ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol.35, No.1

A RANDOMIZED HEURISTIC FOR THE CONTAINER LOADING PROBLEM: FURTHER INVESTIGATIONS

Mykolas Juraitis, Tomas Stonys, Arūnas Starinskas, Darius Jankauskas, Dalius Rubliauskas

*Department of Multimedia Engineering, Kaunas University of Technology
Studentų St. 50, LT–51368 Kaunas, Lithuania*

Abstract. The knapsack container loading problem is the problem of loading a subset of rectangular boxes into a rectangular container of fixed dimensions such that the volume of the packed boxes is maximized. A new heuristic based on the wall-building approach was proposed earlier. That heuristic divides the problem into a number of layers and the packing of layers is done using a randomized heuristic. Further investigations of randomized heuristic are discussed in this paper. We focused on ways to find proportions of the mixture of heuristics which would lead to better performance of the algorithm. New results are compared with earlier research and some other constructive heuristics. The performance of the corresponding algorithms is experimentally compared for homogeneous and heterogeneous instances. Proposed improvements allow to achieve better filling ratio without increasing the computational complexity of the algorithm.

1. Introduction

The problem addressed in this paper is that of orthogonally packing a subset of some given rectangular-shaped boxes into a rectangular container of fixed dimensions. There are, however, several variants of the container loading problem depending on the objective function and constraints which are applied to packed boxes.

Strip packing. The container has fixed width and height but infinite depth. The problem is to pack all boxes such that the depth of the container is minimized.

Knapsack loading. Each box has an associated profit, and the problem is to choose a subset of the boxes that fits into the container maximizing loaded profit. If the profit of a box is set to its volume, this problem corresponds to the minimization of wasted space.

Bin-packing. All containers have fixed dimensions, and all the boxes are to be packed into a minimum number of containers.

Multi-container loading. Containers may have varying dimensions and the objective is to choose a subset of the containers, which results in the minimum shipping costs.

Several other constraints may be imposed to the above problems: only specific rotations (if any) may be allowed, feasible solutions might be restricted to guillotine cuttable, there may be restrictions on how many boxes may be put on top of each other.

When dealing with the container loading problem we have two types of the problem:

- homogeneous – container consists of identical boxes;
- heterogeneous – many different types of boxes are used.

Most of the algorithms for the container loading problem are very "data dependent": they work well on a specific distribution of the box types.

The problem considered in this paper is the Knapsack Container Loading Problem (KCLP) where we assume that the profit of a box equals its volume. The objective is to fill the container most possible. The following notations will be used through the whole paper: the container has width W, height H and depth D. A set $N = \{1\ldots n\}$ of boxes is given, each box j having width w_j , height h_j and depth d_j . The volume of each box is $v_j=w_jh_jd_j$.

2. Heuristic algorithms

The KCLP is *NP*-hard in the strong sense: only small-sized instances can be solved to optimality and large-sized instances needs heuristic approaches to be applied. The most common heuristic approaches can be classified as wall-building algorithms (Bischoff and Marriott[3], Hemminki[6], and Gehring, Menscher, Meyer[11]), stack-building algorithms (Gilmore and Gomory[15]), guillotine-cutting algorithms (Morabito and Arenales[16]), and cuboid-arrangement algorithms (Bortfeldt and Gehring[7]).

The *wall-building approach*, introduced by George and Robinson [5], fills the container in a number of layers across the depth of the container (see Figure 1).

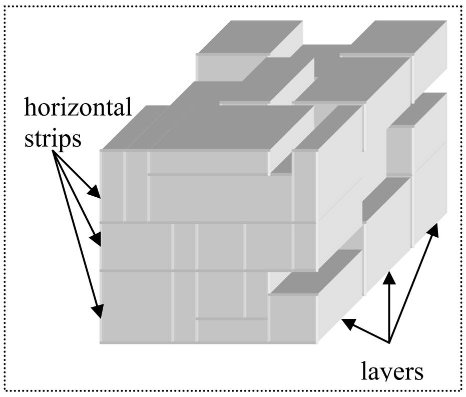

Figure 1. The wall-building approach

This heuristic is based on the normalized layer depths: only layer depths *d,* which equal some box dimensions were considered. The depth of a layer must be carefully selected to obtain a good performance. Then the wall is packed in a greedy way as a number of horizontal strips. Every strip is packed by consecutively inserting boxes with the largest ranking. The ranking is based on the smallest dimension of a box, then on the number of remaining boxes of a given type and finally on the length of the largest dimension.

