Loading truck: optimization available space and cost of packages

Lukasz Joksch
Department of Electronics,
Wroclaw University of Technology
Wroclaw, Poland
E-mail:lukaszjok@gmail.com

Tomasz Kowalik
Department of Electronics,
Wroclaw University of Technology
Wroclaw, Poland
tomasz_kowalik@hotmail.com

Piotr Tazbir
Department of Electronics,
Wroclaw University of Technology
Wroclaw, Poland
E-mail: piter9901@gmail.com

***Abstract*—Every logistic company has to find the best way to load their trucks. Unfortunately this is complex problem and without using proper tools it could take a lot of valuable time or be solved in a bad way. In this article authors examine a problem of loading trucks with packages defined by weight, volume and cost. The problem analysis was made and four algorithms were implemented in order to solve this problem. The algorithms performance is evaluated by comparing each of them to brute force algorithm. It was compared the cost of load, filling available space and time of execution of algorithm. The authors examines which of the algorithms gives the best result in different cases.**

1. Introduction

In many companies there is some problem with optimization of loading goods. This is very important multifaceted trouble, which concerns spending of money, waste of time and human resources management.

1.1. Saving money

When headmasters and managers do not implement optimization algorithms, work of their employees is ineffective. This in turn is associated with rising industry cost. If managers introduced new solution, they would use better company resources. The main advantage is that we are able to waste a place in trucks - developed software gives optimal arrangement of packages or select which loads pack into truck or not. Thanks to this solution is possible to load up the lorry. Drivers drive less routes, so their chief spend less money on fuel and all maintenance costs of the car.

1.2. Faster work results

Another advantage is that the same work, which people were doing before implementing optimization algorithm, they do it in shorter time. This is due to the fact that they do not have to plan in real time position of next packages, because employees have generated list with optimal positions. What is more, optimization process is very efficient, because it is automated - once written application is used repeatedly.

1.3. Responsible use of human resources

The use of optimization work makes people more thoughtful, it reduces the effort in the process of loading trucks. People have more time to regeneration and rest between successive jobs.

2. General overview of knapsack problem

Knapsack problem, often also called rucksack problem, is the most popular question in optimization matter. The main objective is to pack a goods as much as is possible with maximal value and minimal weight. The knapsack problem is said to be a thief problem, because as he, we want to get the most valuable items and put it in limited space(knapsack).

2.1. Definition

Formal definition says that we have a knapsack with maximal capacity W and set N of elements $\{x_1, x_j, ..., x_N\}$, while each element has specified weight $(\mathbf{w})$ and value $(\mathbf{v})$.

In this algorithm three approach are distinguished:

1) 0-1 knapsack problem

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

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leqslant W \text{ and } x_i \in \{0, 1\} \tag{2}$$

2) Bounded knapsack problem (BKP)

$$\text{maximize} \sum_{i=1}^{n} v_i x_i \tag{3}$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leqslant W \text{ and } 0 \leqslant x_i \leqslant c \tag{4}$$

Where "c" is non-negative integer value.

3) Unbounded knapsack problem (UKP)

$$\text{maximize} \sum_{i=1}^{n} v_i x_i \tag{5}$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leqslant W \text{ and } x_i \geqslant 0 \tag{6}$$

3. Mathematical model of knapsack problem

We have a few criteria of benchmarks in this problem. Brute fore always gives us optimal solution, but it is very time-consuming. We have decided to choose brute force algorithm as a pattern in comparison others algorithms.

3.1. Greedy algorithm

In first step this algorithm sort descending elements in data set. Next, it puts this elements to the knapsack (starting from the biggest one). If the weight of element is bigger than rest of maximal loading, it is skipped and algorithm analyse other elements. Algorithm stops when maximal loading of knapsack is reached or when algorithm checks the last element from the data set.

