International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST218389

A Study on Greedy Technique in Container Loading Problem and Knapsack Problem

S. Sathyapriya[1] , V. Arundhathi[2], K. Aiswarya[3], S. R. Aarthi[4], S. Vishnu[5]

[1] Assistant Professor, Department of Mathematics, Sri Krishna Arts and Science College, Coimbatore, Tamil Nadu, India

[2,3,4,5] UG Scholar, Department of Mathematics, Sri Krishna Arts and Science College, Coimbatore, Tamil Nadu, India

Article Info
Volume 8, Issue 3
Page Number : 414-420

Publication Issue
May-June-2021

Article History
Accepted : 15 May 2021
Published : 30 May 2021

ABSTRACT

The main aim of the paper is to use application of greedy algorithm in container loading problem and Knapsack problem. Greedy method gives an optimal solution to the problem by considering the inputs one at a time, checking to see if it can be included in the set of values which give an optimal solution and then check if it is the feasible solution. The Greedy algorithm could be understood very well with a well-known problem referred to as container loading problem and Knapsack problem. The basic Container Loading Problem can be defined as the problem of placing a set of boxes into the container respecting the geometric constraints: the boxes cannot overlap and cannot exceed the dimensions of the container. The knapsack problem is in combinatorial optimization problem. It appears as a sub problem in many, more complex mathematical models of real world problems.

Keywords : Greedy Method, Container Loading Problem, Knapsack Problem

I. INTRODUCTION

Operations research is concerned with the systems in which human behaviour plays an important role. The operations research focuses on the whole system rather than focusing on individual parts of the system. Different types of approaches are applied by Operations research to deal with different kinds of problems. Applications are abundant such as in airlines, manufacturing companies, service organizations, military branches, and government. The range of problems and issues to which it has contributed insights and solutions is vast. In that A Greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. Hence, it is extremely important to reason about the correctness of the greedy strategy before using it to solve a problem

II. METHODS AND MATERIAL

A Greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage.[1] In many problems, a

greedy strategy does not usually produce an optimal solution, but nonetheless, a greedy heuristic may yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time. In mathematical optimization, greedy algorithms optimally solve combinatorial problems having the properties of matroids, and give constant-factor approximations to optimization problems with submodular structure. An algorithm is called greedy if it follows the problem-solving heuristic of making the locally optimal choice at each stage with the aim of finding a global optimum.

Specification:

In general, greedy algorithms have five components:

- ✓ A candidate set, from which a solution is created.
- ✓ A selection function, which chooses the best candidate to be added to the solution.
- ✓ A feasibility function, that is used to determine if a candidate can be used to contribute to a solution.
- ✓ An objective function, which assigns a value to a solution, or a partial solution, and
- ✓ A solution function, which will indicate when we have discovered a complete solution

Types of Solutions:

1. Feasible Solution:

Now when a problem arises, we have many plausible solutions to this problem. Yet, taking into consideration the condition set on that problem, we choose solutions that satisfy the given condition. Such solutions that help us get results meeting the given condition is called a Feasible Solution

2. Optimal Solution:

A solution is called optimal when it is already feasible and achieves the objective of the problem; the best result. This objective could either be the minimum or maximum result. The point here to be noticed is that any problem will only have one optimal solution.

APPLICATION OF GREED METHOD:

The greedy method can be applied to many real life problems. Two samples of those applications is taken:

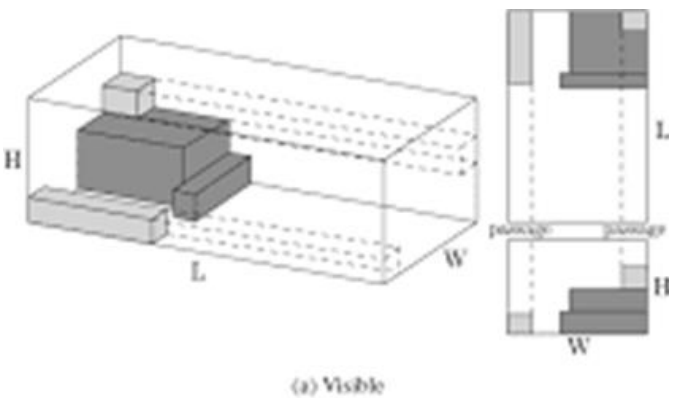
1. container loading problem
2. knapsack problem

