# Using hidden space in optimization of space utilization

Riste Marevski[1], Ivan Chorbev[1], Viktor Todorovski[1]
[1] Faculty of computer science and engineering, University of Ss Cyril and Methodius, "Rugjer Boshkovikj" 16, P.O. Box 393, 1000 Skopje,R. of Macedonia
riste.marevski@yahoo.com, ivan.chorbev@finki.ukim.mk, viktor.todorovski@yahoo.com

**Abstract.** The topic of this paper is the use of advanced algorithms in order to solve the problem of optimal use of available space. There are a lot of algorithms that try to solve this problem but most of them are not taking into consideration the available space into the concave elements. In this paper we describe how to use this space in order to find the optimal solution. Most of the algorithms that solve this problem use genetic algorithms as a base for the optimization. Some of them also use heuristics in order to implement expert knowledge. Our approach is based on an algorithm that groups the elements utilizing the available space from concave elements and then continues with the optimization phase. The optimization phase is implemented as a genetic algorithm that uses specific problem heuristics.

**Keywords: space utilization, transportation optimization, combinatorial optimization, packaging problem, cargo loading optimization, genetic algorithms, heuristics**

S. Markovski, M. Gusev (Editors): ICT Innovations 2012, Web Proceedings, ISSN 1857-7288

## 1. Introduction

Optimal use of space is a generic problem area that includes various problem subtypes that are being solved with varying efficiency and quality. A lot of researches try to solve problems of this type and there are a lot of proposed algorithms that ought to solve them. But the proposed algorithms are very often limited to simple geometric forms, usually rectangular or in some cases cylindrical forms [8]. Since these algorithms offer an optimal placement of elements with the mentioned limitations, our aim was to build an algorithm that can be used to place elements with complex shapes. Also, we aimed at utilizing the space inside the elements e.g. the holes in the elements is a space that is not taken into consideration. We refer to this space as "hidden" space since is not "visible" for the algorithms (not taken into consideration). Our approach is mainly focused on using this "hidden" space in order to produce an optimal solution.

We propose a two phase algorithm where in the first phase the elements are grouped, while the second phase is a somewhat standard loading optimization phase. The grouping phase of the solution that we propose can be incorporated in any optimization algorithm as an enhancement. When this phase is over, the algorithm can continue with an optimization phase. The second phase can implement any optimization algorithm, not limited to genetic algorithms, b-tree search etc. In this paper we present an application of an adapted genetic algorithm.

Researchers have looked for a solution to this problem for a long time in the past. Apparently, back in the sixties of the last century [4] the need for optimal utilization of space emerged as a topic of more serious research. Since then this problem is quite researched and thus a number of techniques for its solution are proposed. However, optimizing the utilization of space has remained a popular research topic even today. Proofs of the continued interest are the papers that are published in the last few years focusing on this subject [1,4,6,24]. The fact that researchers are still working on finding more appropriate and more complete solution to these problems suggests that this is an area where there is room for further improvement.

Optimization of space utilization finds application in areas such as transport of material goods, warehouse storage, packaging, etc. [5].

The rest of the paper is organized as follows: section 2 describes the previous research on this topic, the limitations of the proposed solutions and the areas that can be improved. Section 3 describes our algorithm in detail. Section 4 describes the experimental results, while in section 5 the conclusion of this research is stated. In section 6 we present the future work aims and the planned improvements.

## 2. Previous research

Several ways of solving the problem of optimal utilization of space can be found in the literature. These solutions have reached the maximum utilization and offer an optimal solution for specific cases only. However, most of them are characterized by

certain limitations that make these algorithms applicable to only a small number of real world situations. Most common assumption for the algorithms that solve the problem of optimal placement of elements in a given finite space is that all elements must be in a form of a box [2,3,5,14,16,17,18]. For this specific case there are a lot of algorithms that can find the optimal solution. But in the real world elements often are in a form different from the form of a box. In this case the optimum utilization of space is more complicated and should be done in a different way. The actual research in this area offers some examples of solving the problem of optimal utilization of space when items should be placed in a form different than the shape of a box. In most of the cases the elements are in the shape of a cylinder [8].

The main characteristic of most of the applications that solve the problem of optimal placement of the elements is to satisfy two main conditions. The first condition to be satisfied is the positioning of the elements in a way that does not exceed the space available for loading. The second condition is that the elements must not overlap each other. The applications that try to find the optimal solution are mainly based on these two conditions. However, a much larger number of factors affect the optimal placement of material goods such as restrictions on the rotating elements, stability of the packed elements, the complexity of the arrangement of the elements, limiting the maximum weight and fragility of the elements [1,4,21]. Some specific research and commercial applications for this purpose offer solutions that partially implement a fraction of these factors during decision making. Another feature that most studies do not take into consideration is the use of extra space that comes from elements that contain holes [6.9], which certainly leaves room for further research in this area. The main purpose of this paper is to make an improvement in this area. The purpose of our research is to implement a solution that will use this unused space and thus make a further step forward and offer a more complete solution to this real word problem.