3.2. Genetic algorithm

It works like this:

- generate random starting population containing N individuals;
- calculate adaptation function for each individual;
- repeat creating new generations until the stop condition is not reached:
  - selection (choosing individual for reproduction);
  - create new population containing N individuals using crossing and mutation;
  - rate new individuals by calculating their adaptation function

3.3. Dynamic programming

Knapsack problem can be solved in pseudopolynominal time using dynamic programming method. w_1, w_w, , w_n is weight of each element on c_1, c_2, , c_n is them values. Algorithm have to maximize the sum of values of elements in the way that their weight cannot be greater than W. When A(i) is the greatest possible value which can be received when the weight is not greater then "i", than $A(W)$ is the solution of the problem. $A(i)$ is recursion defined:

$$A(0) = 0 \tag{7}$$

$$A(i) = max\{c_j + A(i - w_j) : wj \leqslant i\} \tag{8}$$

3.4. Brute force algorithm

This method checked all possibilities to solve the problem using given data set. This algorithm always find the best solution. Brute force can only be used to compare with other algorithms in laboratory conditions because due to its computational complexity (2n) it last too much time to used it in daily work.

4. Research plan

First of all we got to know with optimization problem, different algorithms which could solve the problem and choose three of them. Next we were looking for publications which describe every chosen algorithm. The publications helped us to solve the problem and also made the final report from our research. In next step we distributed all tasks which were necessary to finish our project. At the same time we create schedule which helped us to do everything in time. Then we created data sets which were using in programme. After that we implemented the chosen algorithms. After that we made user friendly graphical user interface. We also implemented making graphs. Next we started to test our programme and we were fixing all bugs. Then we selected parameters of algorithms so as they are similar to real world. In next step we analyse all algorithms with the best settings and compare to the Brute Force Algorithm. At the end we presented all results in transparent way, expressed conclusion and created the report.

5. Research

We created three data sets to do the research:

- Easy contains 15 elements;
- Medium contains 5310 elements;
- Hard contains 16100 elements;

We used all 3 data sets to research three algorithms: Greedy, Genetic and Dynamic Programming. Brute Force method was tested only with the Easy data set, because for other

ones the time of execution was near to infinity. Each algorithm was tested without modifications (we checked results only for weight of packages and truck and only for volume packages and truck) and with modifications (we checked results for weight and volume of packages and truck).

6. Software solution

To make research environment easier and our work effective we wrote graphical software. We have chosen JAVA language and SWING library. JAVA is very popular programming language with huge community and support. What is more when we implemented code in this language, our software is available on different platforms - Windows, Linux, Mac OS etc. .

6.1. Program features

As a priority we set ourselves following assumptions:

- original idea and solutions;
- user-friendly graphical interface, which does not need learning manuals or taking part in training;
- loadings goods from .csv file - it is popular file format in a lot of external software;
- ability to select a researching algorithm from drop-down menu;
- transparent view of end results;
- ability to show graphs to compare results from all researches;
- all of data instance are be represent by id number;
- adjustable input parameters in all of algorithms.

6.2. Graphical interface

Below are screenshots programmed environment:

