The Three-dimensional Knapsack Problem

Optimizing the utility of cargo space

Date: 24 January 2018

Project Phase: 1.3

Group Number: 12

Group Coordinator: Carlo Galuzzi

Group members: Andrei Atomei, Matteo Anzidei, Philippe Jacob,

Ramses Kamanda, Amanda Kane

### Abstract

Finding the optimal way to utilise cargo space is essential to a logistics company. The challenge that is encountered, can be generalised into a three-dimensional knapsack problem. This report focuses on describing and evaluating the greedy algorithm, the backtracking algorithm, a linear programming approach as well as an innovative approach that combines combinatorial optimisation with a backtracking algorithm, all of which can be used to solve this kind of problem.

### Keywords

Three-dimensional Knapsack Problem, Greedy Algorithm, Dancing Links (DLX), Linear Programming, Combinatorics

## Contents

[Abstract](#_o17nemob6fq1) 1

[Keywords](#_yy84e2csupw1) 1

[Contents](#_ouwx65qvcuz6) 2

[**1.Introduction**](#_hl7t19dtq24v) **3**

[**2. Methodology**](#_3j5i9p2qnrbo) **5**

[2.1. Greedy Algorithm](#_l3v89u683myj) 5

[2.2. Recursive and backtracking algorithm](#_j2d8rlpa0pkx) 5

[2.3. Linear Programming combined with a backtracking algorithm](#_cu3aempp3o0o) 6

[2.4 Combinatorial Optimisation combined with a backtracking algorithm](#_9jmlanmh3yzm) 7

[2.5 Software Architecture](#_3dlrlwe3q9uj) 8

[**3. Experiments**](#_cey93x1u7qjv) **12**

[Question A:](#_3vaxo2fljywk) 13

[Question B:](#_5vzjuvnmlqzq) 13

[Question C:](#_wqattzjl0m7c) 13

[Question D:](#_lip539ax7bu4) 14

[**4. Results**](#_n2q3tm7g0myb) **15**

[**5. Expectations, Comparisons and Evaluations**](#_qn7jykr81d7) **15**

[5.1 The Greedy Algorithm](#_uuzdyi1jgwfs) 15

[5.2 Expectations related to the recursive backtracking algorithm](#_ipzorwj6si2o) 15

[5.3 Expectations about the Linear Programming Approach](#_q83az5st8jln) 16

[5.4 Expectations about the Combinatorial Optimisation Approach](#_8aad2g2a5g9r) 16

[**6. Conclusion**](#_bj86r32mqcjs) **16**

[**7.References**](#_2538kospsznx) **17**

[Further Reading](#_88i6asz3tjwn) 18

# 1.Introduction

Cargo space optimisation is a common challenge encountered by logistics enterprises. It consists of organising and loading the available cargo space in the most efficient way by using mathematical models that take into account a given number of formalised constraints. Maximising the utility of their cargo space is essential to the economic exploitations of transportation and shipping companies. Improper loading of cargo space results in negative effects on the company’s profits as well as on the payload and on employees. A logistics company can overcome and control these limitations in order to maximize its profits and efficiency, by using computer-assisted optimisation methods. (Limbourg, Schyns and Laporte, 2011)[1]. The difficulty of this task increases as the parcels that are shipped in a cargo space of a certain size, differ in their attributed values and shapes.

This report aims to solve a three-dimensional knapsack packing problem and illustrate methods that can be implemented to reach an optimal solution. A three-dimensional knapsack problem, consists of fitting a given number of items with certain dimensions and associated values into a given space (Martello, Pisinger and Vigo, 2000)[2]. More precisely, this project report illustrates how a logistics company can load parcel boxes of two different sets into a truck with the dimensions (16.5m x 4.0m x 2.5m) and maximise the total value of the fitted cargo while maximising the occupied space of the cargo space. The boxes of the first set have the following attributes; A (1.0m x 1.0m x 2.0m), B (1.0m x 1.5m x 2.0m) and C(1.5m x 1.5m x 1.5m) with values of 3, 4 and 5 units respectively. The second set contains three pentomino shaped parcels (L, P and T) that are formed by five cubes of size 0.5m x 0.5m x 0.5m. The task at hand is essentially a knapsack problem. “We are given n items and a container (knapsack). Each item j, j {1..., n}, has a profit pj and two types of weight: wj and uj” (Martello and Toth, 2003) [3]. Where in this case, wj and uj are the dimensions and volume of the parcels and pj is the total value of the cargo.