## 3. Loading Algorithm

The loading algorithm consists of two phases. The first phase is referred to as a Grouping phase. In this phase we group the elements aiming to utilize the available hidden space (space that arises from holes in the elements) in an optimal way. In the second phase we place the already grouped elements in the available space of the storage using genetic algorithm in order to find the optimal arrangement of elements.

### 3.1 Grouping phase

In the initial phase the elements are grouped placing one or more elements in the empty space within another element. This way the set of elements for storage is reduced enabling more elements to be placed in the available space. The implementation of the grouping algorithm works as follows:

The algorithm maintains a list of elements and a list of available spaces that arise from the elements. Starting from the element with maximum volume we loop through the elements and try to find an available space within one of the elements using the

Best Fit approach. This approach makes sure that elements will be compressed as much as possible.

Within the list of elements that are maintained, each element is described with a set of values that enable simple implementation of the Best Fit approach. The set of values includes the volume of the element, the shape of the element, the size of the element, list of void spaces inside the element as well as values that are used to describe the real position of the element in the space (orientation of the element and actual position). Shape representation is implemented using regular geometric forms. Each element can be described with a set of simple geometric forms. Simple elements are described as a box, a cylinder or a sphere. More complex elements are described by a composition of simple geometric forms. The available space which needs to be filled with elements as well as void space in the elements is also represented using the same notation. The use of simple geometric forms enables finding the space that best fits with simple checks. This reduces the complexity of the algorithm itself as well as its execution time.

For each element the algorithm tries to find the space that fits best. For example, if the element has a shape of a box and also the space is in a shape of a box with the same dimensions the algorithm will place the element in that space utilizing 100% of the space. If this is not possible, the algorithm searches for the element-space combination that has the smallest difference between the volume of the space and the volume of the element. The space that is left unfilled is represented as a composition of simple geometric forms. In many cases, when one element is placed inside another, the space that is left empty can be represented as different compositions of simple geometric forms (Figure 1). For example, if we place a smaller box into another box the empty space can be represented with composition of three other boxes in six different ways. Three of them are shown on Figure 1. When this is the case, our algorithm adds all eighteen boxes to the list of available space. This is done in order to support all different elements that can be placed in this space. These spaces are referenced to each other and when one of them is used for placing of another element, the spaces from five other combinations are removed from the list. The available space that arises from the selected space is divided using the same logic.

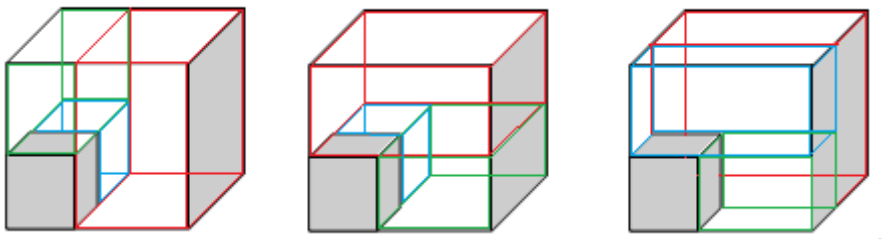

**Figure 1. Different division of space**

### 3.2 Optimization phase

The optimization phase is implemented using a genetic algorithm. This phase can also be implemented with other algorithms, but for the needs of this research we have used an already established approach. The chromosome is represented with three lists of integers (Figure 2). The first list represents the index of the elements in the element set. The second list is the orientation of the elements. The third list represents the side of the previous element to which the current element is placed to. With this representation we represent the whole loading plan. The loading starts from the left bottom corner and each element are placed next to the previous one using a bottom-up approach.

The crossover and mutation operations are done separately on each sub-list and are adapted to be in accordance with this representation [9].

The fitness function is implemented as a penalty function. If an element exceeds the available space the algorithm assigns 100 negative points to this solution. The second measure is the volume of the void space and can be from 0 to 100. The best solution is the solution with minimum negative points.