We can find a lot of different heuristics based on the wall-building approach, where different ranking functions are applied. The major disadvantage of all wall-building heuristics is that they are greedy: making locally optimal choices, however, may lead to bad final solutions. To obtain a better performance, it is necessary to have some kind of backtracking possibility.

Tree-Search heuristic is an extension of the wall-building approach where a tree-search strategy is used to find a set of layer depths and strip widths. To decrease the complexity, an *m*-cut approach is used for the enumeration where only a fixed number of subnodes are considered for every branching node[8]. At each branching node, it is considered M_1 different layer depths selected according to a specific ranking rule. Each layer is then filled as a number of strips, where the strips may be oriented horizontally or vertically. Again, only a limited number M_2 of different strip widths are considered. Then a single strip is filled optimally by solving a Knapsack Problem (KP). For every layer depth $d'_1,\ldots, d'_{M1}$, a filling is chosen which has the overall best use of the volume.

The algorithm presented by Baltacioglu[2] builds walls or layers along any of the six faces of the given container. This heuristic employs both layer packing and wall building approaches. One of the most important methods, used in this algorithm, is a *layer-in-layer packing approach*: packing a sub-layer into any of the available unused spaces in the last packed layer (see Figure 2).

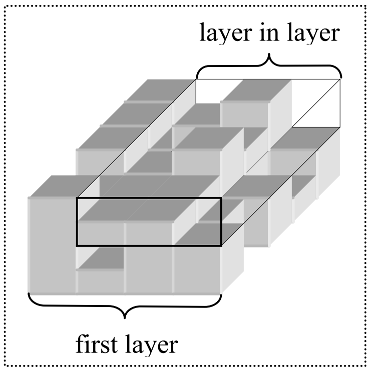

Figure 2. Layer in layer packing

Another new, and the most important, feature of this heuristic is an adaptation of human behavior and intelligence to decide which box to pack. Humans prefer to pack boxes to reduce packing surface irregularities. They eyeball the dimensions of any gap to be packed and pick the most suitable boxes to keep the topology as smooth as possible. This heuristic works in the same way. While packing a layer, it attempts to retain a flat forward packing face. In each step, the dimensions of the gaps to be filled are determined, all the eligible boxes and their orientations are analyzed. The algorithm performs layer packing or wall building thus reducing the problem to a systematic series of two-dimensional packing problems.

2.1 Randomized Heuristic

After some experiments, three different heuristic fitness functions were chosen for selecting next box for packing:

1. $f_1 \rightarrow \max\{v_i\}$

A box with the largest volume is the most valuable.

2. $f_2 \rightarrow \max\{v_i / V_i\}$, where V_i is the bounding sphere volume.

A box with the largest ratio *box_volume/bounding_sphere_volume* is the most valuable. We assume that cube is the best shape by its nature. According to the ratio present in this function, we can decide how close the box is to the cube by its shape.

3. $f_3 \rightarrow \max\{\min\{w_i, h_i, d_i\}\}$

A box, whose smallest dimension is the largest, is the most valuable.

Dividing a container into several layers could give better solutions. A container is divided into layers along its largest dimension. The layer depth, which gives the maximum value to the ranking function, is chosen for the packing. We use ranking functions described by Pisinger[14]:

4. $f_k^1 = \sum_{i=1}^{n} 1(w_i = k \vee h_i = k \vee d_i = k) \; for \;\; k = \alpha,\ldots,\beta$

5. $f_k^2 = \sum_{i=1}^{n} 1(\max\{w_i, h_i, d_i\} = k) \qquad for \;\; k = \alpha,\ldots,\beta$

6. $f_k^3 = \sum_{i=1}^{n} 1(\min\{w_i, h_i, d_i\} = k) \quad for\ k = \alpha, ..., \beta$

7. The fourth frequency function f^4 returns the dimension of the box whose smallest dimension is the largest among the remaining yet unpacked boxes. It's the same as (3) heuristic for box selection:

$f^4 = \max\{\min\{w_i, h_i, d_i\}\}, \quad i = 1...n$

The experiments showed that it's possible to obtain better results using a mixture of these heuristics. For each layer we had tried four layer depth values obtained using these heuristics and chosen the solution with better filling value. The best solution was achieved by taking the two best ranked layer depths according to (4) and (7) heuristics and selecting the better filled one. This approach works as a limited backtracking possibility.

