A Comparison Between Exact and Approximate Method for Solution of General One-Dimensional Cutting Stock Problem

Peter Trkman and Miro Gradišar
Faculty of Economics
Kardeljeva ploscad 17
1000 Ljubljana
Slovenia

Keywords: cutting, optimisation, branch & bound, decision tree

Received: July 17, 2003

The paper describes exact solution of general one-dimensional cutting stock problem (G1D-CSP) where all stock lengths are different. Branch & Bound (B&B) optimization method is used. The solution is cutting plan with minimized overall trim loss in such a way that order lengths are cut in exactly required number of pieces and only one stock length is in general not cut to the end. If it's long enough then it can be used later and is not treated as a waste. G1D-CSP can also be solved approximately with Sequential Heuristic Procedure (SHP). Comparison between B&B and SHP is presented. It is shown that exact solution is better when the size of the problem does not exceed certain limit. The question is, how to determine this limit, which can be different in different practical situations. An approach, based on decision trees, for the selection of appropriate method for each individual case, is proposed. Numerous examples are calculated.

1 Introduction

One-dimensional stock cutting occurs in many industrial processes [4,8,9,16,19] and during the past few years it has attracted increased attention of researchers from all over the world [1,2,11,12,18,21]. Most standard problems related to one-dimensional stock cutting are known to be NP-complete and in general a solution can be found by using approximate methods and heuristics. However, the unbelievable development of computers and constantly growing processing power are pushing the complexity limit of the cutting problems, where exact methods could be used, slightly up. Therefore importance of exact methods is growing and the number of practical situations, where they can be used, increases [8,14, 18].

Dyckoff [3] classifies the solution of cutting stock problems into two groups: *pattern oriented* and *item-oriented*. In the *pattern-oriented* approach, at first, order lengths are combined into cutting patterns, for which - in a succeeding step - the frequencies, that are necessary to satisfy the demands, are determined. This approach can be applied when the stock is of the same length [6] or of few groups of standard lengths [7]. *Pattern-oriented* approach is most common in practice and therefore the type of the problem we get in this case is designated as Standard One-Dimensional Cutting Stock Problem (S1D-CSP) [5,12,13]. To solve (S1D-CSP) the classic LP-based Gilmore and Gomory's "delayed pattern generation" [6,7] or any other LP-based Method (LPM) is mostly used [16,18,19,22,23].

The *item-oriented* approach is characterized by individual treatment of every item to be cut. If stock lengths are all different then frequencies others than 0 and 1 cannot be determined and only an *item-oriented* solution can be found [9,10].

An *item-oriented* approach is more general and can be theoretically applied regardless of the *assortment of large objects* [3,12,13]. Therefore the problem we get in this case is designated as General One-Dimensional Cutting Stock Problem (G1D-CSP) [11,13]. There are two possibilities to solve G1D-CSP: exact methods (branch and bound, dynamic programming) or approximation algorithms in form of SHP [20]. SHP seems to be better regarding robustness and potential usefulness in a wide range of cases. Also time complexity of SHP is in general lower. On the other hand exact solution is better where the size of the problem doesn't exceeds acceptable limits and becomes intractable. But even in such cases approximate solution obtained in some acceptable time period, as a temporary result of exact method, can be comparable with the one of SHP.

Many examples of exact methods for solving different types of cutting problems are described in literature [8,18]. But we didn't found any exact solution of G1D-CSP so far.

If there are many methods available for the solution of G1D-CSP, then it can be difficult to select the right one. Therefore we decided to propose an approach based on decision trees for the selection of suitable solution method, which will take in the account the problem size, the quality of the solver and the computer speed.

This paper is organized as follows: in next section the definition of cutting problem is given. Exact solution development, using Branch & Bound optimization method in the form of a computer program, is presented with numerous practical examples. In section 4 those results are compared with the results, acquired with SHP. Finally a new approach, based on decision trees for the selection of suitable solution method for each individual case, is proposed. It is shown that this approach can indeed lead to reduced trim loss.

