Optimal Cutting Problem

Ana Avdzhieva, Todor Balabanov, Georgi Evtimov,
Detelina Kirova, Hristo Kostadinov, Tsvetomir Tsachev,
Stela Zhelezova, Nadia Zlateva

1. Problems Setting

One of the tasks of the Construction office of company STOBET Ltd is to create large sheets of paper containing a lot of objects describing a building construction as tables, charts, drawings, etc. For this reason it is necessary to arrange the small patterns in a given long sheet of paper with a minimum wastage.

Another task of the company is to provide a way of cutting a stock material, e.g. given standard steel rods, into different number of smaller sized details in a way that minimizes the wasted material.

2. Problems Description

Task 1

A large piece of paper with width $X = 1000$ mm and length $Y = 15000$ mm is given. Over this paper many small rectangles (corresponding to tables, charts, drawings, etc.) with dimensions (a_i, b_i), $i = 1, \dots, n$ should be arranged. The goal is to arrange the rectangles in such a way that they fill the entire width of the paper using minimal length of the sheet.

Until the study group session the company has no solution to this task.

Task 2

An unlimited stock of rods with standard section and constant length L meters is available. According to the construction chart, n_i number of details with length l_i meters each, $i = 1, \dots, m$ has to be cut from the stock rods. The purpose is to produce the desired quantities of details minimizing the wasted stock material.

The company handled the task using self developed algorithm which was slow and inefficient because it was based on exhaustive search.

3. Problems Identification

Both problems are identified as cutting-stock problems. In Operations Research, the cutting-stock problem is an optimization problem of cutting standard-sized pieces of stock material into pieces of specified sizes while minimizing material wasted. In terms of computational complexity, the problem is an NPcomplete problem reducible to the knapsack problem and it can be formulated as an integer linear programming (LP) problem. Task 2 is classic one dimensional (1D) cutting-stock problem while Task 1 is two dimensional (2D) cutting-stock problem which is more complex.

4. Formulation and Solution Approaches

Task 2: (1D) cutting-stock problem

The standard formulation for the (1D) cutting-stock problem (but not the only one) starts with a list of m orders, each requiring n_i pieces of length l_i, $i = 1, \dots, m$ that have to be cut from stock material (rod) of length L. We then construct a list of all possible combinations of cuts (often called "patterns"), associating with each pattern a positive integer variable x_j representing the number of stock material pieces to be slit using pattern j. The linear cutting-stock integer problem is then:

$$\begin{aligned} &\min \sum_j w_j x_j \\ &\sum_j a_{ij} x_j = n_i, \qquad \forall i = 1, \dots, m \\ &x_j \geq 0 \ \text{ and integer}, \end{aligned} \tag{1}$$

where a_{ij} is the number of times order i appears in pattern j and w_j is the cost (often the waste $w_j = L - \sum_i a_{ij} l_i$) of pattern j. In order for the elements a_{ij}, $i = 1, \dots, m$ to constitute a feasible cutting pattern, the following restriction must be satisfied

$$\begin{aligned} &\sum_{i=1}^m a_{ij} l_i \leq L, \\ &a_{ij} \geq 0 \ \text{ and integer}. \end{aligned}$$

Instead of problem (1) for minimizing the waste we prefer to consider the equivalent to it problem for minimizing the total number of utilized rods which is known also as bin packing problem:

$$\begin{aligned} &\min \sum_j x_j \\ &\sum_j a_{ij} x_j = n_i, \qquad \forall i = 1, \dots, m \\ &x_j \geq 0 \ \text{ and integer}. \end{aligned} \tag{2}$$

In general, the number of possible patterns grows exponentially as a function of m and it can easily run into the millions. So, it may therefore become impractical to generate and enumerate the possible cutting patterns.

