Application Of The Greedy Algorithm In Multiple Constrain Knapsack Optimization Problems In Goods Transportation

Gradina Nur Fauziah
Makassar Marine Polytechnic, Indonesia
E-mail:gradina.nur.f@pipmakassar.ac.id

ARTICLE INFO
Received
March 24 2022
Revised
13 April, 2022
Approved
23 May 2022

Keywords:
Greedy, Optimization, Algorithm, Knapsack Proble, Multiple Troubled Backpacks

ABSTRACT
Maximization and minimization problems are common problems occurred in commercial activities. One of the methods to solve the problem is by using optimization. The use of optimization has proven to be beneficial in improve productivity performance. Transportation is one of the areas that are related to optimization. Problems related to obtaining maximum profits in transporting goods with the use of limited transport capacity is remain as major problems. This problem is often analogous to using the Knapsack Problem theory. Greedy algorithm applied to Knapsack Multiple Constrain calculations provides better results than manual calculations.

Introduction

The growing competition in the business world, encourages the use of technology to be able to guarantee maximum profit by only using resources as efficiently as possible from various existing limitations (Shin & Kehm, 2012). An example in daily life is in the problem when someone chooses an object from a set of objects, each of which has a weight and a value to be loaded into a storage medium with certain space limitations so that profits can be obtained. maximum of these objects. This kind of problem is called the knapsack problem (Rachmawati & Chandra, 2013).

Sometimes human limitations in solving knapsack problems without using tools are to finding the optimum solution. Especially if there are too many objects occured, the calculations will be more difficult. Time efficiency is also the important factor (Prasetiyowati & Wicaksana, 2013) (Assi & Haraty, 2018). Therefore, we need a method as well as an application program that can help solve the knapsack problem.

The knapsack problem is divided into three, namely integer knapsack, bounded knapsack, and unbounded knapsack. The problem with the integer knapsack is to determine which objects should be loaded or not loaded on the storage medium. The

How to cite: Fauziah, Gradina Nur (2022) Application Of The Greedy Algorithm In Multiple Constrain Knapsack Optimization Problems In Goods Transportation, *Jurnal Syntax Admiration 3*(5) https://doi.org/10.46799/jsa.v3i4.421
E-ISSN: 2722-5356
Published by: Ridwan Institute

bounded knapsack problem is determining how many parts of each object will be loaded in the storage media, while in the unbounded knapsack the problem is determining for unlimited items (Martello & Toth, 1990) (Boyer et al., 2010).

Knapsack problems can be solved in various ways. There are several algorithmic strategies that can produce optimal solutions including greedy algorithms, dynamic programming, branch and bound, brute force, and genetics. But these algorithms have different characteristics and have different time complexity . Greedy algorithm is the most common algorithm that is often used to solve this Knapsack problem (Pisinger, 1999).

One of the most frequently studied Knapsack problems is the integer Knapsack with one problem, while in this paper the researcher tries to solve the Knapsack problem but with more than one Knapsack problem. Previous research has tried to solve the Knapsack integer problem with the Greedy Algorithm and Dynamic Programming (Arista, 2013) and solve the Multiple Constraint Knapsack problem with the Dynamic Programming method (Hilviah, 2015) (Martello & Toth, 1990).

Research methods

The steps of the research carried out to solve the Knapsack Multiple Constrain problem with the application of the Greedy Algorithm are conducting a literature study on the Multiple Constraint Knapsack problem in the field of transportation of goods (Pisinger, 1995) (Pisinger & Toth, 1998).

Conducting a literature study on the application of the Greedy algorithm to solving the Multiple Constraint Knapsack problem (García-Martínez et al., 2014). Identify the problems and constraints faced in the application of the Greedy algorithm as a problem solving. Conducting experiments using simple programming applications to simulate the solution to the Multiple Constrain Knapsack problem in the transportation of goods. Conduct analysis and discussion of the simulation results. Make conclusions from research activities that have been carried out (Toth, 1980).