2 Problem definition and solution method

G1D-CSP is one-criterial minimization problem. The criterion is overall trim loss. G1D-CSP satisfies two general conditions:

1. If there is abundance of material order lengths are cut in exactly required number of pieces (In S1D-CSP this number can be greater than demanded.).
2. Only one stock length cannot be cut to the end. The result is residual length that can be used later (In S1D-CSP all used stock lengths are cut to the end.).

Details in problem definition are similar to those in [12] and [10]. The difference regarding [12] is that we also included orders that cannot be fulfilled entirely due to shortage of material in stock. The difference regarding [10] is that distribution of uncut pieces by order lengths is not included. It is more difficult to control factors other than trim loss with exact methods than with SHP. Other differences regarding both [12] and [10] doesn't affect the content of the problem but only the way of expression, which is adapted to the solution method.

First we need to decide whether we will use branch and bound method or some dynamic programming. Branch and bound (B&B) exact method was chosen. There are two reasons for that. First B&B is standard method and second, the market is offering many OR computer packages with B&B. Some of them allow using B&B as a subroutine. This means that it could be included in some application program. In our case the application program collects data, checks whether there is an abundance or shortage of material, solves the appropriate model and displays the results.

Problem is defined as follow:

For every customer order a certain number of stock lengths is available. In general all stock lengths are different. We consider the lengths as integers. If they are not originally integers we assume that it is always possible to multiply them with a factor and transform them to integers. It is necessary to cut a certain number of order lengths into required number of pieces. The following notation is used:

s_i = order lengths; $i = 1,...,n$.

b_i = required number of pieces of order length s_i.

d_j = stock lengths; $j = 1,...,m$.

x_{ij} = number of pieces of order length s_i having been cut from stock length j.

y_j indicates whether the stock length j is used in the cutting plan (y_j=1 if stock length j is not used in the cutting plan).

u_i indicates whether the remainder of stock length j is greater than UB (upper bound for the trim loss). u_j can equals 1, only if the remainder is greater than UB. Stock length that is longer than UB does not necessarily count as trim loss.

δ_j indicates the remainder of the stock length j.

t_j indicates the extent of trim loss relating to stock length j. t_j=δ_j for all used stock lengths, except for one that is longer than UB and can be returned to the stock and used again later (where u_j=1) and all unused stock lengths (where y_j=1).

Two cases are possible:

Case 1: the order can be fulfilled as the abundance of material is in stock.

(1) $\min \sum_{j=1}^{m} t_j$ (minimize trim loss which is smaller than UB)

s.t.

(2) $\sum_{i=1}^{n} s_i \cdot x_{ij} + \delta_j = d_j\,(1-y_j)$ $\forall j$ (knapsack constraints, the total length of pieces cut from a stock lengths cannot be longer than the total length of the stock)

(3) $\sum_{j=1}^{m} x_{ij} = b_i$ $\forall i$ (demand constraints, the numbers of pieces are all fixed)

(4) $UB - \delta_j + UB\cdot(u_j - 1) \le 0$ $\forall j$ (u_j can be 1, only if the remainder of stock length is longer than UB)

(5) $\sum_{j=1}^{m} u_j \le 1$ (maximum number of used stock lengths that do not count as trim loss)

(6) $\delta_j - t_j - (u_j + y_j)\cdot(\max d_j) \le 0$ $\forall j$ (t_j equals δ_j for all stock lengths where $u_j = t_j = 0$; otherwise t_j can be 0)

(7) $UB \le \max s_i$

$x_{ij} \ge 0$, integer $\forall\, i, j$

$t_j \ge 0$ $\forall j$

$\delta_j \ge 0$ $\forall j$

$u_j \in \{0,1\}$

$y_j \in \{0,1\}$.

Case 2: the order cannot be fulfilled entirely due to shortage of material (the distribution of uncut order lengths is not important).