This report starts with giving a description and explanation of the methods and approaches that are commonly used to solve three-dimensional knapsack problems. Namely; the greedy algorithm, backtracking algorithms and the linear programming approach. In addition to these, it illustrates an innovative approach using a mixture of combinatorial optimisation and the greedy algorithm, that this project group has come up with.

Following the descriptions of the different methods, this report will give results for the experiments that aim to test which one of these is most suitable and efficient to solve the knapsack problem at hand.

The fourth chapter, gives graphical representations of the computational results obtained from these experiments.

These are then compared and evaluated in the next part of the report to see which one yields the best approximation to an optimal solution.

Finally, it concludes by giving the most efficient method to solve the knapsack problem at hand.

Additionally to this written report, a user-friendly computer application, based on the most efficient method to solve this three-dimensional knapsack problem has been designed and implemented. The application finds and gives a three-dimensional representation of the optimal solution to the task.

# 2. Methodology

## 2.1 Greedy Algorithm

The greedy algorithm was designed to solve optimisation problems by either maximising or minimising a specific objective function in every step that it takes so as to reach global optimum. Cormen, Leiserson and Rivest, (1990) further state that because the algorithm tries to find a local optimum at every step, it is unlikely that it will be able to reach a global optimal solution. However, they believe that the greedy heuristic of this method allows to yield locally optimal solutions, that approximate in a reasonable time, to the global optimum. [5]

To solve the knapsack problem at hand, the following steps are taken in order to maximise the total value of the cargo. First the parcels will be sorted according to highest value, the algorithm will then pack the parcels with the highest value first, so as to optimise the total value of the cargo. It will continue to take the parcel with the highest value in every step until the truck has reached maximum capacity. If at some point the parcel does not fit in any configuration in the truck the algorithm will go on to pick the box with the next highest value and checks whether this will fit into the truck. This step is repeated until it has tried and exhausted all blocks in decreasing value order. See appendix 1 for the pseudocode.

## 2.2 Recursive and backtracking algorithm

Knuth (1968) defines the backtracking algorithm as a general algorithm to optimisation problems such as the knapsack problem, by “incrementally building candidates to the solution and abondonning each partial candidate”, through backtracking when it finds that the “candidate does not complete to a valid solution”. The so called partial candidates are in the case of the three-dimensional knapsack problem at hand the parcel boxes that are supposed to be fitted into the cargo space of a truck. It is further claimed that the backtracking algorithm is likely to be much faster to solve for such a problem than a method based on brute force. This claim is based on the algorithm’s ability to “eliminate a large number of candidates” that cannot be included to find an optimal solution with a single test. [7]

The recursive and backtracking algorithm that solves the given problem, first checks whether or not the truck is fully loaded. This is the case when the entire space is filled. If this is not the case, it chooses the box with the highest value and tries to place it into the first empty space that it can find. If the box fits into the empty space, the method repeats this first step. If the box does not fit, it will first rotate the parcel and try to fit it into the space until it has tried every available rotation. In the case where the box does not fit even after trying with all rotations, the method tries to fit in the next available parcel. If none of the available parcels fit, the algorithm goes back one iteration, and tries a new order of fitting the parcels into the truck.

Nevertheless, the traditional backtracking algorithm attempts to find a solution with respect to 3D space, which is fairly demanding and likely to fail given the number of possibilities that have to be taken in consideration. As a result, another version of the algorithm was considered along, which is an adaptation of Knuth`s backtracking algorithm: the code divides the truck into layers of two-dimensional blocks, for which it finds the smallest divisor array of a two-dimensional array for a layer and tries to find a solution for it using a two-dimensional version of the backtracking algorithm. If no solution is found, then the code will look for another divisor array, and so on. If the algorithm cannot find a solution for any divisor of a layer, it will try another side of the truck and solve with the layers from the new side. If the code finds a solution, it fills the cargo with the solution (which is possible due to the fact that the array is a divisor of the cargo) and returns its outcome as an answer. The conventional name used for such a technique was researched to be “Shelf-sorting”.