First of all, every layer is filled using all three mentioned greedy approaches (1-3). After that, one additional heuristic for selecting next box for packing was implemented:

8. A box having at least one dimension equal to the current layer depth is the most valuable; if there is no such box, then the most valuable box is assumed by previous heuristics.

Next, we pack using a mixture of (1), (2), (3), (8) greedy (pure) heuristics and Monte-Carlo search: a mixture approach may lead to better results than using pure heuristics and Monte-Carlo search separately as it was shown in [6]. This part of the algorithm is executed a given number of iterations (the same for all layers).

Heuristics (1), (2), (3) and (8) are used to determine which box should be packed next. For choosing where to place the next box, we maintain a list of possible box placement locations. Initially this list contains only one point, which is a corner of the container (or layer when packing layer). Then we search for box, which could be placed at this point and fits into container (layer), i.e. does not exceed container (layer) dimensions. If the box fits into given point, then we remove this point from the list and add three new points. If the box does not fit, then it is rotated and it's tried again to place the box. If the box does not fit in any orientation, then it is ignored.

The main advantage of our algorithm is its independency from homogeneousness of the problem. Unfortunately, usually most of the algorithms are very "data dependent". Another advantage of our algorithm is its run time scalability. In most cases after just a few iterations it achieves a quite good solution, which differs from the solution got after a reasonable amount of time only by 3-6%. On the other hand, the experiments show that the convergence of solution using our algorithm is near logarithmic and after some point there is no sense to expect to obtain a better solution after an acceptable time.

2.2 Further Investigations

It became obvious from experimental results that changing proportions of the mixture lets achieve better results for specific packing problems. Our new goal was to find the best proportions of the mixture while remaining algorithm's independency from homogeneousness of the problem and additional restrictions.

We derived the proportions of the mixture by utilizing the genetic algorithm (GA). A population consisting of 20 individuals was used. BT-20 data set was used for evaluation of individuals. The average fill of all 100 problems was used for evaluation. Each chromosome had 5 genes which were represented as numbers from the interval [0; 100) and corresponded to proportions of the mixture. The ith component of the mixture p_i is calculated by the formula:

$$p_i = \frac{g_i}{\sum_{j=1}^{5} g_j}, \qquad (1)$$

where g_i is value of the gene. For creation of new generation, elitist strategy was used. All individuals better than average were moved to a new generation. Two new mutants from these individuals were created. To reach population of size 20, all other new individuals were generated by one point crossing of individuals already in new generation. The proportions from 26th generation are presented in Table 1.

Table 1. Proportions of the mixture derived by utilizing genetic algorithm

Probability	Method
0,063	heuristic (8)
0,278	heuristic (1)
0,051	heuristic (2)
0,519	heuristic (3)
0,089	Monte-Carlo search

As was mentioned before, changing proportions of the mixture allows to achieve better results for specific packing problems. Naturally, if we could find a mechanism to choose specific mixture for each problem, we would expect an overall increase of the algorithm's performance while packing wide range of problems. A possible solution could be to use artificial neural network (ANN). An input to such network would be some parameters describing problem type. An output would be proportions of the mixture.

We have investigated usage of multilayered perceptron for this particular problem. Proportions of the mixture were calculated by the same formula (1) which was used for GA assuming that g_i is the output from the ith neuron in the output layer. Thus the ANN had 5 neurons in the output layer. Four input values were passed to the ANN:

1. value 1 to introduce bias;
2. *1 / number_of_different_box _types;*

3. *1 / number_of_total_box_count;*

4. *total_boxes_volume / container_volume*.

The above attributes should allow to distinguish various types of packing problems. For example, attribute 2 represents homogeneousness of the problem. Of course these attributes are not universal and not allow to differentiate among all imaginable packing problems but experimentally we have shown that they are sufficient for 7 types of problems which were used to evaluate our algorithm and compare it with other works.

The sigmoid function was used as transfer function for neurons in all layers. Good results were achieved using multilayered perceptron with one hidden layer and 6 neurons in it.

For training of the ANN, we utilized the genetic algorithm once more. This time population of 10 individuals was used. The best individual was moved to a new generation. Three individuals were generated by uniform crossing of the best individual and random individual taken from the best five (not counting the best one). Four individuals were generated by mutating random individuals from best five (not counting the best one but including already crossed of even mutated individuals). Mutation of each weight was performed with probability 0.2. Mutated weight was changed by random number drawn from a Gaussian distribution with rew mean and standard deviation 1. The last 2 individuals were generated using uniformly distributed random weights from the interval [-1; 1).