(1) $\min \sum_{i=1}^{n} \delta_j$ (minimize sum of trim losses)

s.t.

(2) $\sum_{i=1}^{n} s_i \cdot x_{ij} + \delta_j = d_j$ $\forall j$ (knapsack constraints)

(3) $\sum_{j=1}^{m} x_{ij} \le b_i$ $\forall i$ (demand constraints)

(4) $x_{ij} \geq 0$, integer $\forall\, i, j$

$\delta_j \geq 0$ $\forall\, j$.

Unutilized stock length that is larger than some *UB* can be used further and is not considered as waste. The question is how to determine *UB*. This depends on the relation between available material and total needs.

Let's consider the case 1 first. If sufficient stock lengths are available there will be cutting plans with "no trim loss" but ever growing stocks. To prevent this an additional condition (case 1, condition (5)) has to be set: only one residual length may be longer than the *UB*. *UB* can be set arbitrarily between 0 and max s_i. The larger *UB* means greater the cutting problem and higher trim loss. *UB* = min s_i is found in practice [9].

In case 2, however, *UB* is not included in the model. If, for example, *UB* is reduced to min s_i, this would lead to the following problem: As the aim of the algorithm is the minimization of overall trim loss, this could lead to unfulfilled requirements for the longest order lengths, even if the overall trim loss would be small and the aim would be achieved according to the logic of the algorithm. The trim losses, which would be longer than *UB* but shorter than the longest order lengths, could remain unutilized. For that reason *UB* shouldn't be less than max s_i. On the other hand if the *UB* would be set to max s_i any trim loss longer than max s_i can certainly be used to cut an additional order length, so *UB* equal or longer than max s_i would not have any influence on the solution. Therefore UB is not included in the 2nd model.

3 Results

For all calculations *MPL/CPLEX* solver on the PC (AMD, 1300 MHz) was used. The data was generated and saved in MS Excel and the solver was called with Visual Basic for Applications. In first experiment we found a solution of a problem described in [10]. The improved solution (Fig. 1) was obtained in 10.1 seconds after examining 59467 integer nodes. The total trim loss is 1 cm, while the solution in [10] has a trim loss of 2 cm. In Fig. 1 firstly all details about order and stock lengths are shown (length and number of pieces demanded for each order length). Then the detailed cutting plan is presented (which and how many order lengths are cut from each stock length) and lastly the trim loss for each stock length and number of realized and unrealized order lengths.

The presented problem is relatively small (4 stock and 5 order lengths) and therefore appropriate for exact method. However with the growing number of integer variables the complexity of the problem and the solution time grow quickly. Sometimes it is not possible to find the optimal solution within reasonable time limit. Fortunately B&B works in such a way that it approaches gradually to the optimal solution and in the mean time offers temporary results, which can be near to optimum. To test the correlation between time limit and trim loss each problem instance in following experiments was solved with 6 different time limits.

For generation of problem instances we decided to use problem generator *PGEN* [12,13]. With *PGEN* it is possible to regenerate test data using the same seed, then to find the solution with some other method and compare the results. Input data are generated according to problem descriptors as random sample of one or more test problems. Problem descriptors are:

n	- number of different order lengths
v_1, v_2	- lower and upper bound for order lengths, i.e. $v_1 \leq s_i \leq v_2$ ($i = 1,...,n$)
d	- average demand per order length
m	- number of non-standard stock lengths
u_1, u_2	- lower and upper bound for non-standard stock length, i.e. $u_1 \leq d_j \leq u_2$ ($j = 1,...,m$).
r	- number of consecutive generated problem instances.

Test problems have been generated with the following values of parameters:

- determination of order lengths and demands:

By assigning different values to the problem parameters n (n = 5, 10, 15), v_1 and v_2 (v_1 = 100 and v_2 = 200, v_1 = 200 and v_2 = 400, v_1 = 300 and v_2 = 600) and d (d = 5, 10, 15) and combining them with each other 27 problems have been generated.

