_j/a_j)$ | rank |
|---|---|---|---|---|
| $x_1$ | 20 | 4 | 5 | 2 |
| $x_2$ | 108 | 16 | 6.75 | 1 |
| $x_3$ | 27 | 9 | 3 | 3 |
| $x_4$ | 12 | 5 | 2.4 | 4 |

If we start to load with the highest value of good; $x_2$ is loaded first according to sorting, the reduced capacity is 16, the remaining capacity is 24 - 16 = 8. Then $x_1$ is loaded, the reduced capacity becomes 4, the remaining capacity becomes 8 - 4 = 4. The remaining 4 items have no good to be loaded, the process ends. In this case, the solution is $x_1 = x_2 = 1$, $x_3 = x_4 = 0$ and $Z = 128$. The output of the program developed using the above steps is as shown in Figure 5.



Net Kar : 20,108,27,12

Değer 1 : 4,16,9,5

Toplam Kapasite 1 : 24

İki Boyutlu

Değer 2 :

Toplam Kapasite 2 :

```
x1 = x2 = 1 , x3 = x4 = 0

Z = 20 + 108 = 128
```

Figure 5. One-dimensional 0-1 KP Greedy program output.

When we look at the results of branch-bound algorithm and Greedy algorithm, it is clear that we get the optimal solution with much economical processing.

### 5.2.1. Problem: One-dimensional 0-1 KP having different numbers of variables

Problems with variable numbers of 2, 3, 4, 8, 10, 15, 20, 25, 50 for 0-1 KP have been solved with the aid of the developed code and pom-QM program package.

Table 2. Solution results of one-dimensional 0-1 KP.

| Dimension | Pom-QM solution | Greedy solution |
|---|---|---|
| 2x1 | 8 | 8 |
| 3x1 | 17 | 17 |
| 4x1 | 21 | 21 |
| 8x1 | 52 | 43 |
| 10x1 | 60 | 60 |
| 15x1 | 87 | 87 |
| 20x1 | 147 | 126 |
| 25x1 | 653 | 653 |
| 50x1 | - | 61 |

According to Table 2, pom-QM program and Greedy software solutions have been found different for 8x1 and 20x1 problems. For the 50x1 problem, the pom-QM program could not produce a solution (since the number of variables in pom-QM is limited to 30). However, with Greedy program developed, the result was found to be 61.

### 5.3. Problem: One-dimensional pure KP

The sample problem with 4 variables and single constraint is given below;

Objective function:

$$Max\ Z = x_1 + 8x_2 + 30x_3 + 12x_4 \tag{12}$$

Constraints:

$$\begin{aligned} 4x_1 + 5x_2 + 6x_3 + 8x_4 &\leq 70 \\ x_i \geq 0 \ &and\ integer \end{aligned} \tag{13}$$

According to the branch-bound algorithm, the result of the problem was found to be $x_1 = 1, x_3 = 11, x_2 = x_4 = 0$ and $Z = 331$. If we solve the problem with pom-QM, the result will be the same.

The following steps can be defined to solve the problem with Greedy heuristic.

**Step 1.** All the benefit $r_j = (c_j/a_j)$ values are calculated.

**Step 2.** Benefit values ($r_j$) are ranked. If there are same benefit values, their rank is randomly determined.

**Step 3.** The item offering the most benefit is put into the knapsack. The following benefit value is examined. If it does not exceed the capacity, it is put into the knapsack, if it exceeds, then the following item is examined.