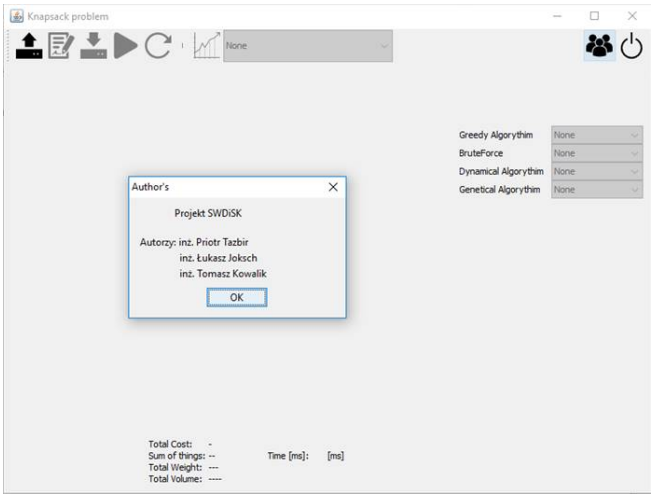

Figure 1. Main program view

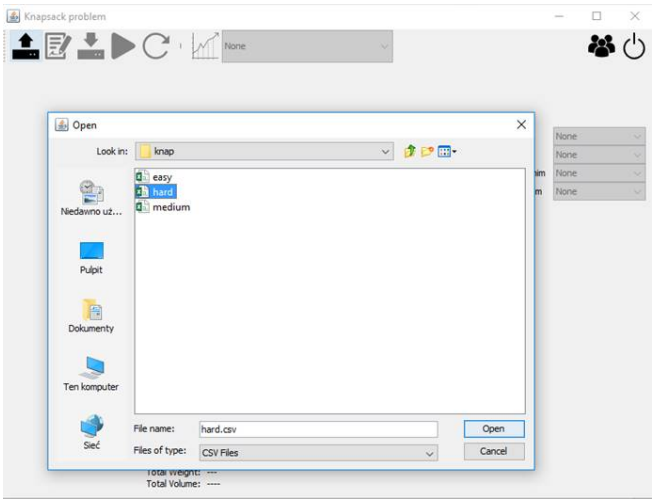

Figure 2. Window of opening files

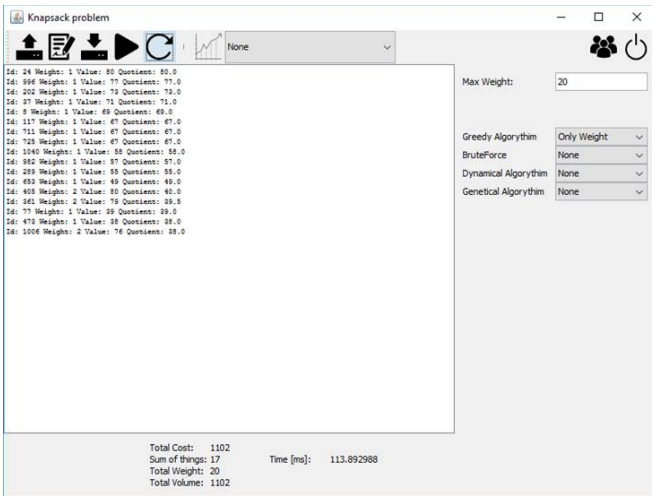

Figure 3. Program dashboard

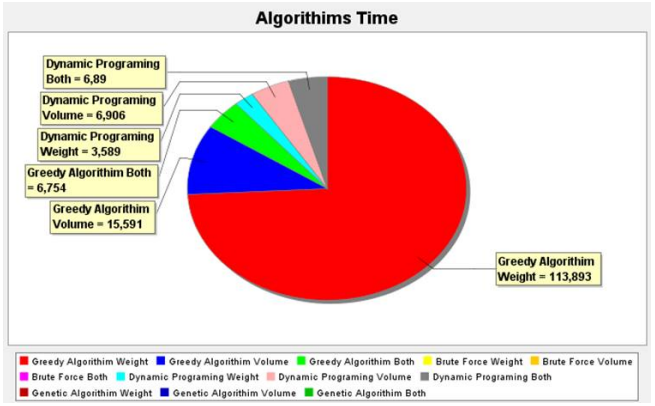

Figure 4. A sample graph

7. Results

As a result we got graphs, which present the degree of loading and average time of performing algorithm. Only selected charts are shown in this article. Only for easy set Brute Force is presented on charts, because calculation time is too long for bigger (more than 20 elements) datasets on personal computers. If we had more efficient computer or access to supercomputer cluster, we would try to solve this problem.