- determination of non-standard stock lengths:

Number of non-standard stock lengths m varies from 5 to 15, lower bound u_1 from 1000 to 3000 and upper bound u_2 from 2000 to 6000.

The details about generation of problem descriptors and determination of seed for sequences of test problems are shown in the dynamic programming scheme of the procedure *PROGEN*.

Procedure PROGEN:

```
for i = 1 to 3
  for j = 1 to 3
    for k = 1 to 3
      $n \leftarrow i \cdot 5$
      $v_1 \leftarrow j \cdot 100$
      $v_2 \leftarrow j \cdot 200$
      $d \leftarrow k \cdot 5$
      $m \leftarrow k \cdot 5$
      $u_1 \leftarrow k \cdot 1000$
      $u_2 \leftarrow k \cdot 2000$
      $c \leftarrow int(\frac{m}{10}) \cdot 9+1$
      $seed \leftarrow m+10 \cdot c \cdot d+10 \cdot c^2 \cdot v_2+1000 \cdot c^2 \cdot v_1+1000000 \cdot n$
      $r \leftarrow 10$
      call PGEN (n, v1, v2, d, m, u1, u2, seed, r)
    next k
  next j
next i
```

PROGEN procedure is implemented with Visual Basic. Problem descriptors generated with *PROGEN* procedure for 27 test cases are presented in Table 1. In the table lower and upper bounds for stock and order lengts are shown, as well as the number of different order and stock lengths and the average demand per order length. The seed for generation of test problems which enables everyone to generate the same test problems is also shown.

For each test case procedure *PGEN* generates 10 consecutive problem instances (r=10). In total there are 270 problem instances, 150 with abundance and 120 with shortage of material.

Each of the 270 problem instances was solved 6 times (using the appropriate model either for lack or abundance of material) with different time limits (time limits were set at 2, 10, 20, 30, 45 and 60 seconds)

The generated data and the solution for first generated instance of the first case are presented in Fig. 2. The optimal solution with trim loss 1 cm was found in 3.9 seconds. The meaning of columns is the same as in Fig. 1.

Stock length 2 is not used in cutting plan and stock length 1 is not cut to the end. Since δ_1=989 and UB=102, t_1=0. So stock length 1, which is larger than UB, can be used later and is not considered a waste.

The summarized results for all 270 instances with $UB = \min s_i$ with different time limits are presented in Table 2. For each of the different time limits the total trim loss, percent of trim loss and the number of optimally solved instances is shown. The 3rd column indicates how many instances were solved optimally within the given time limit. In each row 10 problem instances are summarized. Trim loss is calculated as the sum of trim losses of all 10 instances, the percentage is calculated as the average of 10 percentage losses.

In 2 seconds an optimal solution for 57 cases was found, in 60 seconds for 110 cases. The average trim loss varies from 0% to 2.4%.

The trim loss is the largest in case number 7 although the solution is optimal in all 10 instances. Low values of n, d, m mean that in this case the problems are relatively easy to solve, however due to small ratio between stock lengths and order lengths as well as the small number of possible combinations, the optimal solution has a relatively high trim loss. To a lesser extent the same is also true for case No. 4. In other cases the trim loss is 1% or lower. Even better results could be obtained by increasing the time limit for the solution.

4 Comparison with CUT procedure

We have compared results of proposed exact method with the results of SHP CUT described in [10]. The results are shown in Table 3. As in previous table 10 instances are summarized in one line. UB is set to $\min s_i$. In the second column it is indicated whether there is enough material or not. Y/N means that in some problem instances there is enough material, in others not. The total trim loss and percent of trim loss with both methods (exact and heuristic method CUT) are shown for each case.

In total within the given time limit (60 seconds) the exact method found a better solution in 64 instances, while the CUT procedure in 139 cases, in 67 cases both methods found the solution with the same trim loss (not necessarily the same solution though). The exact method has better results for cases with smaller d and m (in all cases with d<=5 and m<=5 a better solution was found with the exact method even with small time limits), the CUT procedure for larger cases (d>=10 and m>=10).