3. Experimental Results

To get a more balanced picture on the algorithms, we have to consider a wide range of instances – from homogeneous problems to strongly heterogeneous problems. Unfortunately, the experimental results and comparison with other algorithms show that usually most of them are very "data dependent": they are dedicated for one type of problems – homogeneous or heterogeneous.

For the weakly heterogeneous problems Hemminki[6] obtained a filling ratio of 84.5%. Using genetic algorithms to choose the depths of the layers, Hemminki obtained an average filling ratio of 87.0%. Bischoff and Ratcliff[4] report a filling ratio around 83.0% for problems similar to the weakly heterogeneous problems. Bortfeldt and Gehring[7] obtained filling ratio of 83.9% for the same instances. Gehring and Bortfeldt[9] obtained a filling ratio of around 87.7% for weakly heterogeneous instances similar to those of Bischoff and Ratcliff.

To make our research results comparable with the others, we had used 7 classes of data presented by Bischoff. The data files are available on Imperial College Management School web page – *OR-Library* (http://www.ms.ic.ac.uk/info.html). Usually, these data sets are used as a test suite for comparisons (benchmark instances). Each class of the data contains 100 data instances with 3, 5, 8, 10, 12, 15 and 20 box types, respectively. The same container dimensions are used in all of the problems: $587 \times 233 \times 220$.

The following parameter values were used for randomized heuristic to obtain current results:

- for each layer, we tried to find a solution using the mixture presented in Table 1;
- the mixture was tried for each layer for 1000 iterations;
- for each layer, there were considered 2 candidate depth values got using heuristics (4) and (7);
- the packing procedure was repeated 5 times.

Repeating packing procedure several times makes sense because different layer fillings can yield different candidates for the next layer, so we can get different results.

We present the results of comparison (according to utilization of the container) of eight algorithms in Table 2: Bischoff and Ratcliff[10] (denoted by B.R.), Gehring and Bortfeld[11] (denoted by G.B.), Bortfeld and Gehring[12] (denoted by B.G.), Terno, Scheithauer, Sommerweiß, and Riehme[13] (denoted by T.S.S.R.), Pisinger[1] (denoted by P.), Baltacioglu[2] (denoted by B.), randomized heuristic [17] (denoted by J.R.S.) and randomized heuristic using proportions of the mixture derived by utilizing genetic algorithm (denoted by GA)

The best competitors to our algorithm seem to be methods developed by Pisinger[1] and Baltacioglu[2]. The algorithm proposed by Pisinger can achieve better maximum utilization of the container, but our minimum case for most of data sets is better. The dispersion of utilization of the container for our algorithm is less what means more stability. Besides, our algorithm is less "data dependant. As it is obvious from Table 2, all other methods are dedicated for some special data sets. Our algorithm remains more stable through all classes of data sets. We also see that proportions of the mixture derived by utilizing genetic algorithm significantly improve packing when we allow all possible box rotations. Although it is worth noting that improvement is not equal for all data sets.

We have investigated the change of effectiveness of the solutions while changing rotation constraints for boxes. The average results are shown in Table 3. It's clear from the table that the best result can be achieved when all possible rotations are allowed. Proportions of the mixture derived by utilizing genetic algorithm also improve packing when only given box rotations are allowed but degrade results when no rotations are allowed at all. This is expected because evaluation function of the genetic algorithm used for finding better mixture was based on the case where all possible rotations of the boxes were allowed. By choosing different evaluation function we could improve algorithm performance for other specific types of the problem.

Table 2. Comparison of different algorithms according to container filling (%)