1) Easy set - 12 elements in excel file

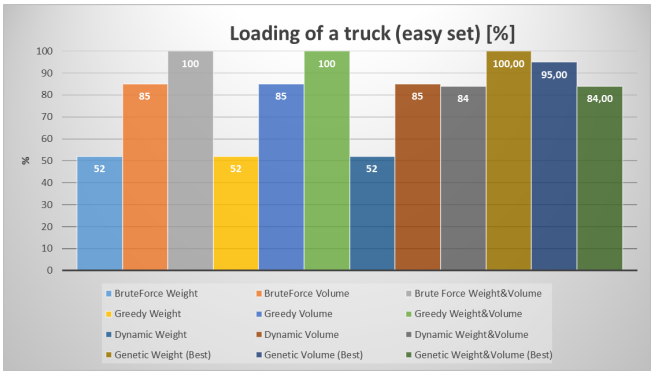

Figure 5. Percentage loading of a truck

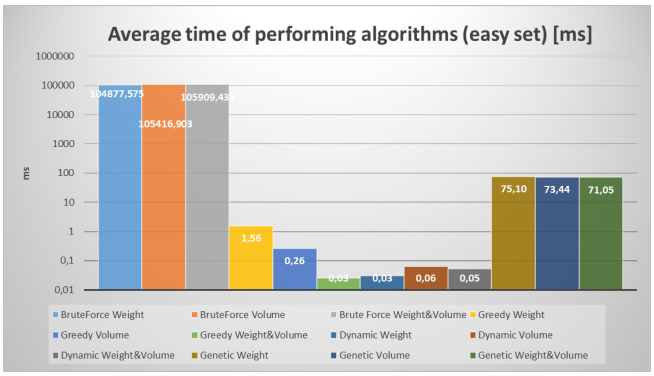

Figure 6. Average time of performing algorithms

As we can see on first diagram (Figure 1.) the best algorithm, as we take into account only weight or only volume of packages and truck, is Genetic Algorithm. In first case it found solution which loads a 100% of maximum loading of a truck. In second case it obtained 95%, which is very good result. However, if we want to consider both, the weight and the volume of packages and track at the same time, the best is Brute Force Algorithm and Greedy Algorithm. These algorithms obtained 100% of loading of the truck. If we take a look on the second diagram (Figure 2.) we can see that the fastest algorithm almost every time was Dynamic Programming. A little bit slower was Greedy Algorithm. As we were expecting the slowest algorithm was Brute Force Algorithm. Brute Force Algorithm tested with the smallest data set was executing almost 2 minutes while other algorithms was executing less than 1 second and usually even much more faster. The best (large fulfilment and high speed of execution) algorithm for the small data sets is Genetic Algorithm - if we look only on weight or only on volume, and Greedy Algorithm - if we want to consider these both things.

2) Medium set

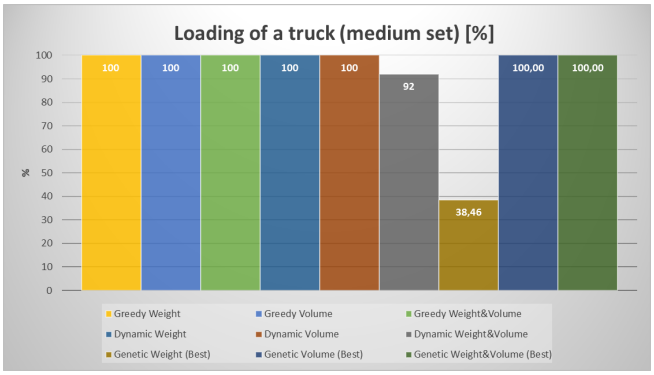

Figure 7. Percentage loading of a truck

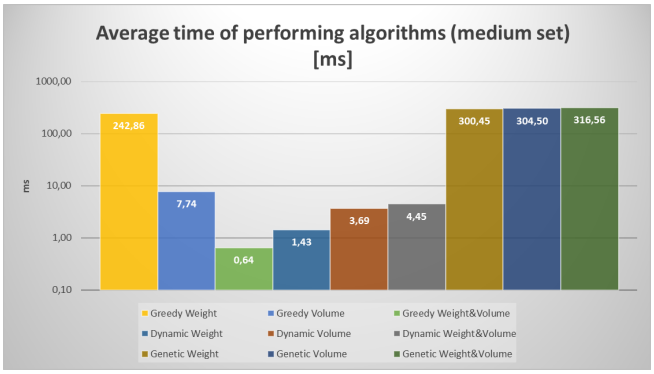

Figure 8. Average time of performing algorithms

As we can see on first diagram (Figure 3.) almost all algorithms give very well results. Only in two approach we can see anomaly: in Dynamic algorithm limited by both weight and volume and in Generic algorithm limited by weight.

Greedy algorithm limited by both weight and volume take the least time. Whereas 3. figure we agree that choosing this approach in this dataset could be the best solution.