## 2.3 Linear Programming combined with a backtracking algorithm

The requirements for the three-dimensional knapsack problem can be represented by linear relationships. To achieve the best outcome, in this case the highest total value of boxes that fit into the cargo space of a truck, one can use an optimization method called linear programming (Bonizzi, 2017) [4]. Bonizzi further states, that to reach the optimal solution among all feasible alternatives, which are based on a mathematical model, the linear programming method requires an objective optimization function that is subject to linear constraints. In the case of the three-dimensional knapsack problem at hand, the objective function will look as follows;

Where (Z) stands for the total value, (valueA,B and C ) stand for the respective values of the three different boxes type A,B and C and (x1, 2 and 3) stand for the respective number of boxes placed into the cargo space. For the problem at hand, we have two types of constraints. At first, the constraints that are based on the respective volume of the available quantity for each box type;

Where (volumeA,B and C) stand for the volume of each respective type of box and (givenA, B and C) stand for the number of boxes that are given for each box type respectively. The second constraint that one should take into account is based on the volume of the cargo space of the truck which in this case is 165 m3;

Once, the problem is formalized in such a mathematical model, Bonizzi recommends to use the simplex method to solve for the optimal solution. This optimal solution which is based on the volumes of the boxes and the truck as well as the total value, is then passed over to the recursive and backtracking algorithm, which finds the optimal way to fit the boxes into the cargo space based on their shapes.

## 2.4 Combinatorial Optimisation combined with a backtracking algorithm

An innovative approach that allows to solve the three-dimensional knapsack problem at hand, is to create a set of possible combinations of boxes that are then passed to a backtracking algorithm, that is an adaptation of the Dancing Links Algorithm to solve for the optimal fit. More specifically, this approach starts by using a brute force method that considers the given quantity of boxes for each box type and creates upon this information a set of optimal combinations. At this level of the approach, a combination is considered optimal when the combined volume of all the boxes of the combination at hand are smaller or equal to the total available volume of the cargo space. The obtained set of possible combinations is then ordered into a list, starting with the combination that gives the highest total cargo value and is closest to the maximum available cargo space volume. In a further step, the method passes the list to the greedy algorithm, which starts with the highest ranked combination and tests whether or not it is possible to fit it into the truck. The solution is obtained once this is the case. If the tested combination does not fit, it is rejected and the greedy algorithm uses the next combination on the list. This last step is repeated until the an optimal combination that fits into the truck is found.

## 2.5 Software Architecture

This section of the report will discuss the design phase for the program. A Unified Modeling Language (UML) diagram is provided(Fig.1) to illustrate the relationships between the classes. This, along with the method documentation in the code, will allow any company to then design and implement the program.

The Interface is written using a JavaFX library, making use of the Box data structure, in addition, an original design is provided: with use of checkboxes, the user is able to remove and add back the pieces to see inside the truck (Figures 2,3,). The user is able to zoom in and out of the truck to view it further by holding the right-click mouse pad whilst scrolling up or down.