Container Loading Problem

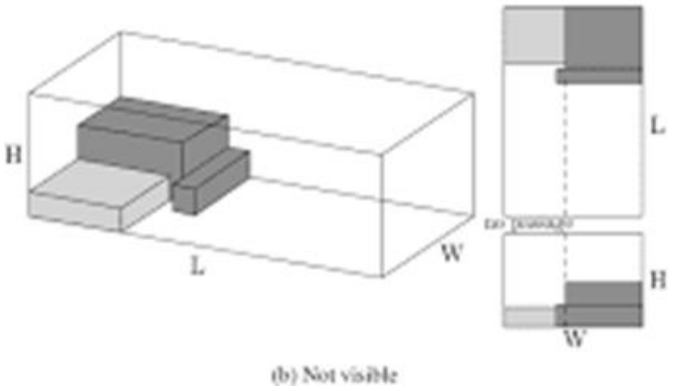
The basic Container Loading Problem can be defined as the problem of placing a set of boxes into the container respecting the geometric constraints: the boxes cannot overlap and cannot exceed the dimensions of the container.

Constructive Algorithm:

The basis of our constructive algorithm is the constructive procedure proposed by Parren˜o et al. (41) for the container loading problem. The algorithm works on maximal spaces. As the selected box is packed into a new space, three new maximal spaces are created. The constructive algorithm uses an updated list of maximal spaces and a list of boxes for the current client still to be packed. The most difficult part in this case is the management of the empty spaces when all the boxes for a client have been packed. For completeness we will outline its main steps here.



(a) Visible



Container Loading Problem

Step 1: Initialization

S= the list of empty maximal spaces created when packing the boxes. Initially, S is just the empty container.
B = {B_1, B_2, ..., B_n }, the set of boxes still to be pached, ordered by customer.

Step 2: Choosing the maximal space from S

We have a list S of empty maximal spaces, which are the largest empty parallelepipeds available to be filled with boxes. We use two alternative strategies to choose the maxi- mal space.
We alternate between choosing the maximal space with the minimum distance to the back of the container and choosing the maximal space with the largest coordinate z. The reason behind these strategies is to fill the back of the container first or to pile boxes forming stacks.

Step 3: Choosing the boxes to pack

Once a maximal space S has been chosen, we consider the remaining boxes of the cur- rent customer fitting into S in order to choose which one to pack. If there are several boxes of the same dimensions, we consider the possibility of packing a block, that is, packing several of these boxes arranged in a rectangular array with several rows and columns.

Step 4: Updating the list S for the current customer

Unless the box or block fits exactly into space S, packing it produces new empty maxi- mal spaces which will replace S in the list S. Moreover, as the maximal spaces are not disjoint, the box or block being packed can intersect with other maximal spaces which will have to be reduced.

Once the new spaces have been added and some of the existing ones modified, we check the list and eliminate possible inclusions. The list B is also updated and the max- imal spaces that cannot accommodate any of the boxes still to be packed are eliminated from S. If S =Ø or B=Ø, the procedure ends. Otherwise, if we have not finished packing the boxes for the current customer, we go back to Step 1.

Step 5: Updating the list S for a new customer.

Constraint:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

where x_i is the number of the containers ,
where w_i is the weight of the container, where C is the Capacity of the container.