5 Selection of the method

Although it is clear from table 3 that exact method is more suitable for smaller cases and heuristics for larger, we need more precise criteria for the selection of solution method in each individual case. The main question that needs to be answered is, what is the maximum size of the problem that can be solved optimally within the given time limit. The question can be answered by using mathematical analysis of computational complexity. But for precise answer we would need a very precise data of speed of particular processor executing specific instructions generated by specific compiler and detailed data about solver. This data is usually not available. Even if they would be, the mathematical analysis would be extremely complicated. Therefore we decided to answer the question by using statistics. The new approach based on the creation of decision tree and its use for the selection of the right method is presented in this chapter.

The main idea of our approach is to generate a large number of cases with different parameters by using the problem generator and then solve them with the selected method and the given time limit. The time limit can be chosen arbitrarily and depends on what is considered as a maximum acceptable solution time in some specific practical situation.

Each case is than assigned either a class value 1 (if optimal solution was found within selected time limit) or 0 (otherwise). Those parameters and class values are then used as the data for decision tree classifiers. Decision trees were chosen as our kind of the problem fulfills the key requirements that are needed for successful implementation of decision trees (as listed by Quinlan [17]):

- attribute-value description: each test case in our example can be described with the same attributes (number of stock and order length, average demand per order length etc.),
- predefined classes: each case is assigned to one of the two predefined classes (either the case can be solved optimally within time limit or not),
- discrete classes: both classes in our example are discrete,
- sufficient data: sufficient number of problem instances can be automatically generated and solved using problem generator and solving procedure,

- "logical" classification models: our example can be expressed as decision trees or sets of production rules.

This approach can be used in various cases:
- to determine whether a certain problem should be solved optimally or heuristically. An example is shown in detail in this section,
- to determine which factors have the greatest influence on time complexity of the problem for the proposed solution method,
- to determine the appropriate size of the sub problem. Sometimes it is possible to either divide a problem in a set of smaller sub problems and solve each sub problem optimally or to solve the majority of the problem heuristically and only the small part optimally (the part that is the most important or can cause higher loss). In order to use his kind of a method, an appropriate size of the sub problem can be determined with this approach.

The example of the latter is shown in [15].

As stated earlier we have decided to test the cases with number of stock and order lengths between 5 and 10. The following procedure was used to determine 243 test cases. 5 test instances were generated for each case so we had 1215 problem instance in total:

```
for g=1 to 3
 for h=1 to 3
  for i=1 to 3
   for j=1 to 3
    for k=1 to 3
      $u_1 \leftarrow 1000 . g$
      $u_2 \leftarrow 2000 . g$
      $m \leftarrow (j . 2)+3$
      $d \leftarrow (i . 2)+3$
      $v_1 \leftarrow h . 100$
      $v_2 \leftarrow h . 200$
      $n \leftarrow (g . 2)+3$
      $seed \leftarrow 1000000 .n+1000 .v_1+10 .v_2+10 .d+m$
      $r \leftarrow 5$
      call PGEN $(n, v_1, v_2, d, m, u_1, u_2, seed, r)$
    next k
   next j
  next i
 next g
next h
```

The meaning of the variables is the same as in the previous example.

Each problem instance was then solved with the MPL/CPLEX solver and for every instance the solution time, total trim loss and the fact whether the problem instance was solved optimally or not was recorded. All cases were then distributed into two classes: 1 (optimally solved cases) and 0 (cases not solved optimally).

The whole experiment, which means generating the data and solving all problem instances within the time limit of 1 minute, took just over 10 hours. MS Excel was used for collecting and saving the results. The procedure for the whole experiment was written in Visual Basic for Application. The first 150 problem instances (5 problem instances are summarized in one line) and their solutions are shown in Table 4. The meaning of the variables is the same as in Table 1.