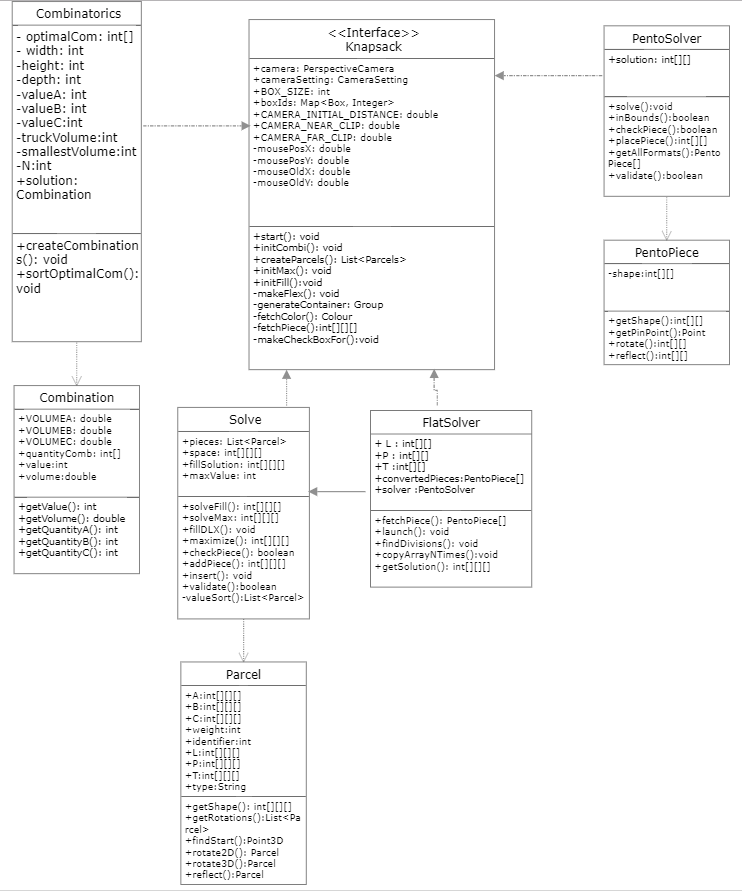


Fig 1. Class Diagram

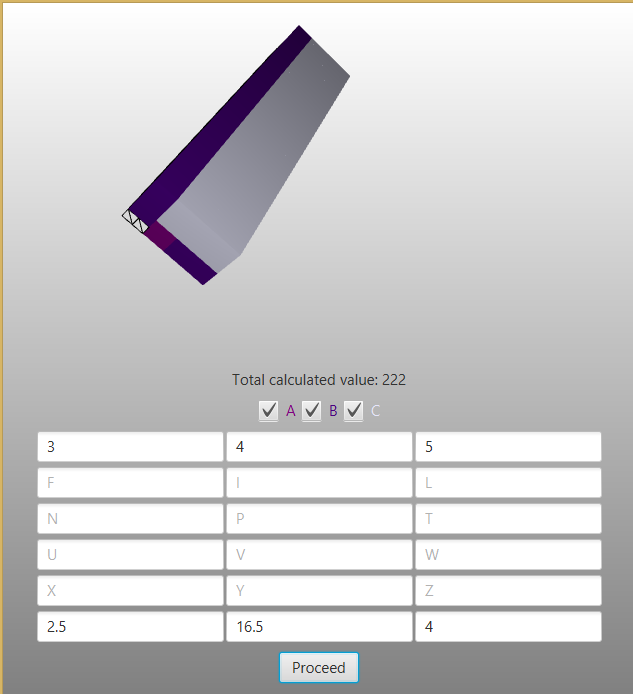


Fig.2. Representation of 3 Parcels, A;B;C with values 3,4,5 respectively in a truck of 2.5 width x 16.5 height x 4.0 depth, showing also spaces unfilled.