Results and Discussion

A. Examples of Application of Greedy Algorithm in Goods Transportation Problem

In table 1 there is a conveyance with a maximum transport capacity of 100 kg and dimensions of 3 m x 2 m x 2 m (maximum volume = 12 m^3), there are 7 items to be transported with the following sizes:

Table 1. Table detailing the weight, volume and profit of each item

j item	Weight (Kg) w1h	Volume (m3) w2j	Profit (Rp) pj
1	14	0.5	30000
2	20	2	50000
3	12	1	80000
4	6	3	75000
5	30	2.5	40000
6	10	3	60000

j item	Weight (Kg) w1h	Volume (m3) w2j	Profit (Rp) pj
7	15	2	30000

Greedy algorithm will be used to find the optimal value from the example above, where there are two kinds of decisions that can be taken, namely:

a. Goods transported (1)
b. Goods not transported (0)

The application of each Greedy algorithm is shown in the table below:

a. Greedy by Profit

In table 2 below, the results of the example calculation above are presented using Greedy by Profit, namely by sorting the profit of each item in descending order (large to small).

Table 2. Greedy by Profit

Item J	W1 j	W1 j	Pj	Status
3	12	1	80000	1
4	6	3	75000	1
6	10	3	60000	1
2	20	2	50000	1
5	30	2.5	40000	1
1	14	0.5	30000	1
7	15	2	30000	0
Total	**92**	**12**	**335000**	

With the Greedy by Profit algorithm, the optimal solution is obtained by transporting goods 1 to 6 and goods 7 not being transported. The maximum profit obtained is Rp. 335,000

b. Greedy by Weight

In table 3 below, the results of the calculation of the example above are presented using Greedy by Weight, namely by sorting the weight of each item in ascending order (small to large).

Table 3. Greedy by Weight

Item J	W1 j	W1j	Pj	Status
4	6	3	75000	1
6	10	3	70000	1
3	12	1	80000	1
1	14	0.5	30000	1
7	15	2	30000	1
2	20	2	50000	1
5	30	2.5	40000	0
Total	**77**	**11.5**	**325000**	

With the Greedy by Weight algorithm, the optimal solution is obtained by transporting goods 1,2,3,4,6 and 7 and goods 5 not being transported. The maximum profit obtained is Rp. 325,000

c. *Greedy by Volume*

In table 4 below, the results of the calculation examples above are presented using Greedy by Volume, namely by sorting the volume of each item in ascending order (small to large).

Table 4. Greedy by Volume

Item J	W1 j	W1 j	Pj	Status
1	14	0.5	30000	1
3	12	1	80000	1
7	15	2	30000	1
2	20	2	50000	1
5	30	2.5	40000	1
6	10	3	60000	1
4	6	3	75000	0
Total	**91**	**8**	**230000**	

With the Greedy by Volume algorithm, the optimal solution is obtained by transporting goods 1, 2, 3, 5 and 7 and goods 4 and 6 not being transported. The maximum profit obtained is Rp. 230,000

d. *Greedy by Density*

1) Profit / Weight

In table 5 below, the results of the calculation of the example above using Greedy by Density are obtained from profit by weight, namely by sorting the profit value of each weight of each item in descending order (large to small).

Table 5. Greedy by Density of Weight

Item J	W1 j	W1j	Pj	Status
4	6	3	75000	1
3	12	1	80000	1
6	10	3	60000	1
2	20	2	50000	1
1	14	0.5	30000	1
7	15	2	30000	1
5	30	0.5	40000	0
Total	**77**	**11.5**	**1333.33**	

With the Greedy by Density of Weight algorithm, the optimal solution is obtained by transporting goods 1, 2, 3, 4, 6 and 7 and goods 5 not being transported. The maximum profit obtained is Rp. 325,000

2) *Profit /* Volume

In table 6 below, the results of the calculation of the example above using Greedy by Density are obtained from profit by volume, namely by sorting the profit value of each weight of each item in descending order (large to small).

Table 6. Greedy by Density of Volume

Item J	W1 j	W1 j	Pj	Pj/W 2j	Status
3	12	1	80000	80000	1
1	14	0.5	30000	60000	1
4	6	3	75000	25000	1
2	20	2	50000	25000	1
6	10	3	60000	20000	1
5	30	2.5	40000	16000	1
7	15	2	30000	15000	0
Total	**92**	**12**	**3350000**		

With the Greedy by Density of Volume algorithm, the optimal solution is obtained by transporting goods 1 to 6 and goods 7 not being transported. The maximum profit obtained is Rp. 335,000

Table 7 shows a comparison of the use of each of these algorithms in the sample questions.