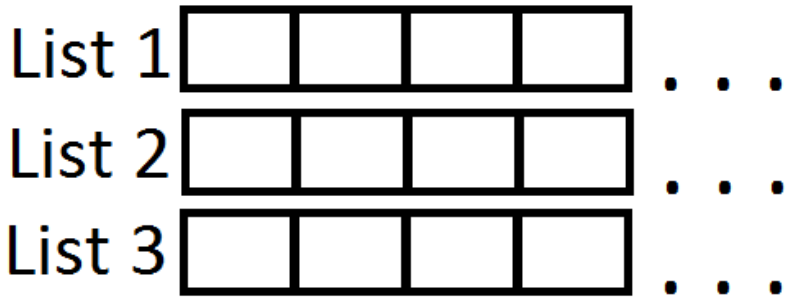


**Figure 2. Chromosome representation**

### 3.3 Heuristics

Both of the phases implement heuristics in order to use expert knowledge during the placement of the elements. Using the expert knowledge we take in consideration a lot of important factors like fragility of the elements, the max weight that can be placed on top of them, the allowed orientation of the elements. The expert knowledge is implemented as a set of rules. When an element needs to be placed in a space, the algorithm checks the rules and decides if the element can be placed that way. For example, when an element needs to be placed inside another element by the grouping algorithms, or when the element needs to be placed next to another element by the genetic algorithm operators, the algorithm checks if the element can be rotated. If not, the algorithm does not consider this case as a possible solution which results in a reduced time complexity of the algorithm.

## 4. Experimental results

The utilization of hidden spaces is highly dependent of the elements structure and properties. If the element set is consisted of elements that have a lot of free space (pipes for instance) and also the element set contains small elements, this algorithm is much better than the ones that work only with regular geometric forms and without considering the holes in the elements.

We have made different simulations with different variations of the algorithm. The experiment was done with a small data set (with 10 elements) in order to compare the different variations. As a next step we are going to test the solution with larger data sets. At the beginning we tried the algorithm omitting the first phase. In this case we used a genetic algorithm that can arrange elements that contain holes. Then we added the grouping phase. Use of the grouping phase reduced the time needed to achieve the maximum result. With element grouping the element set was simplified which enables the genetic algorithm to reach the maximum producing less generations. We also tried to use a genetic algorithm that works only with simple geometric forms. This time we did not take into consideration the holes that stayed empty at the end of first phase because this space was already used in an optimal way. This change reduced the time complexity because the genetic algorithm fitness function was much simpler. There was no change in the efficiency of the solution despite this simplification. Figure 3 shows the comparison between different variations of the algorithm. The x axis represents the number of generations while the y axis represents the fill level of the container in percentage. The chart shows that simplification of the element set with inclusion of the grouping phase reduces the generations needed to reach the optimal solution.

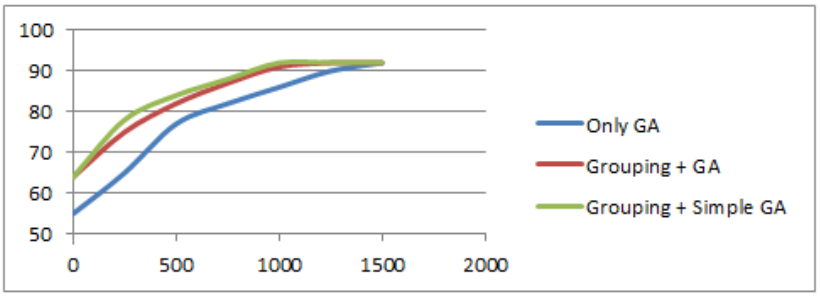


**Figure 3. Comparison of the different variations**

## 5. Conclusion

Current loading optimization algorithms have not reached the maximum efficiency and can still be improved. Using the hidden space offers a big improvement when there are a lot of elements with such characteristics. Such algorithms are applicable in many real world areas, for example for packing furniture, pipes etc.

Although this optimization problem can be solved with only an optimization algorithm, adding a simple phase at the beginning reduces the element set (depending on the structure of the element set). This enhancement reduces the time complexity of the algorithm.

Since this kind of algorithms are applicable in many areas [24] it is worth to research further in this area.

## 6. Future work

In the real world, the set of elements that need to be loaded is often the same with the set of some of the previous loading operations. With continual usage of the algorithm, the solutions can be stored and the next time the algorithm is initiated, it can start from the best previous solution for the given set of elements. Also, the stored solution can be applied if the solution is not improved during a limited number of algorithm iterations.

Also, implementing packing patterns as an expert knowledge will improve the time complexity and the efficiency of the algorithm. The pre-saved patterns can be a starting point for the algorithm.

We also want to improve this algorithm by implementing more factors as expert knowledge. By implementing new factors the solution can be applicable for optimizing the loading for transport purposes.