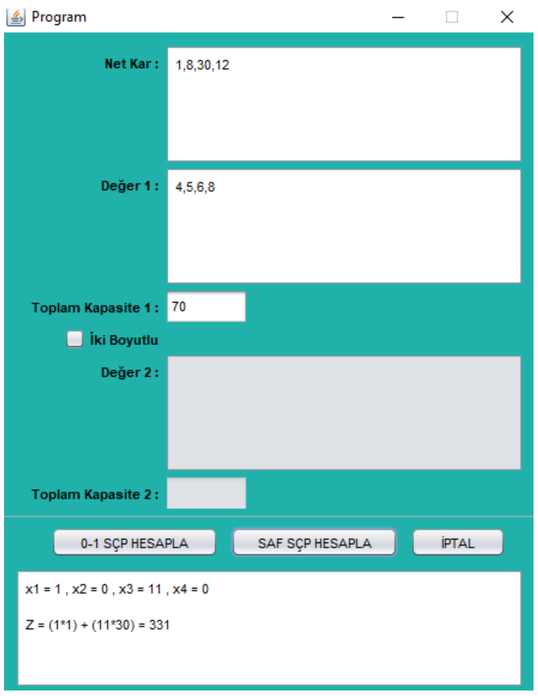
**Step 4.** If all the items have been checked, then the operation is concluded. Otherwise, it is returned to step 3.

According to these steps, the solution of KP is as in Table 3.

Table 3. One-dimensional pure KP benefit values.

| $x_j$ | $c_j$ | $a_j$ | $r_j = (c_j/a_j)$ | rank |
|---|---|---|---|---|
| $x_1$ | 1 | 4 | 0.25 | 4 |
| $x_2$ | 8 | 5 | 1.6 | 2 |
| $x_3$ | 30 | 6 | 5 | 1 |
| $x_4$ | 12 | 8 | 1.5 | 3 |

If we start to load with the good with the highest benefit value in such a way as not to exceed the maximum capacity; according to the rank order, first $x_3 = 11$ and the remaining capacity is 70 - 66 (6x11) = 4. $x_2$ and $x_4$ exceed the capacity, $x_1 = 1$ and the remaining capacity is 4 - 4 = 0. In this case, Greedy solution of the problem was found to be $x_1 = 1, x_3 = 11, x_2 = x_4 = 0$ and $Z = 331$. The program output of the solution with Greedy algorithm is given in Figure 6.



1,8,30,12

4,5,6,8

70

```
x1 = 1 , x2 = 0 , x3 = 11 , x4 = 0

Z = (1*1) + (11*30) = 331
```

Figure 6. One-dimensional pure KP Greedy program output.

Looking at the solutions with branch-bound and Greedy algorithms, it is seen that the optimal solution is achieved more economically, as in the one-dimensional 0-1 KP.

### 5.3.1. Problem: One-dimensional pure KP with different numbers of variables

For pure KP, problems with 2, 3, 4, 8, 10, 15, 20, 25, 50 variables were solved. The results of the problems are given in Table 4.

Table 4. Results of solutions for one-dimensional pure KP.

| Dimension | Pom-QM solution | Greedy solution |
|---|---|---|
| 2x1 | 24 | 24 |
| 3x1 | 96 | 96 |
| 4x1 | 66 | 15 |
| 8x1 | 257 | 257 |
| 10x1 | 288 | 288 |
| 15x1 | 100 | 100 |
| 20x1 | 226 | 15 |
| 25x1 | 800 | 628 |
| 50x1 | - | 200 |

As can be seen in Table 4, pom-QM program package results for 4x1, 20x1 and 25x1 are different when compared to the results of Greedy program. For 50x1, as the number of constraints with pom-QM exceeded 30, no solution could be found. The result of Greedy program was found to be 200.

### 5.4. Problem: Two-dimensional 0-1 KP

The current study addressed the constrained problems. In the previous problems, Greedy solutions of single constrained examples for KP were calculated from the utility values. In this problem, we will focus on how the two-dimensional (or two-constrained) KP is solved with Greedy algorithm.

Gorski et al. [19] put forward a method for solving two-dimensional KP problems. They gave their theorems for the method in the study together with the proofs and tried to explain the method with examples. In this method, according to the capacity differences for two constraints (for $b_1 > b_2 > 0$), Greedy algorithm can be applied to two constrained problems.

The problem given by Equation (14) below is solved according to differences between the benefit values and its' effect on the capacity, and the results are given.