An alternative approach uses column generation method. This method solves the cutting-stock problem by starting with just a few patterns. It generates additional patterns when they are needed. For the one-dimensional case, the new patterns are introduced by solving an auxiliary optimization problem called the knapsack problem, using dual variable information from the linear problem. There are well-known methods for solving the knapsack problem, such as branch and bound and dynamic programming. The column generation approach as applied to the cutting-stock problem was pioneered by Gilmore and Gomory in a series of papers published in the 1960s, see e.g. [2].

Modern algorithms can solve to optimality very large instances of the (1D) cutting-stock problem but their program implementations are in rule commercial and expensive. Some free software for finding approximate solutions are available.

Although the existing fast algorithm for finding the approximate solution is good enough, for specific purposes of the company STOBET Ltd. it was necessary to find an algorithm relevant to the data format they use which makes the usage of the available free software inadequate.

Task 1: (2D) cutting-stock problem

The formulation of a higher dimensional cutting-stock problem is exactly the same as that of the one dimensional problem given in (1) and (2). The only added complexity comes in trying to define and generate feasible cutting patterns. The simplest two dimensional case is one in which both the stock and ordered sizes are rectangular which is exactly our case.

5. Group Suggestions and Solutions

- For the 2D cutting-stock problem: we proposed a *genetic type algorithm.*
- For the 1D cutting-stock problem: we proposed a *greedy type algorithm.*

5.1. A Genetic Type Algorithm for Task 1: (2D) Cutting-Stock Problem

Genetic algorithms are a robust adaptive optimization method based on biological principles. A population of strings representing possible problem solutions is maintained. Search proceeds by recombining strings in the population. The theoretical foundations of genetic algorithms are based on the notion that selective reproduction and recombination of binary strings changes the sampling rate of hyperplanes in the search space so as to reflect the average fitness of strings that reside in any particular hyperplane. Thus, genetic algorithms need not search along the contours of the function being optimized and tend not to become trapped in local minima [3].

According to the given objective in Task 1, near optimal solution can be absolutely acceptable. In such situations heuristics is one of the most used ways in solving optimal cutting problems. The difficulties in 2D variation of the problem come from the need that all rectangles should be properly arranged in the x axis and in the y axis simultaneously.

Idea of the genetic algorithm (GA)

The proposed GA goes in the following steps:

```
1. Initialization;
   WHILE ( stop criteria is not met )
   BEGIN
2. Selection;
3. Crossover;
4. Mutation;
5. Evaluation;
   END
6. Report results.
```

When GAs (Genetic Algorithms) are applied there are three common operations to be implemented: problem data encoding, crossover, mutation and selection. In the case of 2D cutting problem we propose following parameters [1]:

- Problem data encoding is done by representing all rectangles with top/left coordinate, width, height and orientation. Rectangles are presented in stored ordered list as chromosome in the GA (known in the literature as permutation encoding);
- Single cut point crossover is implemented, applied by the rules of GA's permutation encoding;

- Mutation operator consists of random rotation of a rectangle into the resulting chromosome and random swap between two rectangles into the resulting chromosome;
- Selection operator is based on uniform random distribution, but result chromosome always replaces the worst chromosome in the population. By this selection approach elitism rule is applied indirectly.

Some modifications were done from the original form of the GAs. At first place, one part of the chromosomes are randomly shuffled, second part is descending ordered by rectangles width and third part is descending ordered by rectangles height in the initialization phase. At second place, one part of the chromosomes are filled with all rectangles in portrait orientation, second part of the chromosomes are filled with all rectangles in landscape orientation and third part or the chromosomes rectangles are in mixed orientation. All this modifications are done in order better genetic diversity to be presented during initialization part of the algorithm.

Fitness value evaluation is done by additional procedures for packing [1]. Because we do not keep information inside the chromosomes for the 2D placement of the rectangles, the additional packing procedure fill the drawing sheet of paper with proper amount of rectangles and after that the used length of paper is calculated and applied as chromosome fitness value. Packing procedure proposed in [1] is also near optimal. It does not search for the best optimal packing. Near optimal packing is fast enough and for better arrangements GA is responsible.