## 7. References


1. Bai-Sheng Chen, Yu-Fu Huang, Intelligent Cargo Loading System for Two-stages Truck Loading Problem, Takming University of Science and Technology, 2011
2. Li Pan, Joshua Z. Huang, Sydney C.K. Chu, A Tabu Search Based Algorithm for Cargo Loading Problem, University of Hong Kong, 2008
3. Shigeyuki Takahara, A Multi-start Local Search Approach to the Multiple Container Loading Problem, Kagawa Prefectural Industrial Technology Center Japan, November 2008
4. Rafael García-Cáceres, Carlos Vega-Mejía and Juan Caballero-Villalobos, Integral Optimization of the Container Loading Problem, Escuela Colombiana de Ingeniería & Pontificia Universidad JaverianaColombia, 2011
5. Oana Muntean, An Evolutionary Approach For The 3d Packing Problem, Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques, 2007
6. Santosh Tiwari, Development and Integration of Geometric and Optimization Algorithms for Packing and Layout Design, Clemson University, 2009

**7.** Kelly FOK, Ming Ka & Andy CHUN, Hon Wai, Optimizing Air Cargo Load Planning and Analysis, Department of Computer Science City University of Hong Kong, 2004

**8.** H.T. Dean, J. N. Baggaley and R.J.W. James, Three Dimensional Container Packing of Drums and Pallets, University of Canterbury, New Zealand, 1999

**9.** Ilkka Ikonen, William E. Biles, Anup Kumar, Rammohan K. Ragade, John C. Wissel, A genetic algorithm for Packing Tree-Dimensional Non-Convex Objects Having Cavities and Holes, University of Louisville, 1997

**10.** G¨unther R. Raidl, Gabriele Kodydek, Genetic Algorithms for the Multiple Container Packing Problem, Department of Computer Graphics Vienna University of Technology, 1998

**11.** Shyi-Ching Liang and Chi-Yu Lee, Hybrid Meta-heuristic for the Container Loading Problem, Department of Information Management, Chaoyang University of Technology, 2007

**12.** Eva Hopper, Two-dimensional Packing utilising Evolutionary Algorithms and other Meta-Heuristic Methods, University of Wales, Cardiff, May 2000

**13.** Sadaaki Miyamoto, Yasunori Endo, Koki Hanzawa, Yukihiro Hamasuna, An optimization system for container loading based on metaheuristic algorithms, University of Tsukuba, Ibaraki, Japan,

**14.** Guntram Scheithauer, Algorithms for the container loading problem, Dresden University of Technology, 1992

**15.** Nikhil Bansal, Alberto Caprara and Maxim Sviridenko, Improved approximation algorithms for multidimensional bin packing problems, IBM T.J. Watson Research Center & DEIS, University of Bologna, 2006

**16.** Tobias Fanslau, Andreas Bortfeldt, A Tree Sarch Algorithm for Solving the Container Loading Problem, University of Hagen, 2008

**17.** Mykolas Juraitis, Tomas Stonys, Arūnas Starinskas, Darius Jankauskas, Dalius Rubliauskas, A Randomized Heuristic For The Container Loading Problem: Further Investigations, Department of Multimedia Engineering, Kaunas University of Technology, 2006

**18.** Robert H. Storer, Joseph C. Hartman, The Container Loading Problem with Tipping Considerations, Lehigh University

**19.** Reinaldo Morabito, Marcos Arenales, An AND/OR-graph Approach to the Container Loading Problem, Universidade Federal de Sao Carlos & Universidade de Sao Paolo, 1994

**20.** A. Bortfeldt, H. Gehring, D. Mack, A parallel tabu search algorithm for solving the container loading problem, A. Bortfeldt et al. / Parallel Computing 29, 2003

**21.** Søren Gram Christensen, David Magid Rousøe, Container loading with multi-drop constraints, Technical University of Denmark, 2007

**22.** Felix T. S. Chan†, Niraj Kumar and Tse Chiu Wong, Three-Dimensional Air-Cargo Loading Problem: An Evolutionary Algorithm Based Approach, Department of Industrial and Manufacturing Systems Engineering, The University of HongKong, 2006

**23.** Sadaaki Miyamoto, Yasunori Endo, Koki Hanzawa, and Yukihiro Hamasuna, Metaheuristic Algorithms for Container Loading Problems: Framework and Knowledge Utilization, Journal of Advanced Computational Intelligence and Intelligent Informatics, 2007

**24.** Riste Marevski, Kostadin Solakov, Enhancing Company's Every Day logistics Using Cargo Loading Optimization, International Conference for Entrepreneurship, Innovation and Regional Development, Sofia 2012

S. Markovski, M. Gusev (Editors): ICT Innovations 2012, Web Proceedings, ISSN 1857-7288