Objective function:

$$Max\ Z = 35x_1 + 85x_2 + 135x_3 + 10x_4 + 25x_5 + 2x_6 + 94x_7 \quad (14)$$

Constraints:

$$2x_1 + 3x_2 + 9x_3 + 0.5x_4 + 2x_5 + 0.1x_6 + 4x_7 \leq 25$$
$$15 + 35x_2 + 105x_3 + 68x_4 + 125x_5 + 25x_6 + 100x_7 \leq 400 \quad (15)$$
$$x_i = 0 \text{ or } 1$$

The utility values are calculated in Table 5 to solve two-dimensional and 7 variables 0-1 KP with Greedy algorithm.

Table 5. Ranking of the two-dimensional 0-1 KP according to benefit values.

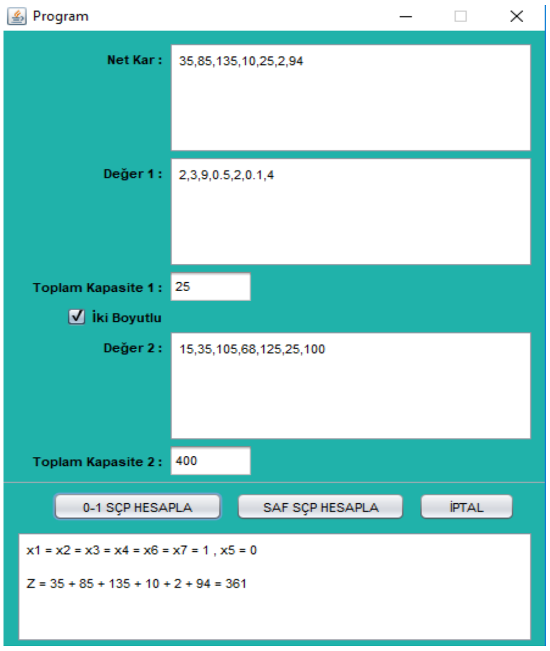
| $x_j$ | $c_j$ | $a_{1j}$ | $a_{2j}$ | $r_{1j}$ | $r_{2j}$ | $r_{1j} - r_{2j}$ | rank |
|---|---|---|---|---|---|---|---|
| $x_1$ | 35 | 2 | 15 | 17.5 | 2.33 | 15.17 | 5 |
| $x_2$ | 85 | 3 | 35 | 28.3 | 2.42 | 25.88 | 1 |
| $x_3$ | 135 | 9 | 105 | 15 | 1.28 | 13.72 | 6 |
| $x_4$ | 10 | 0.5 | 68 | 20 | 0.14 | 19.86 | 4 |
| $x_5$ | 25 | 2 | 125 | 12.5 | 0.2 | 12.3 | 7 |
| $x_6$ | 2 | 0.1 | 25 | 20 | 0.08 | 19.92 | 3 |
| $x_7$ | 94 | 4 | 100 | 23.5 | 0.94 | 22.56 | 2 |

The solution of the problem according to Greedy algorithm steps is given Table 6.

Table 6. The two-dimensional 0-1 KP results for Greedy algorithm.

| | weight (≤ 25) | volume (≤ 400) |
|---|---|---|
| $x_2 = 1$ | 25 – 3 = 22 | 400 – 35 = 365 |
| $x_7 = 1$ | 22 – 4 = 18 | 365 – 100 = 265 |
| $x_6 = 1$ | 18 – 0.1 = 17.9 | 265 – 25 = 240 |
| $x_4 = 1$ | 17.9 – 0.5= 17.4 | 240 – 68 = 172 |
| $x_1 = 1$ | 17.4 – 2 = 15.4 | 172 – 15 = 157 |
| $x_3 = 1$ | 15.4 – 9 = 6.4 | 157 – 105 = 52 |
| $x_5 = 0$ | | |

Greedy solution of the problem was found to be $x_1 = x_2 = x_3 = x_4 = x_6 = x_7 = 1,\ x_5 = 0$ and $Z = 361$. The program solution of the problem is shown in Figure 7.