Algorithm		Data Set						
		BT-3	**BT-5**	**BT-8**	**BT-10**	**BT-12**	**BT-15**	**BT-20**
B.R.	*Min*	73.7	73.8	75.3	78.4	–	–	75.7
	Ave	85.4	86.3	85.9	85.1	–	–	83.0
	Max	94.4	93.8	92.6	90.1	–	–	88.3
G.B.	*Min*	76.7	78.4	81.1	82.7	–	–	84.4
	Ave	85.8	87.3	88.1	88.0	–	–	87.7
	Max	94.3	95.2	92.9	91.6	–	–	90.7
B.G.	*Min*	78.7	79.7	82.4	80.9	–	–	79.9
	Ave	89.0	88.7	88.2	87.4	–	–	83.9
	Max	95.7	95.0	94.0	92.0	–	–	88.4
T.S.S.R.	*Min*	75.7	81.9	83.2	83.1	–	–	80.6
	Ave	89.9	89.6	89.2	88.9	–	–	86.3
	Max	95.9	94.7	93.0	92.7	–	–	89.0
P.	*Min*	79.3	84.9	86.7	86.1	85.2	86.7	85.4
	Ave	88.6	89.9	89.4	88.4	88.9	88.6	88.7
	Max	96.1	95.8	94.7	94.9	94.3	92.9	92.7
B.	*Min*	78.9	84.8	84.5	84.4	84.0	84.3	84.3
	Ave	89.0	89.0	88.4	88.2	87.6	87.4	87.1
	Max	95.3	94.0	92.1	91.9	89.9	91.3	90.2
J.R.S.	*Min*	83.2	85.5	85.3	86.2	85.9	86.3	85.9
	Ave	88.3	88.9	89.2	89.2	89.1	89.1	89.0
	Max	93.7	94.6	92.5	92.8	92.3	93.1	91.9
GA	*Min*	82.4	84.1	84.1	84.9	85.8	86.4	86.4
	Ave	88.4	89.0	89.4	89.5	89.4	89.6	89.5
	Max	92.8	93.7	92.8	92.9	93.1	93.0	91.7

Table 3. Comparison of different algorithms according to container filling (%) when not all possible rotations of the boxes are allowed

Algorithm		Data Set						
		BT-3	**BT-5**	**BT-8**	**BT-10**	**BT-12**	**BT-15**	**BT-20**
J.R.S.	**No rotations**	79.6	82.5	83.7	84.0	84.2	84.1	84.1
	Given rotations	85.6	86.7	87.5	87.8	87.8	87.8	88.0
GA	*No rotations*	79.7	82.3	83.2	83.5	83.6	83.8	83.4
	Given rotations	86.3	87.3	88.0	88.1	88.2	88.2	88.4

As was mentioned in the previous section, we tried to use ANN to map some attributes of the problem to proportions of the mixture which fits well for that particular problem. The ANN was trained with last 10 problems in each data set. Testing was performed by using first 90 problems from each data set. Results are shown in Figure 1.

In order to make a fair comparison, the results presented in Figure 1 were calculated using first 90 problems from each data set. All possible box rotations were allowed. As we see from the Figure 1, the results obtained by utilizing ANN are much better than J.S.R. but not as good as in GA case. This is not surprising because all 100 problems from BT-20 data set were used for training in GA. Naturally, GA provides the largest improvement for BT-20 data set when comparing with J.S.R. Also, the results of ANN case could be degraded due to small training data set. Nevertheless, the ANN method is more universal and provides more predictable improvement of container filling.

4. Conclusions

We have presented several ways to find proportions of the mixture in order to improve the performance of heuristic algorithm for solving the Knapsack Container Loading Problem.

Our tests demonstrate the validity and performance of the algorithm. We experimentally compared our results with those reported in other works. The benchmark data instances were used for the tests. The results show that our algorithm can achieve better filling on the average than others. It is also worth noting that the proposed improvements allow to achieve better filling ratio without increasing computational complexity of the algorithm. The algorithm also provides the stability of computational results (filling of the container) throughout all used data sets: the algorithm works well with homogeneous and heterogeneous types of problems.

Future directions of the research could be searching for better way to map attributes of the problem to proportions of the mixture which fits well for that particular problem instance. Also, the same techniques as presented here could be used not only to decide proportions of the mixture but also to improve heuristics for layer choosing.

