ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/333845063

A Survey of the Knapsack Problem

Conference Paper · November 2018

DOI: 10.1109/ACIT.2018.8672677

CITATIONS
37

READS
443

2 authors:

Maram Assi
Queen's University
8 PUBLICATIONS **113** CITATIONS
SEE PROFILE

Ramzi A. Haraty
Lebanese American University
188 PUBLICATIONS **1,351** CITATIONS
SEE PROFILE

All content following this page was uploaded by Maram Assi on 22 April 2024.

The user has requested enhancement of the downloaded file.

A Survey of the Knapsack Problem

Maram Assi and Ramzi A. Haraty
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: {maram.assi, rharaty@lau.edu.lb}

***Abstract*--The Knapsack Problem (KP) is one of the most studied combinatorial problems. There are many variations of the problem along with many real life applications. KP seeks to select some of the available items with the maximal total weight in a way that does not exceed a given maximum limit *L*. Knapsack problems have been used to tackle real life problem belonging to a variety of fields including cryptography and applied mathematics. In this paper, we consider the different instances of Knapsack Problem along with its applications and various approaches to solve the problem.**

Keywords--Knapsack Problem; NP-hard; dynamic programming; branch and bound; greedy approach.

I. INTRODUCTION

The Knapsack Problem is composed of *n* items, each with its associated weight and profit. The objective is to select a subset of the *n* items that maximizes the profit and that at the same time fits the knapsack. The name of the problem is derived from the situation in which a mountain climber should select a number of items to be included in his knapsack and that he might use during his trip. The difficulty here is that the climber should select items with more profit but in the same time should not exceed the capacity of the knapsack. Scientists have shown interest in this problem because it represents different practical real life situations. In fact, these *n* items could represent n computed data files that need to be stored and we know that there is only W bytes storage available. Therefore the objective is to choose a subset of the files such that the total size does not exceed weight *W* and the stored files' total computing time is maximized. However, the family of Knapsack problems is rich and there are many knapsack variations. In the simple 0-1 Knapsack version, each item can only be picked once. When items should be selected from disjoint sets, then we are encountered with the multiple choice knapsack problems. Another type of knapsack is when there are many knapsacks that can be filled simultaneously.

978-1-7281-0385-3/18/$31.00 ©2018 IEEE

This paper discusses the different types of Knapsack problems and it highlights the theoretical and practical use of this problem.

The remainder of the paper is organized as follows. Section 2 covers the literature review. The binary knapsack problem is explained in section 3. In section 4, we introduce the different types of Knapsack Problem. And in section 5, we conclude the paper.

II. LITERATURE REVIEW

The Knapsack problem is known to be NP and a hard optimization problem. It cannot be solved in polynomial time by a deterministic algorithm. However, dynamic programing can be used to find a solution in pseudo-polynomial time as Martello and Toth explain in their book [1]. The problem dates back to 1897 and has been considered for more than a century [2].

The first solution that comes to mind when describing the simple version of the problem is to try all the combinations, enumerating all the search space solution. However, this exhaustive enumeration tries all 2^n possible subsets where *n* is the total number of items. We will have to try all possible assignment of ones and zeros in the vector representing the solution. If $n = 60$ and supposing a computer can run one billion vectors in 1 sec, then 2^n vectors will require more than 30 years to be computed. This shows that when n becomes large, running this brute force algorithm becomes too expensive, and this is explained by the fact that if number of items is increased by one, the total number of possible subset of items is doubled. In 1954, Bellman introduced the first algorithm using dynamic programming approach to solve the 0-1 knapsack problem [3]. In the sixties, Gilmore and Gomory [4] showed interest in dynamic programming solutions to other versions of knapsack. Kolesar in [5] introduced the first branch and bound solution for this problem since 1967. The algorithm can obtain either approximate or optimal solutions. Another alternative to find a solution to the problem is to adopt a greedy approach. However, greedy approaches cannot always guarantee an optimal solution.