1) Knapsack Problem:

Introduction

The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

Definition:

The most common problem being solved is the knapsack problem, which restricts the number x_i of copies of each kind of item to zero or one. Given a set

of n items num- bered from 1 up to n, each with a weight w_i and a value v_i, along with a maximum weight capacity W,

Maximum Profit

$$\sum_{i=1}^{n} v_i x_i$$

Subject to Constraint

$$\sum_{i=1}^{n} x_i w_i \leq W$$

where $x_i \in \{0, 1\}$

Here x_i represents the number of instances of item i to include in the knapsack. Informally, the problem is to maximize the sum of the values of the items in the knap- sack so that the sum of the weights is less than or equal to the knapsack's capacity.

The bounded knapsack problem (BKP) removes the restriction that there is only one of each item, but restricts the number of x_i copies of each kind of item to a maximum non-negative integer value c :

Maximum Profit

$$\sum_{i=1}^{} v_i x_i$$

Subject to Constraint

$$\sum_{i=1}^{n} x_i w_i \leq W$$

where $x_i \in \{0, 1, 2, 3..., c\}$

The unbounded knapsack problem (UKP) places no upper bound on the number of copies of each kind of item and can be formulated as above except for that the only restriction on x_i.

Maximum Profit

$$\sum_{i=1}^{n} v_i x_i$$

Subject to Constraint

$$\sum_{i=1}^{n} x_i w_i \leq W$$

where $x_i \geq 0$, $x_i \in Z$

III. PROBLEMS

Example 1

A Logistic Company has to load a maximum number of cargo in a ship.The To- tal number of container is 8 and each weighs about 100, 200, 50, 90, 150, 50, 20, 80.There- fore the total capacity of cargo is 400.How should the container be loaded to max- imize the total return?

Solution:-

Given:

Capacity of cargo C = 400

Number of Containers = 8

W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
100	200	50	150	90	50	20	80

Constraint for Container loading problem,

$$\sum_{i=1}^{n} x_i w_i \leq C$$

If the container is loaded $x_i = 1$

If the container is not loaded $x_i = 0$

Arrange the weight of the container in Ascending order.

Container number : Weights

{7,3,6,8,4,1,5,2} : {20,50,50,80,90,100,150,200}

Let the initial value of the solution set is{0,0,0,0,0,0,0,0}

Step 1:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

w_i= 20; C = 400 ; The container is loaded, so x_i = 1.

20×1 ≤ 400

20≤ 400

The weight of the container 7 is less than 400. So container 7 can be loaded.

Solution Set is {0,0,0,0,0,0,1,0}.

Step 2:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

w_i= 50; C = 400 ; The container is loaded, so x_i = 1.

20+50×1 ≤ 400

70≤ 400

The weight of the container 3 is less than 400. So container 3 can be loaded.

Solution Set is {0,0,1,0,0,0,1,0}.

Step 3:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

w_i= 50; C = 400 ; The container is loaded, so x_i = 1.

70+50×1 ≤ 400

120≤ 400

The weight of the container 6 is less than 400. So container 6 can be loaded.

Solution Set is {0,0,1,0,0,1,1,0}.

Step 4:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

w_i= 80; C = 400 ; The container is loaded, so x_i = 1.

120+80×1 ≤ 400

200≤ 400

The weight of the container 8 is less than 400. So container 8 can be loaded.

Solution Set is {0,0,1,0,0,1,1,1}.

Step 5:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

w_i= 90; C = 400 ; The container is loaded, so x_i = 1.

200+90×1 ≤ 400

290≤ 400

The weight of the container 4 is less than 400. So container 4 can be loaded.

Solution Set is {0,0,1,1,0,1,1,1}.

Step 6:

$$\sum_{i=1}^{n} x_i w_i \leq C$$

w_i= 100; C = 400 ; The container is loaded, so x_i = 1.

290+100×1 ≤ 400

$390 \leq 400$

The weight of the container 1 is less than 400. So container 1 can be loaded.