5.2. A Greedy Type Algorithm for Task 2: (1D) Cutting-Stock Problem

Let unlimited quantities of steel rods with different sections (types) having standard lengths L_k are given. Any such rod is said to be of type k. Let the cutting length l_c be given.

As input the details needed for the corresponding project are given. For each detail are known its label, section and length l_i. The number of items with these properties n_i that have to be cut is also known. In Table 1 is given a sample of real data provided by STOBET Ltd.

As output we need for any section (type) k of stock material the number of rods utilized for producing all items with section k, the way that any of these rods was cut (i.e. which items are cut from it and in which order) as well as the waste from any utilized rod and the total percentage of waste obtained from all utilized rods of section k.

Table 1. Sample data

label	section	number, n_i	length, (mm) l_i, mm	single weight (kg/m)	element weight (kg)	total weight (kg)
L windows 9	L 50 × 4	20	5790	3.0500	17.66	353.19
L windows 10	L 50 × 4	2	4912	3.0500	14.98	29.96
plank 20	PLATE 6 × 80	20	170		0.84	16.70
plank 33	PLATE 6 × 80	4	105		0.52	2.07
plank 36	PLATE 6 × 80	24	100		0.49	11.76
plank 47	PLATE 5 × 70	10	70		0.19	1.89
plank 48	PLATE 5 × 180	10	70		0.49	4.91
plank 49	PLATE 5 × 205	3	70		0.56	1.68
plank 50	PLATE 5 × 205	3	63		0.50	1.51
profile 25	100 × 80 × 5	19	5950	9.4900	56.47	1072.84
profile 26	SHS 100 × 4	3	5950	12.0000	71.40	214.20
profile 27	RHS 100 × 50 × 5	6	5800	10.9000	63.22	379.32
profile 28	RHS 100 × 50 × 5	6	5720	10.9000	62.35	374.09
profile 29	SHS 100 × 4	3	3020	12.0000	36.25	108.74
profile30	100 × 80 × 5	1	1000	9.4900	9.49	9.49
profile 31	SHS 100 × 4	6	780	12.0000	9.36	56.16
profile 40	EQA 70 × 7	2	70	7.3800	0.52	1.03
profile 41	EQA 70 × 7	45	55	7.3800	0.41	18.27
profile 42	SHS 100 × 4	6	3974	12.0000	47.69	286.13
profile 44	SHS 40 × 4	24	246	4.4600	1.10	26.33
profile 45	SHS 40 × 4	60	230	4.4600	1.03	61.56
profile 46	SHS 40 × 4	36	230	4.4600	1.03	36.94
profile 47	SHS 40 × 4	16	200	4.4600	0.89	16.06
profile 54	EQA 70 × 7	2	6995	7.3800	51.62	103.25
profile 55	EQA 70 × 7	2	6990	7.3800	51.59	103.17
profile 56	EQA 70 × 7	9	3880	7.3800	28.63	257.71
profile 57	EQA 70 × 7	2	3880	7.3800	28.63	57.27
profile 58	EQA 70 × 7	10	3880	7.3800	28.63	286.34
profile 59	EQA 70 × 7	1	3880	7.3800	28.63	28.63
profile 60	EQA 70 × 7	11	1675	7.3800	13.84	152.21
profile 61	EQA 70 × 7	2	1670	7.3800	13.80	27.60
profile 62	EQA 70 × 7	9	1670	7.3800	13.80	124.21

Idea of the Greedy Algorithm

Sort the details by their section (type).