Net Kar : 35,85,135,10,25,2,94
Değer 1 : 2,3,9,0.5,2,0.1,4
Toplam Kapasite 1 : 25
☑ İki Boyutlu
Değer 2 : 15,35,105,68,125,25,100
Toplam Kapasite 2 : 400

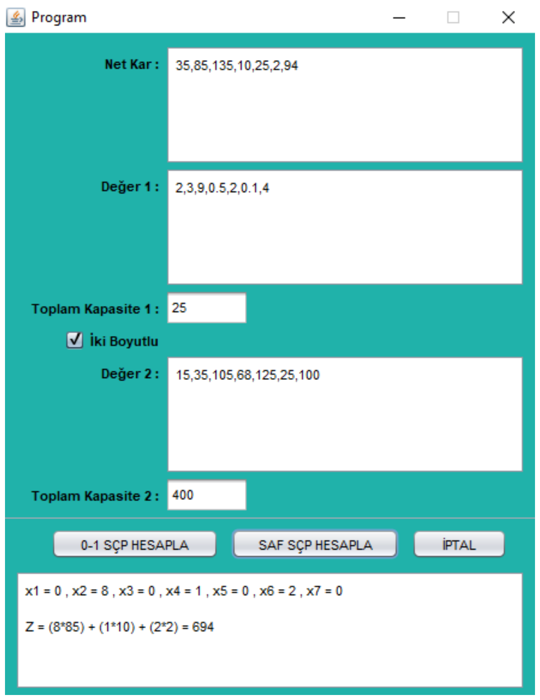
```
x1 = x2 = x3 = x4 = x6 = x7 = 1 , x5 = 0

Z = 35 + 85 + 135 + 10 + 2 + 94 = 361
```

Figure 7. Two-dimensional 0-1 KP Greedy program output

## 5.5. Problem: Two-dimensional pure KP

The problem given by Equation (14) above can be remodeled as pure KP. Greedy program solution according to utility matrix given for the newly defined problem is shown in Figure 8.



Net Kar : 35,85,135,10,25,2,94
Değer 1 : 2,3,9,0.5,2,0.1,4
Toplam Kapasite 1 : 25
☑ İki Boyutlu
Değer 2 : 15,35,105,68,125,25,100
Toplam Kapasite 2 : 400

```
x1 = 0 , x2 = 8 , x3 = 0 , x4 = 1 , x5 = 0 , x6 = 2 , x7 = 0

Z = (8*85) + (1*10) + (2*2) = 694
```

Figure 8. Two-dimensional pure KP Greedy program output.

Greedy solution of the problem was found to be $x_2 = 8, x_4 = 1, x_6 = 2$ and $Z = 694$.

Since there is not enough practice in the literature for solving mixed KP, it is not included in the program.

## 6. Conclusion

This study is designed to investigate whether Greedy heuristic algorithm gives close solutions when compared with classical algorithms. The solutions of problems were obtained with the appropriate classic algorithm and Greedy algorithm. The results were compared and it was seen that Greedy algorithm gave good results to a large extent. Taking all the problems considered in the study into account, it can be interpreted that Greedy algorithm gives good results (same as classic algorithms) in small-sized problems with small number of variables. However, if the number of variables is large, the classic algorithms are limited by the number of constraints and the number of variables; in such cases, the result can be found with Greedy.

The program developed for solving problems having different variable numbers with using Greedy algorithm provided great convenience in making comparisons in the study. The program solves one and two dimensional KP given in the literature. In addition, the program solves one and two dimensional capital budgeting problems with the same structure. Since this program is not limited by the number of constraints or iterations as it is in the pom-QM package, it can produce solutions for every problem.

Greedy algorithm can be used for the selection of initial population for GA, one of the heuristic algorithms. At the same time, it can be used in branch-bound algorithm; one of the classic algorithms, to trim the branches of search tree. For this reason, the demonstration that Greedy algorithm gives close results is important in terms of supporting future research and theoretical knowledge.

In the current study, Greedy algorithm solutions of 0-1 and pure IP problems have been discussed. In future studies, it is aimed to investigate the solution by using Greedy algorithm for mixed IP and more complex IP models. After these studies, it is considered to develop a program for solving more complicated problem models and problems with larger number of dimensions using Greedy algorithm.