Solution Set is *{*1,0,1,1,0,1,1,1*}*.

Step 7:

n

$x_i w_i \leq C$

i=1

w_i= 150; C = 400 ; The container is loaded, so x_i = 1.

$390+150\times1 \leq 400$

$540 \leq 400$

The weight of the container 5 is greater than 400. So container 5 cannot be loaded.

Capacity of Cargo exceeds,we cannot move further.

Hence the Solution Set is *{*1,0,1,1,0,1,1,1*}*.

The containers 1,3,4,6,7,8 only can be loaded.

Example 2

For the given set of items and knapsack capacity = 60 kg, find the optimal so- lution for the fractional knapsack problem making use of the greedy approach.

Item	A	B	C	D
Profit P_i	280	100	120	120
Weight W_i	40	10	20	24

Solution:-

Consider:
n = 4

m = 60 kg

{*w1, w2, w3, w4*}* = *{*40,10,20,24*}
{*p1, p2, p3, p4*}* = *{*280,100,120,120*}

Step 1:

Find out profit per weight Pi/Wi

Item	A	B	C	D
Profit P_i	280	100	120	120
Weight W_i	40	10	20	24
Profit Ratio P_i/W_i	7	10	6	5

Step 2:

Arrange the Profit ratio in descending order

Item	B	A	C	D
Profit P_i	100	280	120	120
Weight W_i	10	40	20	24
Profit Ratio P_i/W_i	10	7	6	5

let's have the capacity **m =60 kgs**

- The first profitable item we have are item B so we select is 60-10=50 now the remaining knapsack capacity is 50 and our selection is 1 (means selected)
- Then we have the next profitable item is item A, so we select 50-40=10 now the remaining knapsack capacity is 10 and our selection is 1(means selected)
- Then we have the next profitable item is item C and its weight is 20 and our knap- sack remaining capacity is 10. Now we are dealing with a greedy approach and select 10/20 items. (like take as we can)
- So our selection is 10/20
- Now we don't have the remaining capacity so we can't take the last item no D so it's selection is 0.

Subject to Constraint :

$$\sum_{i=1}^{n} x_i w_i \leq m$$

Therefore the following B,A,C satisfies the constraints in the given Knapsack problem.

Maximum Profit:

$$\sum_{i=1}^{n} p_i x_i$$

= 100 + 280 + 120 * (10/20)

= **380 + 60**

= **440 units.**

This is the optimal solution. We cannot gain more profit selecting any different combination of items.

IV. CONCLUSION

We have reviewed greedy algorithms and its application. The main contribution is that this algorithm is able to accommodate the versions of the constraints to have ap- peared in the container loading problem and Knapsack Problem and it obtains very good results to find the optimal solution. Greedy Technique is a straight forward design which can be applied to a variety of problems. In every step it selects the optimal value. In this project, it emphasizes main applications of Container loading problem and Knapsack problem. The algorithm is flexible and could be adapted to other constraints, such as the maximum weight of the container, weight distribution, and boxes with profit ratio. In the near future we plan to use this algorithm to work with more practical situations.

V. REFERENCES

[1]. "Operational Research in the British Army 1939–1945", October 1947, Report C67/3/4/48, UK National Archives file WO291/1301

[2]. "Operations research (industrial engineering) :: History – Britannica Online En- cyclopedia". Britannica.com. Retrieved 13 November 2011.

[3]. Black, Paul E. (2 February 2005). "greedy algorithm". Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology (NIST). Re- trieved 17 August 2012.

Cite this article as :

S. Sathyapriya, V. Arundhathi, K. Aiswarya, S. R. Aarthi, S. Vishnu, "A Study on Greedy Technique in Container Loading Problem and Knapsack Problem", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 8 Issue 3, pp. 414-420, May-June 2021. Available at doi : https://doi.org/10.32628/IJSRST218389
Journal URL : https://ijsrst.com/IJSRST218389