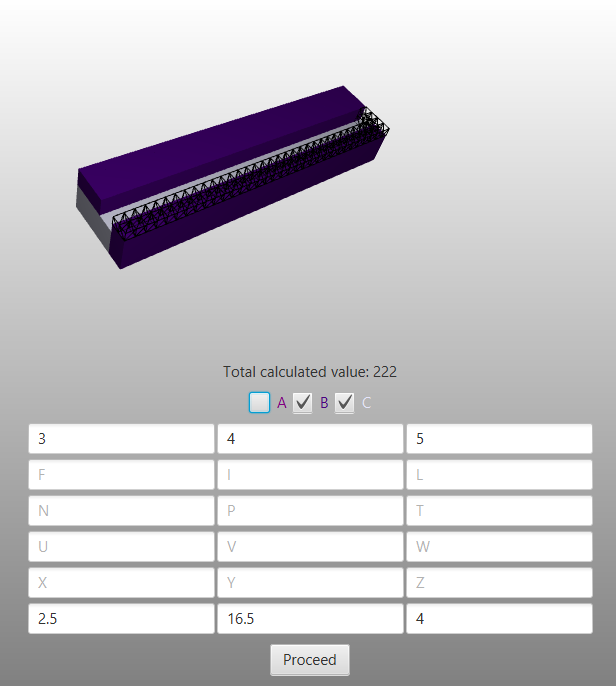


Fig.3 Representation of 3 Parcels same as in Fig 3, excluding the parcel A in the visualisation as it has been unchecked

# 3. Experiments

The first set of experiments in this report are guided towards answering two types of questions in order to draw conclusions on how to efficiently make use of a given space subject to conditions on the total size of the space, the shapes of the parcels and their value. Question one asks whether it is possible to fill the cargo space with three different types of parcels without having any gaps. Question two asks to find the maximum value that can be stored in the cargo space, based on the quantity of the provided parcels. Both questions will be answered subject to two different sets of parcels. Set 1:A,B,C and Set 2: pentominoes L,P,T.

The dimension for the parcels are; A(1.0 x 1.0 x 1.0), B(1.0 x 1.5 x 2.0) and C, 1.5 x 1.5 x 1.5). The pentominoes have the L, P and T shapes and are formed from five cubes with each cube having the dimensions 0.5 x 0.5 x 0.5. The cargo-space of the truck is 16.5 meters long, 4 meters high and 2.5 meters wide.

Question A: Is it possible to fill the complete cargo space with A,B and/or C parcels, without having any gaps? Here the backtrack algorithm and the greedy algorithm will be tested.

Question B: If parcels of type A,B and C represent values of 3, 4 and 5 units respectively, then what is the maximum value that you can store in your cargo-space? To find results for this a greedy algorithm will be used.

Question C: Is it possible to fill the complete cargo space with L,T and/or P parcels, without having any gaps? The Backtrack algorithm and shelf sort algorithm will be used here.

Question D:If parcels of type L,T and P represent values of 3, 4 and 5 units respectively, then what is the maximum value that you can store in your cargo-space? Implement the greedy algorithm.

# 4. Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parcels | A | B | C |  |
| Value | 3 | 4 | 5 |
| Maximum Value for Greedy Algorithm |  | | | 222 |

Shape 1. Question B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parcels | L | P | T |  |
| Value | 3 | 4 | 5 |
| Maximum Value for Greedy Algorithm |  | | | 1084 |

Shape 2. Question D

[COMBINATIONS] [VALUES] [RUNTIME] [TRUCK PROFIT]

[A] [A:1 B:2] [1121] [64]

[A-B] [A:1 B:2] [2106] [86]

[A-B-C] [A:1 B:2 C:3] [1018] [121]

[A-B-C] [A:2 B:1 C:3] [1026] [93]

[A-B-C] [A:3 B:2 C:1] [3096] [86]

Shape 3: The last table are the results of different Tests, using a variation of the Greedy algorithm.

# 5. Expectations, Comparisons and Evaluations

## 5.1 Expectations

### 5.1.1 Expectations towards the greedy algorithm

The Greedy Algorithm has a time complexity of with n representing the number of boxes and m representing the cargo truck. Based on this, it is expected to work fast as it is designed to pick the greediest choice in each stage without having to factor in any future problems.

### 5.1.2. Expectations towards the backtracking algorithm

It was expected that the backtracking algorithm is able to fill the cargo space entirely. It is also expected to be the fastest algorithm of the selection when combined with a divide and conquer algorithm. The precision is guaranteed by its ability to backtrack when it made a mistake and reverse it.

### 5.1.3 Expectations about the Linear Programming Approach

This approach only allows to pass one single solution to the backtracking method. This becomes problematic in the case where the optimal solution reached by the linear programming approach does not fit into the truck, such that the backtracking algorithm can not solve for a reasonable outcome.