Other codes written for the heuristic algorithm produce a solution for a single problem. When the program developed in the current study for this purpose is compared with the other software codes, it can be seen that it is more advantageous as it does not include any constrain, is easy to use and can solve capital budgeting problems having a similar model structure. In future studies, it is aimed to develop codes for other algorithms and then turn them into a program package that can perform heuristic solutions.

## 7. References

[1] Bakır, M.A. and Altunkaynak, B., *Tamsayılı Programlama Teori, Modeller ve Algoritmaları*, Nobel Yayın Dağıtım, Ankara, 2003.

[2] Başkaya, Z., Tamsayılı Programlama Algoritmaları ve Bilgisayar Uygulamalı Problem Çözümleri, Başak Matbaacılık, Ankara, 2005.

[3] Güler, A., *Tamsayılı Programlama Problemleri İçin Garanti Değerli Algoritmalar*, Ege University, Graduate School of Natural and Applied Sciences, Master Thesis, İzmir, 2008.

[4] Akçay, Y., Li, H. and Xu, S.H, "Greedy Algorithm for the General Multidimensional Knapsack Problems", *Annals of Operations Research*, 150, 17-29, 2007.

[5] Liu, L., "Solving 0-1 Knapsack Problems by Greedy Method and Dynamic Programming Method", *Advanced Materials Research*, 282-283, 570-573, 2011.

[6] Lv, J., Wang, X., Huang, M., Cheng, H. and Li, F., "Solving 0-1 Knapsack Problem by Greedy Degree and Expectation Efficiency", *Applied Soft Computing*, 41, 94-103, 2016.

[7] Zhou, Y., Chen, X. and Zhou, G., "An Improved Monkey Algorithm for a 0-1 Knapsack Problem", *Applied Soft Computing*, 38, 817-830, 2016.

[8] Bulut, F. and İnce, İ.F., "Tam Sayı Programlamada Açgözlü ve Sezgisel Aramalar ile 0-1 Sırt Çantası Problemine Yeni Bir Bakış", *Karaelmas Fen ve Mühendislik Dergisi*, 8(1), 89-98, 2018.

[9] Winston, W.L., Operations Research Applications and Algorithms, Canada, 2004.

[10] Taha, H., *Yöneylem Araştırması*, Literatür Yayıncılık, İstanbul, 2000.

[11] Hillier, F. S. and Lieberman, G. J., *Introduction to Operations Research*, McGraw-Hill, New York, 2001.

[12] Schrijver, A., *Theory of Linear and Integer Programming*, A Wiley-Interscience Publication, Amsterdam, 1999.

[13] Keskintürk, T., Topuk, N. and Özyeşil, O., "Araç Rotalama Problemleri İle Çözüm Yöntemlerinin Sınıflandırılması ve Bir Uygulama", *The Journal of Business Science*, 3(2), 77-107, 2015.

[14] Sağır, M., Öztürk, A. and Öztürk, Ö., *Yöneylem Araştırması-2*, Anadolu Üniversitesi Açıköğretim Yayınları, Eskişehir, 2013.

[15] Yıldırım, T., Kalaycı, C.B. and Mutlu, Ö., "Gezgin Satıcı Problemi İçin Yeni Bir Meta Sezgisel: Kör Fare Algoritması*", Pamukkale University Journal of Engineering Sciences*, 22(1), 64-70, 2016.

[16] Pearl, J., Heuristics Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley Publishing Company, 1984.

[17] Durmuş, B., *Tamsayılı Programlamada Klasik ve Greedy Sezgisel Algoritma Sonuçlarının Karşılaştırılması*, Muğla Sıtkı Koçman University, Graduate School of Natural and Applied Sciences, Master Thesis, Muğla, 2018.

[18] Alwan, H.O. and Farhan, N.M., "Load Restoration Methodology Considering Renewable Energies and Combined Heat and Power Systems", *International Journal of Engineering and Technology*, 7(2.6), 130-134, 2018.

[19] Gorski, J., Paquete, L. and Pedrosa, F., "Greedy Algorithms for a Class of Knapsack Problems with Binary Weights", *Computers and Operations Research*, 39, 498-511, 2012.