These 1215 cases were then used as the inputs for building a decision tree. First we had to decide which variables to use as attributes. Obviously the variables that are expected to have the influence on the computational complexity of the model should be used. However the number of variables and constraints alone is not a sufficient indicator of time complexity of the problem. Therefore we have chosen the following variables:
- *m*, *n*, *d* - obviously those variables have the influence on the size of the model as m and n influence the number of variables and constraints in the model, while *d* influence the number of possible combinations.
- v_1, v_2, u_1, u_2 were not included as absolute values but as part of the following ratios:
- *r* - the ratio between the average stock length and average order length $(u_1+u_2)/(v_1+v_2)$. Earlier it was statistically established that higher ratio means better solutions with heuristic method [10], however the influence of this ratio on exact solution method was not yet studied,
- *q* - the ratio between the available material and total needs. Problems with higher *q* should be easier to solve than those with this ratio closer to 1.

70% of the data was used as training, 30% as test data. To avoid over fitting of the data the test required two branches with at least 10 cases. The decision tree shown in Fig. 3 was generated using C5 program. The numbers in the brackets mean how many of the training cases belong to this leaf. The first number is the number of correctly and the second of incorrectly classified cases.

For example: the problems where the ratio between available material and total needs is greater than 2.18239 and number of stock lengths is less or equal to 7 the problem should be solved with exact method. Out of 128 test problems, that fulfill those conditions, a better solution with exact method was found in 126 cases. Other conditions can be explained similarly.

From Fig. 3 it is obvious that for this sort and size of the problem *q* has the greatest influence on the complexity of the problem, followed by *n*. On the other hand the influence of the number of order lengths and average demand per order length is surprisingly low. That finding was also used in the selection of sub problem for the C-CUT algorithm [15] where only number of stock lengths and partly the ratio between total material and total needs are pre-set for all sub problems while the number of order lengths and average demand per order length are determined on a case by case basis.

To avoid over fitting of the data the decision tree was tested on the problems mentioned in the first part of the paper. The comparison of the results between CUT and exact method was shown in Table 3. Obviously it would be possible to solve each problem with both methods and keep the best result. However that would require additional time and effort. On the other hand it is possible to solve each problem just once with the method chosen with decision tree.

Using decision tree we've got interesting results. The following cases were solved with the exact method: 1,4,7,11,12,13,16,19,21,22 and 25 while others were solved heuristically. The right decision was made in 21 out of 25 cases (2 cases have the same results with both methods). From the 4 mistakes 3 were resulted in only a marginally higher trim loss. In case 21 the trim loss was considerably higher (438 cm instead of 0).

The most important result is that the total trim loss would be 5593 cm if we would solve all problems with CUT, 32268 cm with exact method and 5310 cm if we solve each problem with the method chosen by the decision tree. This shows that proposed approach can indeed lead to improved results, compared to the results acquired with just one of the available methods.

The other advantage of proposed approach is that it takes both the processing power of the computer and the quality of the solver into account. Obviously the exact method would be selected more often if the experiments would be carried out on considerably faster computer or with better integer programming solver.

6. Conclusion

The article firstly examines the exact solution of G1D-CSP in cases with surplus and lack of material. B&B method and a problem generator *PGEN* for generation of G1D-CSP instances were used.

Three experiments are presented. In the first a better solution for previously published problem is shown. In the second the proposed method was tested by solving 270 problem instances. In the third the comparison with SHP CUT was made. We find out that our method gives better solution for smaller problems, the CUT procedure for larger. The proposed method also approximately shows how close the solutions are to the optimum, while CUT gives no such indication.

The new approach based on decision trees is introduced in order to establish which cutting method should be used. Using a decision tree an appropriate method can be chosen for each individual case based on its size and the probability that the problem of this size can be solved optimally within the given time limit.

The practical implementation of the approach was shown on the example of G1D-CSP, however it can be applied on other types of cutting problems as well. Numerous examples are calculated. The results show a high degree of certainty that the chosen method is the best for specific problem, which reflects in lower trim loss.