### 5.1.4 Expectations about the Combinatorial Optimisation Approach

In order to avoid the problem that can emerge when the single optimal solution obtained by the linear programming approach, is not solvable by the backtracking method, one can replace it with the previously introduced combinatorial optimization method.

## 5.2 Booklet questions

### Question A:

*Is it possible to fill the complete cargo space with A,B and/or C parcels, without having any gaps?*

**Answer:** The algorithm used for filling with 3 dimensional shapes was a backtracking one that would attempt to place pieces with respect to the whole cargo, meaning that the number of permutations available would exceed the capacities of the program. Therefore, the algorithm used to answer this question could not give a concrete answer, as it fails to process any non-trivial configuration of the cargo

### Question B:

*If parcels of type A,B and C represent values of 3, 4 and 5 units respectively, then what is the maximum value that you can store in your cargo-space?*

**Answer:** The primary maximization algorithm that was used to address the question was a Greedy one. The results gave a maximum value of 222 for a cargo of size 16.5 x x 2.5 x 4.0.

### Question C:

*Is it possible to fill the complete cargo space with L,T and/or P parcels, without having any gaps?*

**Answer:** The initial algorithm that attempted to answer this question was the same algorithm that was used for Question A. Thus, the algorithm could not produce a solution. However, since pentomino pieces can be controlled in a 2d environment alone, the second approach used a Shelf-sort algorithm, i.e. the program would attempt to solve layers of the cargo, and if a solution is found, it would copy it for the rest of the layers. Following the second approach, the program could find a solution in which the cargo is filled by using only P pieces.

### Question D:

*If parcels of type L,T and P represent values of 3, 4 and 5 units respectively, then what is the maximum value that you can store in your cargo-space?*

**Answer:** Given the promising result achieved with the Greedy algorithm for Question B, the Greedy approach was applied to Question D as well. The results reached a maximum value of 1084 for a cargo space of size 4 x 2.5 x 16.

## 5.3 Discussions

Given that every question was approached with two algorithms, the variations targeting the optimal solutions were rather unstable, which is to be expected. The two main approaches, Greedy and Backtracking, could even be combined to find a filling option that also maximizes value, yet the two can barely run in polynomial time individually. The issue regarding Question A that made it unanswerable was that the problem could not be scaled down to a two-dimensional field, demanding a three-dimension consideration of the space, which brings along too much complexity for a normal utility algorithm to handle. Overall, the conducted experiments suggest that looking for an optimal solution is not quite practical, as the improvement of the result is not even nearly proportional with the amount of time and computational effort demanded.

# 6. Conclusion

In conclusion, it can be validated that there is no single algorithm that can address the two types of questions that are formalized in the three dimensional knapsack problem in polynomial time. However, it is possible to answer the questions with specialized algorithms that are adapted to each problem.

# 7.References

[1] Limbourg, S., Schyns, M. and Laporte, G. (2011). Automatic aircraft cargo load planning. *Journal of the Operational Research Society*, 63(9), pp.1271-1283.

[2]Martello, S., Pisinger, D. and Vigo, D. (2000). The Three-Dimensional Bin Packing Problem. *Operations Research*, 48(2), pp.256-267.

[3]Martello, S. and Toth, P. (2003). An Exact Algorithm for the Two-Constraint 0–1 Knapsack Problem. *Operations Research*, 51(5), pp.826-835.

[4]Cormen, Leiserson, and Rivest (1990). *Introduction to Algorithms, Chapter 17 "Greedy Algorithms"* p. 329

[5]Bonizzi Pietro (2017). Linear Programming. *Lecture 6 of Introduction to data science and knowledge engineering, pp.4-10.*

[6]Cormen, Leiserson, and Rivest (1990). *Introduction to Algorithms, Chapter 17 "Greedy Algorithms"* p. 329

[7] Knuth, D. (1968). *The art of computer programming*. Reading Mass.: Addison-Wesley Publ.

## Further Reading

[A]Fekete, S., Schepers, J., & van der Veen, J. (2007). An Exact Algorithm for Higher-Dimensional Orthogonal Packing. *Operations Research*, *55*(3), 569-587.

[B]Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research*, 141(2), pp.382-392.