Table 7. Comparison of each algorithm

j item	Greedy by weight	Greedy by volume	Greedy by profit	Greedy by density of weight	Greedy by density of volume	Optimal solution
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	0	1	1	1	1
5	0	1	1	0	1	1
6	1	0	1	1	1	1
7	1	1	0	1	0	0
Total weight	77	91	92	77	92	92
Total volume	11.5	8	12	11.5	12	12
Total profit	325000	230000	335000	325000	335000	335000

Based on the table above, the application of each Greedy algorithm gives different results. The optimal solution is obtained by applying the Greedy by Profit and Greedy by Density of Volume algorithms. However, the Greedy algorithm applied in this example is quite close to the expected optimal solution.

Analysis

Based on the theory and examples of the Greedy algorithm in solving the Multiple Constrain Knapsack problem, the Pseudocode of the Greedy algorithm is as follows:
function Knapsack(input C : object_set, K1 : real, K2 : real) → solution_set
{ Generates a solution to the multiple constraint knapsack problem with a greedy algorithm that uses an object selection strategy based on profit (pj), weight (w1j), volume (w2j), density of weight (pj/w1j), density of volume (pj/w2j). The solution is expressed as a vector X = x[1], x[2], …, x[n].

Assumptions:
All Goods have the same shape
For Greedy by profit, all objects are sorted based on the decreasing pj value.
For Greedy by weight, all objects are sorted based on the increasing value of w1j.
For Greedy by volume all objects are sorted by increasing w2j value.
For Greedy by density of weight all objects have been sorted based on the decreasing pj/w1j value.
For Greedy by density of volume, all objects are sorted by decreasing pj/w2j values}

Declaration

j, TotalWeight : integer , TotalVolume : non integer
Available: boolean
x : solution_set

Algorithm:

```
for j 1 to n do
  x[j] 0 { initialize each fetch state of object i with 0 }
endfor
j 0
TotalWeight 0
TotalVolume←0
Available true
while (j← n) and (Available) do
{ check the jth object }
  j j + 1
  if
TotalWeight + w[1h] K1 and TotalVolume +w[2h] K2
then
{ insert object Cj into knapsack }
x[j] 1
TotalWeight TotalWeight + w[1h]
TotalVolume←TotalVolume + w[2h]
else
Available false
x[j] 0
{ object Ci not included in knapsack }
```

  endif
end while
{ j > n or not Available }
return x

Flow chart
Figure 1 illustrates the general flowchart for the implementation of the Greedy algorithm on the Multiple Constraint Knapsack Problem