Knapsack problems appear in actual life decision making problems specified by a resource allocation and a cost constraint. It is a combinatorial optimization problem where having a finite set of objects; we seek to find an optimal solution. The knapsack problem is classified as one of Karp's 21 NP-complete problems. Knapsack was shown to be NP-complete by reducing exact cover to Knapsack.

In 1976, Diffie and Helman proposed the knapsack problem as first scheme to public key encryption systems [6]. In 1978, Merkle and Helman investigated the problem thoroughly also. However, this approach for encryption was disregarded when it's been proved that it was not secure enough.

III. BINARY KNAPSCAK

The most common version of knapsack problems is the 0-1 knapsack known as binary variant. Consider we have n items and each item j has value v_j and a weight w_j. Let x_j be the number of copies of the selected item j. In the binary variant of the problem, each item is either selected once or not. Hence, x_j can have the value of zero or one. Consider that W is the maximum weight that can be carried in the bag and that the weights and the values of the items are non-negative, the problem is given by:

Maximize,

$$\sum_{j=1}^{n} v_j x_j$$

constrained by,

$$\sum_{j=1}^{n} w_j x_j \leq \mathrm{W}$$

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

The objective is to maximize the sum of the values of the selected items without surpassing the limit of the knapsack, in this case W. Consider the following decision making situation: a stockholder possessing a capital of x dollars and wishing to invest in some of the v possible investments. Knowing the expected profit and the cost of each investment, the problem can be referred to a knapsack problem. The goal is to maximize the investment contribution without surpassing the available resources. Another real life application of the binary knapsack problem is the problem of loading shipping containers. Knowing in advance the weight or the volume, the constraint can be that the space is scarce.

While the brute force approach to solve the knapsack problem with n items takes $O(2^n)$, a better running time can be obtained using dynamic programming. Dynamic programming is an approach for solving optimization problems. The main idea behind it is dividing the problem into a set of simpler sub-problems. In order to solve the problem, we first solve the sub-problems, and then combine obtained solutions to form the overall solution. Below is the pseudo-code of the dynamic programming algorithm to solve binary Knapsack:

```
Knapsack
{
  for (w = 0 to W)
      V[0, w]=0;
  for(j = 1 to n )
     for (w = 0 to W)
       if (w[j] <= w)
          V[j,w]=max{V[j -1, w],v[j] + V[j -1, w-w[j]]}
       else
          V[j, w]= V[j -1, w]
 return V[n.W];

}
```

The time complexity is clearly improved and is on $O(nW)$.

IV. KNAPSACK VARIANTS

The family of knapsack is composed of several variants of the classical problem. We assume that the profit, weight and capacity of the knapsack are positive integers. The constraints that lead to a new formulation of the problem include the number of times an item can be included in the knapsack, the number of available knapsacks, and if the item is to be included entirely or it is possible to include portions of it.

The bounded knapsack (BKP) is a modified version of the binary knapsack. While in the later variant each item can be chosen at most once, the bounded version of the problem allows each item to be included a multiple of times. m_j is the upper bound on the availability of items of a specific type j. We can notice that BKP can be considered a generalization of the binary knapsack where $m_j = 1$ for all j. Martello and Toth [7] explain that solution for the BKP can be obtained by transforming the problem to the 0-1 version and apply one of the algorithms used to solve the binary version. The maximization problem can be expressed as:

$$\sum_{j=1}^{n} w_j x_j$$

Subject to

$$\sum_{j=1}^{n} w_j x_j \leq \mathrm{W}$$

where $x_j \in \{0, 1, \ldots, m_j\}$

A more general variant of the above problem is the unbounded knapsack (UKP) in which an unlimited number of each type of items can be added. In fact, even though an unlimited number of each item type can be used, the number of items used is still bounded by the capacity of the knapsack. So a solution for the BKP can be used for the UKP by setting the upper bound b_j of the item *j* equal to the maximum amount of items of type *j* allowed in the knapsack relatively to the capacity (b_j= *W*/ b_j). Recursive solutions cannot be adopted because there are an unlimited number of items. Branch and bound and dynamic programming can be used to solve the UKP. The general formula of the UKP is:

Maximize

$$\sum_{i=1}^{n} v_j x_j$$

subject to

$$\sum_{j=1}^{n} w_j x_j \leq \mathrm{W}$$

where $x_j \geq 0$ (integer)

Another instance of the knapsack problem is the fractional knapsack (FKP). This type of knapsack is also known as continuous. In this variant of the problem, fractions or pieces of the item can be chosen and not necessarily the whole material is selected. The formula that summarizes the problem is:

Maximize

$$\sum_{j=1}^{n} v_j x_j$$

subject to

$$\sum_{j=1}^{n} w_j x_j \leq \mathrm{W}$$

where $1 \geq x_j \geq 0$

While the classical version of the problem is NP-hard, this variant of the problem can be solved in polynomial time. Let *R* be the profit/weight ratio:

$$R = \{\frac{v_1}{w_1}, \ldots, \frac{v_n}{w_n}\}$$

Ratios are sorted in decreasing order using one of the sorting procedures in $O(n\log n)$. An optimal solution to the problem can be obtained by adopting a greedy approach. We start by adding to the knapsack items having the largest ratio. We keep on adding the entire portion of the item as long as the limit capacity is not reached. If we reach a point where the entire material can't fit in the bag, then we select only fraction of that the next item and the solution is reached. As mentioned above and because of the sorting step, the problem takes $O(n\log n)$. However thanks to an algorithm proposed by Bernhard and Jens in [8] that computes the weighted median, FKP can be solved in polynomial time $O(n)$. FKP problems have been used to solve several resource sharing problems. An example of simple problem that uses FKP is: Suppose you are in a town that contains several gas stations. You have to fill your tank to reach another city. Unfortunately, all the station is almost out of gas. Moreover, each station has a specific price for the gasoline. Knowing the cost of gasoline in each station along with the quantity available, the minimization problem of filling your tank with a minimal total cost can be formulated using a FKP. The items represent the gasoline and the gas tank the knapsack.

If we consider that the weight (v_j) and the corresponding profit (v_j) of each item *j* are equal, then a new version of Knapsack problem arises. The goal of the problem becomes to compute the subset of weights that maximizes the sum without exceeding the capacity *W* of the knapsack. The problem is known as the Subset-Sum problem, Value independent Knapsack problem or Stick stacking problem. According to Martelo and Toth [7], the subset sum problem may solve the two-processor scheduling problem. The later problem can be described as having m identical machines (in this case two) and a fixed number of jobs say n where each job has a processing time and the goal is to assign the processes to the machines in a way that minimizes the maximum load . Subset-Sum problem can be formulated as:

Maximize

$$\sum_{j=1}^{n} w_j x_j$$

subject to

$$\sum_{j=1}^{n} w_j x_j \leq \mathrm{W}$$

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

An additional generalization of the 0-1 knapsack is the Multiple Knapsack Problem. In this instance of the problem, there are m containers instead of only one and each having a capacity c_j. A new variable $x_{ij}\{0, 1\}$ is introduced and that indicates if item *j* is contained in container *i* or not. The decision is not only whether a specific item will be included or not but also to which bag it should be added to. Multiple processor scheduling is one application of this problem. In addition, task allocation among autonomous agents, continuous double-call auctions [9], vehicle/container loading [10] can also be mentioned. The new formulation of the problem will be:

Maximize

$$\sum_{i=1}^{m} \sum_{j=1}^{n} v_i \, x_{ij}$$

subject to

$$\sum_{j=1}^{n} w_j \, x_{ij} \leq c_j$$

$$\sum_{i=1}^{m} x_{ij} \leq 1$$

where $x_{ij} \in \{0,1\}$

Given a budget and possibly additional constraints, the Transportation programming is the process of choosing projects for funding. Multiple choice knapsack problems can be used to find an optimal solution to the former mentioned problem. The mentioned variation of the classical knapsack has also other real application including production scheduling. In the Multiple choice knapsack, the items are divided into different classes and one item should be selected from each class It. is expressed as:

Maximize

$$\sum_{i=1}^{m} \sum_{j=1}^{ni} v_{ij} x_{ij}$$

subject to

$$\sum_{i=1}^{m} \sum_{j=1}^{ni} w_{ij} x_{ij} \leq W$$

$$\sum_{j=1}^{ni} x_{ij} = 1$$

where $x^{ij} \in \{0,1\}$

The 0-1 knapsack problem can be a special case of multiple constraint knapsack problem with only one constraint taken into consideration. The 0-1 multi-constrained knapsack problem occurs when there are multiple constraints to be taken into consideration, for example weight limit and the volume limit. It can be formulated as follows:

Maximize

$$\sum_{j=1}^{n} v_j x_j$$

subject to

$$\sum_{i=1}^{n} w_{ij} x_j \leq W_j$$

where $x_j \in \{0,1\}$ and W_j is the limit of a specific constraint.

For example, in the field of economy, a real application could be the following. Consider we have a set of project (*n* items), and a set of m constraints that are in this case the resources. Each of these project has a profit v_j and a resource consumption W_{ij}. The aim is to select a subset of the *n* projects that maximizes the profit without exceeding the corresponding resource limit b_j. Regarding this variant of knapsack, exact algorithms and heuristic methods has been developed to solve the problem. Exact algorithms include dynamic programming methods, branch and bound, and systematic approach. In the case of large instance problems, various heuristic optimization techniques are adopted to reach an optimal solution including simulated annealing, Tabu search and genetic algorithms.

One of the complex variants of the KP is the Multidimensional Multiple choice Knapsack Problem (MMKP). Given a number of classes *m*, each class i contains n_i items. Within the class *i*, the profit of an object *j* is v_{ij}.

Having l dimensions of weight, the weight of the object *j* at dimension *k* is W_{ijk}. On each dimension, the knapsack has a corresponding capacity W_k. The objective is to select one object of each class in order to maximize the total profit. However, the total weight of each dimension should not exceed the corresponding limit of each corresponding knapsack. This variant of knapsack has several real life applications from which we can mention: management of Quality of service in the field of computer network [11], and some resource allocation problems. The quality of service problem describes the features of a network that aim to provide the best performance required for each type of application. These features can include link topology, network design, etc. The formulation is:

Maximize

$$\sum_{i=1}^{m} \sum_{j=1}^{ni} v_{ij} x_{ij}$$

subject to

$$\sum_{i=1}^{m} \sum_{j=1}^{ni} w_{ijk} x_{ij} \leq W_{kj} \ , k = 1, \dots, l$$

$$\sum_{j=1}^{ni} x_{ij} = 1$$

where $x_{ij} \in \{0, 1\}$

In some variants, when the objective function is quadratic, a new type of knapsack takes place, known as Quadratic Knapsack Problem and can be formulated as follows:

Maximize x_T Qx,

such that

$$\text{ax} \leq W, \quad \text{x} \in B_n$$

or minimize x_T Qx,

such that

$$\text{ax} \geq W, \quad \text{x} \in B_n$$

where *Q* is *n* x *n* non-negative matrix, a represents a positive n-vector, B_n the set of all 0-1 n vectors. One example based on this definition of the problem is the decision version of the clique problem. This problem is a special case of the maximization version. In fact, *Q* can be the adjacency matrix, a_{i-j} for all *i* and *W* = *k*. The clique exists if and only if the optimum value obtained is *k*(*k*-1).

The online knapsack problem, firstly studied by Vercellis and Marchetti-Spaccamela in [12], is a variant in which we do not know all the items in advance. Items are given one after the other and upon the arrival of an item, one should immediately decide to pack it or not. Different versions of the problem exist. Some versions of the problem allow the removal of an added item; some are known as weighted and some other as unweighted. In the unweighted version known also as uniform the profit of the item is equal to its size. As an example of real world application of online knapsack we can mention the keyword online auctions. Known search engine including Google and Yahoo, sell to advertiser advertising positions in the results of the search. A number of bidders bid for a specific keyword and bidders are allowed to revise their bids dynamically. With each click, the bidder associates a value *v* (profit of the keyword). *W* is the budget over a specific period of time.