Remark. The problem generator PGEN may be obtained from the authors of this article both in source-code and as subroutine executable under Windows. The source code (in VBA) for all experiments, presented in this paper, is also available.

References

[1] Antonio J., Chauvet F., Chu C., Proth J., The cutting stock problem with mixed objectives: Two heuristics based on dynamic programming, European Journal of Operational Research 114 (1999) 395-402.

[2] Bischoff E. E, Wascher G., Cutting and Packing, European Journal of Operational Research 84 (1995) 503-505.

[3] Dyckhoff H., A typology of cutting and packing problems, European Journal of Operational Research 44 (1990) 145-159.

[4] Ferreira J. S., Neves M. A. and Fonseca P., A two-phase roll cutting problem, European Journal of Operational Research 44 (1990) 185-196.

[5] Gau T., Wascher G., CUTGEN1: A problem generator for the Standard One-dimensional Cutting Stock Problem, European Journal of Operational Research 84 (1995) 572-579.

[6] Gilmore P. C. and Gomory R. E., A linear programming approach to the cutting stock problem, Operations Research 9 (1961) 849-859.

[7] Gilmore P. C. and Gomory R. E., A linear programming approach to the cutting stock problem, Part II, Operations Research 11 (1963) 863-888.

[8] Goulimis C., Optimal solutions for the cutting stock problem European Journal of Operational Research 44 (1990) 197-208.

[9] Gradišar M., Jesenko J., Resinovič G., Optimization of roll cutting in clothing industry, Computer & Operation Research 24 (1997) 945-953.

[10] Gradišar M., Resinovič G., Jesenko J., Kljajić M.: A sequential heuristic procedure for one-dimensional cutting, European Journal of Operational Research 114 (1999) 557-68.

[11] Gradišar M., Kljajić M., Resinovič G., A hybrid approach for optimization of one-dimensional cutting, European Journal of Operational Research 119 (1999) 165-174.

[12] Gradišar M., Resinovič G., Kljajić M., Evaluation of algorithms for one-dimensional cutting, Working paper No. 90, Faculty of economics, University of Ljubljana, 1999.

[13] Gradišar M., Resinovič G., Kljajić M., Evaluation of algorithms for one-dimensional cutting, Computers & Operations Research 29 (2002) 1207-1220.

[14] Gradišar M, Trkman P., Indihar Štemberger M.: Exact solution of general one-dimensional cutting stock problem. Working paper No. 123, Faculty of Economics, University of Ljubljana, 2001

[15] Gradišar M., Trkman P.: A combined approach for solution of general one-dimensional cutting stock problem. Working paper No. 124, Faculty of Economics, University of Ljubljana, 2002.

[16] Haessler R. W., Vonderembse M. A., A procedure for solving the master slab cutting stock problem in the steel industry, AIIE Trans 11 (1979) 160-165.

[17] Quinlan J. R.: C 4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[18] Schilling G., Georgiadis M., C., An Algorithm for the Determination of Optimal Cutting Patterns, Computers & Operations Research 29 (2002) 1041-1058.

[19] Stadtler H., A one-dimensional cutting stock problem in the aluminium industry and its solution, European Journal of Operational Research 44 (1990) 209-23.

[20] Sweeney P. E. and Paternoster E. R., Cutting and packing problems: A Categorised, Application-Orientated Research Bibliography, Journal of the Operational Research Society 43 (1992) 691-706.

[21] Vance P.H., Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem,Computational optimization and applications 9 (1998) 211-28.

[22] Vanderbeck F., Computational Study of a Column Generation Algorithm for Bin Packing and Cutting Stock Problems, Research Papers in Management Studies, No 14, University of Cambridge, 1996.

[23] Wascher G., Gau T., Generating Almost Optimal Solutions for the Integer One-dimensional Cutting Stock Problem, Working Paper No. 94/06, Institut fur Wirtschaftswissenshaften, Technische Universitat Braunschweig, 1994.