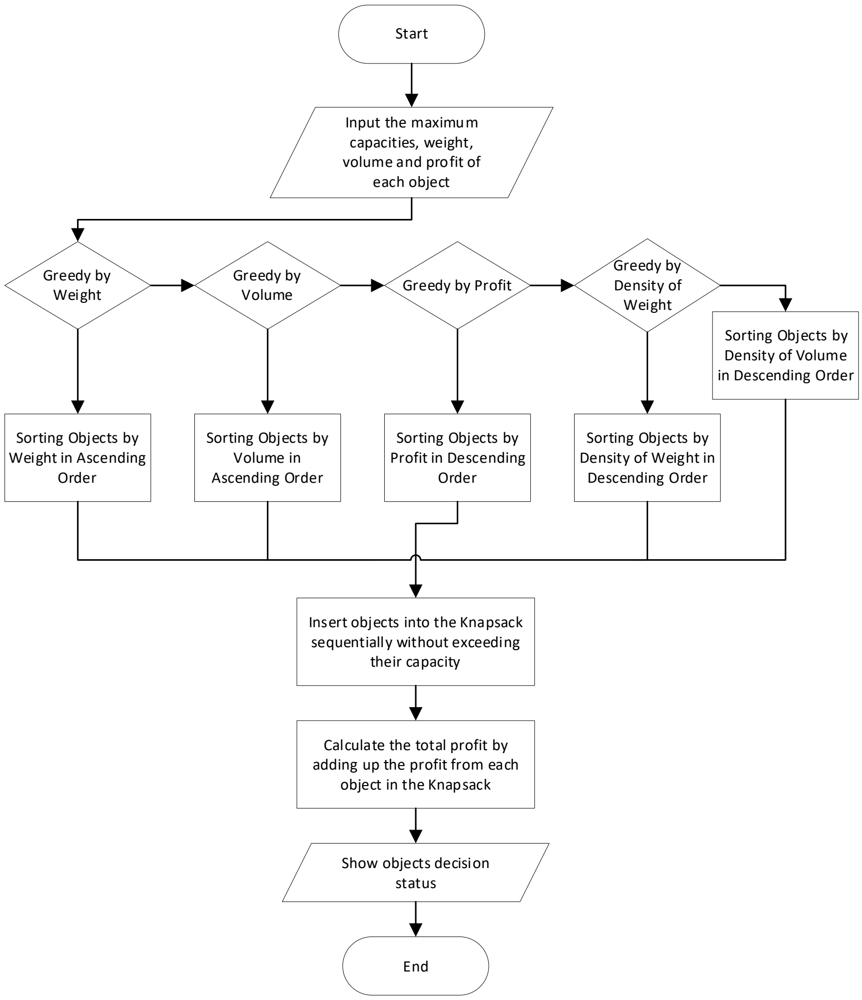

Image 1.
Flowchart of Greedy Algorithm application on Multiple Constraint Knapsack

Conclusion

From the results of the analysis, it can be concluded that the performance of Greedy by Weight and Greedy by Volume is lower when compared to Greedy by Profit and Greedy by Density which are close to the optimal solution for the 0/1 Multiple Constrain Knapsack Problem.

BIBLIOGRAPHY

Arista, W. M. (2013). Penerapan Algoritma Greedy dan Dynamic Programming pada Permasalahan Integer Knapsack.Google Scholar

Assi, M., & Haraty, R. A. (2018). A survey of the knapsack problem. *2018 International Arab Conference on Information Technology (ACIT)*, 1–6. Google Scholar

Boyer, V., El Baz, D., & Elkihel, M. (2010). Solution of multidimensional knapsack problems via cooperation of dynamic programming and branch and bound. *European Journal of Industrial Engineering*, *4*(4), 434–449. Google Scholar

García-Martínez, C., Rodriguez, F. J., & Lozano, M. (2014). Tabu-enhanced iterated greedy algorithm: a case study in the quadratic multiple knapsack problem. *European Journal of Operational Research*, *232*(3), 454–463. Google Scholar

Hilviah, F. (2015). Penerapan Algoritma Dynamic Programming dan Algoritma Backtracking pada Permasalahan Multiple Constraints Knapsack 0-1. Google Scholar

Martello, S., & Toth, P. (1990). Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc. Google Scholar

Pisinger, D. (1995). A minimal algorithm for the multiple-choice knapsack problem. European Journal of Operational Research, 83(2), 394–410. Google Scholar

Pisinger, D. (1999). An exact algorithm for large multiple knapsack problems. European Journal of Operational Research, 114(3), 528–541. Google Scholar

Pisinger, D., & Toth, P. (1998). Knapsack problems. In Handbook of combinatorial optimization (pp. 299–428). Springer. Google Scholar

Prasetiyowati, M. I., & Wicaksana, A. (2013). Implementasi Algoritma Dynamic Programming untuk Multiple Constraints Knapsack Problem (Studi Kasus: Pemilihan Media Promosi di UMN). Seminar Nasional Aplikasi Teknologi Informasi (SNATI), 1(1). Google Scholar

Rachmawati, D., & Chandra, A. (2013). Implementasi algoritma greedy untuk menyelesaikan masalah knapsack problem. *Jurnal: Saintikom*, *12*(3). Google Scholar

Shin, J. C., & Kehm, B. M. (2012). *Institutionalization of world-class university in global competition* (Vol. 6). Springer Science & Business Media. Google Scholar

Toth, P. (1980). Dynamic programming algorithms for the zero-one knapsack problem. *Computing*, *25*(1), 29–45. Google Scholar