3) **Hard set**

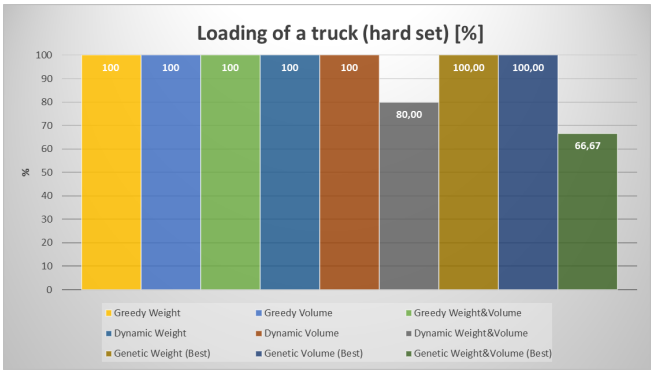

Figure 9. Percentage loading of a truck

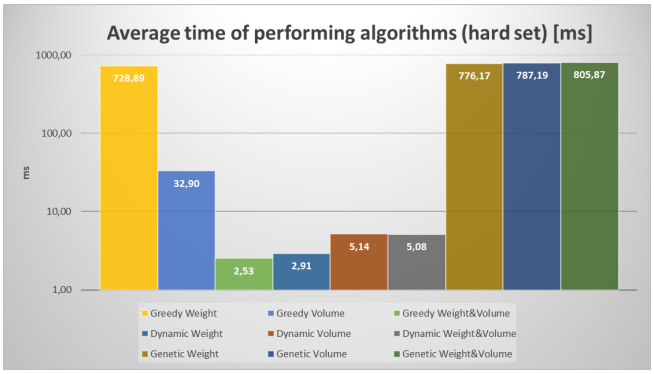

Figure 10. Average time of performing algorithms

As we can see on first diagram (Figure 5.) almost all algorithms give very well results. Again only in two approach we can see anomaly: in Dynamic algorithm limited by both weight and volume and in Generic algorithm limited by both weight and volume.

On the figure 5. we can see that Greedy algorithm limited by both weight and volume take the least time. Whereas 3. figure we agree that choosing this approach in this dataset could be the best solution.

8. Conclusion

- Applying our programme in transport companies brings lots of benefits in example: saving time to load a truck or maximize cost of each loading;
- After research each algorithm, we proved that the best one was ”Greedy Algorithm”, which we modified and it checked if each package was not too heavy (weight) or too big (volume). It was turning the best result of fulfilment a truck and also it was the fastest algorithm;
- After research each algorithm, we proved that the best one was ”Greedy Algorithm”, which we modified and it checked if each package was not too heavy (weight) or too big (volume). It was turning the best result of fulfilment a truck and also it was the fastest algorithm;
- The size of data set was mattered. The bigger set the longer was execution of each algorithm. We expected that kind of phenomenon;
- It was impossible to test ”Brute force Algorithm” using data set with lots of packages, because we didn’t dispose so big computing power;

Acknowledgments

We would like to thank our professors: Leszek Koszalka and Wojciech Kmiecik for following us and support in hard moments. We also have to thank our two friends: Mariusz Lont and Marek Wosko, who helped us to invent a subject of our research (loading trucks).

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Wprowadzenie do algorytmw.* Warszawa: Wydawnictwa Naukowo-Techniczne, 2003

[2] Michael R. Garey, David J. Johnson: *Computers and intractability: a guide to the theory of NP-completeness.* San Francisco: W.H. Freeman, 1979

[3] Antoni Niederlaski,*Programowanie w logice z ograniczeniami*, wydanie 2 poprawione, Gliwce 2014

[4] Richard Neapolitan, *Podstawy algorytmw z przykadami w C++*, Kumarss Naimipour, Helion 2004

[5] Jon Bentley, *Pereki oprogramowania / Pereki programowania*, WNT/Helion, 2011

[6] Piotr Wroblewski, *Algorytmy, struktury danych i techniki programowania.* , 3rd ed. Gliwice, Helion, 2003.

[7] Wikipedia article on day 14-08-2016, `https://en.wikipedia.org/wiki/Knapsack_problem`

[8] Wikipedia article on day 14-08-2016, `https://pl.wikipedia.org/wiki/Problem_plecakowy`