There exist a set of knapsack like problems including Nested knapsack, Collapsing knapsack, Non-linear knapsack and Inverse-parametric knapsack.

V. CONCLUSION

The knapsack problem is one of the most commonly used studied problems. The classical knapsack is the 0-1 knapsack known as binary. Many variant of the knapsack problem exist. In this paper, we discussed the different types of Knapsack problems along with possible method to solve some of them. Moreover, the various applications of the different variants are explored.

ACKNOWLEDGEMENTS

This work as supported by the Lebanese American University – Beirut, Lebanon.

REFERENCES

[1] S. Martello, and P. Toth, "A mixture of dynamic programming and branch-and-bound for the subset-sum problem", Management Science, 30, pp. 765-771, 1984.

[2] G. Mathews, "On the partition of numbers", Proceedings of the London Mathematical Society, Vol. 28, pp. 486-490, 1997.

[3] R. Bellman, "On the theory of dynamic programming", Proceedings of the National Academy of Science, USA, 38, pp. 716–719, 1952.

[4] P. C. Gilmore and R. E. Gomory, "The theory and computation of knapsack functions", Operations Research, 14, pp. 1045–1075, 1966.

[5] P. J. Kolesar, "A branch and bound algorithm for knapsack problem", Management Science, 13, pp. 723-735, 1967.

[6] W. Diffie and M. Helman, “New directions in cryptography”, IEEE Transactions on Information Theory 22(6): pp. 644–654, 1976.

[7] S. Martello and P. Toth, Knapsack problems: Algorithms and computer implementations [M]. John Wiley & Sons, 1990.

[8] K. Bernhard and Y. Jens, “Fractional Knapsack and weighted median”, Combinatorial Optimization: Theory and Algorithms, Algorithms and Combinatorics 21, Springer, pp. 459–461, ISBN 9783642244889, 2012.

[9] J. R. Kalagnaman, A. J. Davenport and H. S. Lee, “Computational aspects of clearing continuous call double auctions with assignment constraint and invisible demand”, Electronic Comer Research, 1 pp. 221-238, 2001.

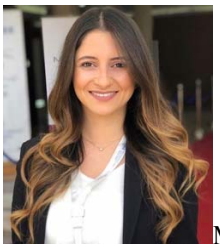
[10] S. Eilon and N. Christofides, “The Loading Problem”, Management Science, Vol. 17, No. 5, Theory Series, pp. 259-268, 1971.

[11] C. Lee, J. R. Lehoczky, R. Rajkumar, and D. Siewiorek, “On quality of service optimization with discrete QoS options”, in: RTAS’99: Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium, IEEE Computer Society, Washington, DC, USA, 1999.

[12] A. Marchetti-Spaccamela and C. Vercellis, “Stochastic on-line knapsack problems”, Mathematical Programming, 68: pp. 73–104, 1995.

AUTHORS

Ramzi A. Haraty is an associate professor of Computer Science in the Department of Computer Science and Mathematics at the Lebanese American University in Beirut, Lebanon. He is the chairperson of the Arab Computer society. He is a program evaluator (PEV) for ABET. He is also the capstone projects coordinator for MEPI’s Tomorrow’s Leader program. He received his B.S. and M.S. degrees in Computer Science from Minnesota State University - Mankato, Minnesota, and his Ph.D. in Computer Science from North Dakota State University - Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 110 books, book chapters, journal and conference paper publications. He supervised over 110 dissertations, theses and capstone projects. He is a member of the Association of Computing Machinery, Institute of Electronics, Information and Communication Engineers, and the International Society for Computers and Their Applications.



Maram Assi is software analyst at Acteos Production – SAL Offshore. She received her B.S. and M.S. degrees in Computer Science at the Lebanese American University - Beirut, Lebanon. Her research interests include genetic algorithms, optimization methods, and smart grid computing. She has two conference paper publications.

View publication stats