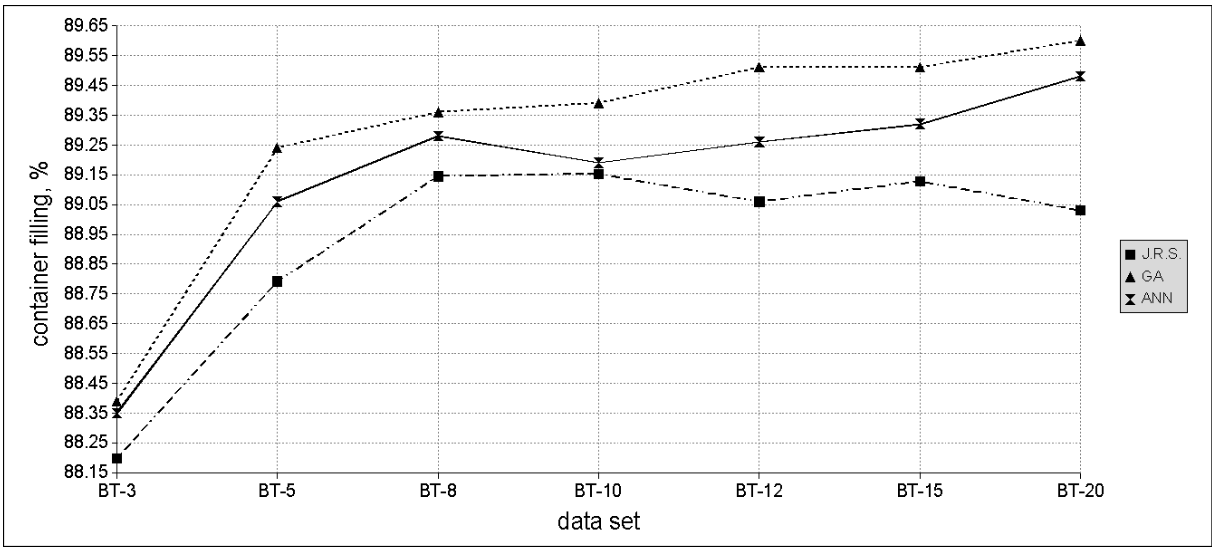

Figure 1. Comparison of algorithms according to container filling (%) when packing first 90 problems from each data set

References

[1] **D. Pisinger.** A tree search heuristic for the container loading problem. *Ricerca Operativa,* 28, 1998, 31-48.

[2] **E. Baltacioglu.** The distributer's three-dimensional pallet-packing problem: a human intelligence-based heuristic approach. *Working paper, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, USA,* 2001.

[3] **E.E. Bishoff, M.D. Marriot.** A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44, 1990, 267-276.

[4] **E.E. Bishoff, M.S.W. Ratcliff.** Loading Multiple Pallets, *Journal of Operational Research Society,* 46, 1995, 1332-1336.

[5] **J.A.George, D.F.Robinson.** A heuristic for packing boxes into a container. *Computers and Operations Research,* 7, 1980, 147-156.

[6] **Hemminki.** Container loading with variable strategies in each layer. *ESI-X, EURO Summer Institute, Jouy-En-Josas, France,* 1994.

[7] **A. Bortfeldt, H. Gehring.** Applying Tabu Search to Container Loading Problems. *Operations Research Proceedings,* 1997, 533-538.

[8] **T. Ibaraki.** Enumerative Approaches to Combinatorial Optimization. *Part* 2, *Annals of Operations Research*, 11, 1987.

[9] **H. Gehring, A. Bortfeldt.** A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research,* 44, 1997, 401-418.

[10] **E.E. Bischoff, M.S. W.Ratcliff.** Issues in the development of approaches to container loading. *OMEGA*, 23, 1995,:377-390.

[11] **H. Gehring, A. Bortfeld.** Ein genetischer Algorithmus für das Containerbe-Ladungsproblem. *Technical Report 227, FB Wirtschaftswissenschaft, Fern Universität Hagen*, 1996.

[12] **A. Bortfeld, H. Gehring.** Ein tabu search-Verfahren mit schwach heterogenem Kistenvorrat. *Technical Report 240, FB Wirtschaftswissenschaft, Fern Universität Hagen,* 1997.

[13] **J. Terno, G. Scheithauer, U. Sommerweiß, J. Riehme.** An Efficient Approach for the Multi-Pallet Loading Problem. *Institute for Numerical Mathematics, Technical University Dresden, Dresden, Germany,* 1997.

[14] **D. Pisinger.** Heuristics for the container loading problem. *European Journal of Operational Research,* 141, 1997, 143-153.

[15] **P.C. Gilmore, R.E. Gomory.** Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13, 1965, 94-120.

[16] **R. Morabito, M. Arenales.** An AND/Orgraph approach to the container loading problem, *International Transactions in Operational Research,* 1, 1994, 256-267.

[17] **M. Juraitis, A.Riškus, T.Stonys** A Randomized Heuristic for the Container Loading Problem. *Information technology and control, Kaunas Technologija, No.*1 (26) 2003, 23-31.

Received December 2005.