For details with any particular section k:

(1) Set $L = L_k$ and make an ordered list S of all items sorted in descending order by their lengths. All items i with $l_i > L$ are labeled as impossible for cutting

Table 2. Details of section EQA 70×7

	detail	n_i	l_i, mm
1	profile 40	2	70
2	profile 41	45	55
3	profile 54	2	6995
4	profile 55	2	6990
5	profile 56	9	3880
6	profile 57	2	3880
7	profile 58	10	3880
8	profile 59	1	3880
9	profile 60	11	1875
10	profile 61	2	1870
11	profile 62	9	1870

(this is caused by constructive error and the input data have to be corrected) and they are excluded from S.

(2) Take a stock rod of section k. Denote its unused part (remainder) by L_r, so $L_r = L$.

(3) If $S \neq \emptyset$, then take from S the first item i with $l_i \leq L_r$, cut it from the rod and update the remainder $L_r = L_r - l_i - l_c$ and the list $S = S \setminus \{i\}$ (i.e. discard the item i from the list).

If $S = \emptyset$ then STOP.

If $L_r < \min_{i \in S} l_i$ then the cutting of the current rod is terminated, the waste is $l_w = L_r$ and go to (2). Otherwise, go to (3).

The proposed greedy algorithm provides approximate solution.

As a rough estimate for the minimal number of rods that have to be utilized for the solution, one may take the maximum of the numbers k_1 and k_2, where k_1 is the the smallest integer greater than or equal to the total length of all items divided by L, and k_2 is the number of items with length $l_i > L/2$.

For the sample data in Table 1 the approximate solution obtained by the proposed algorithm is presented in Table 5.

To illustrate how the greedy algorithm works on the sample data given in Table 1, we consider section EQA 70×7 which is known to be of length $L = 6000$ mm and recall the cutting length is $l_c = 5$ mm. In Table 2 there are 11 different details with this section.

The proposed greedy algorithm was implemented in JAVA, Lisp, MS Excel. Many experimental tests were made with significant amount of data provided by STOBET. The suggested greedy algorithm improved significantly the speed of finding an approximate solution. How near to the optimal solution is the

Table 3. List S of details of section EQA 70 × 7

detail	section	n_i	l_i, mm
profile 54	EQA 70 × 7	2	6995
profile 55	EQA 70 × 7	2	6990
profile 56, 57, 58, 59	EQA 70 × 7	22	3880
profile 60	EQA 70 × 7	11	1675
profile 61,62	EQA 70 × 7	11	1670
profile 40	EQA 70 × 7	2	70
profile 41	EQA 70 × 7	45	55

obtained one depends of the distribution of the particular item lengths. Because of the nature of the company projects this distribution is in most cases even. For the sample data it is shown in Figure 1.

Table 4. Solution for section EQA 70 × 7 with $L = 6000$ mm

number of rods	pattern	3880	1675	1670	70	55	wastage, l_w
1	t_1	1	1	0	2	4	45
5	t_2	1	1	0	0	7	15
1	t_3	1	1	0	0	6	75
4	t_4	1	1	0	0	0	435
11	t_5	1	0	1	0	0	440

For convenience, details with the same length are considered as details of one and the same type. Of course, they keep their own different labels and after the cutting is completed it is clear from which rod and at which place on it they were cut.

In this example all items with length 3880 mm are considered as one kind of detail and the same is for items with length 1670 mm. So, finally we have 7 details and 95 items, see Table 3.

If there are some $l_i > L$ the algorithm excludes them from consideration. Here this is the case for details labeled "profile 54" and "profile 55".

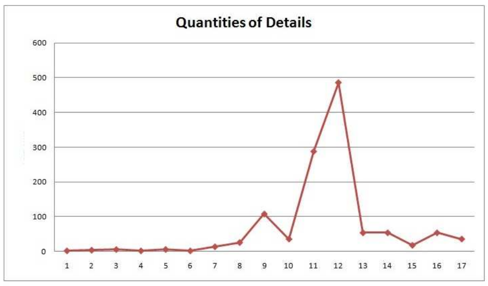
So, we start with the first rod. After cutting one item of third length $l_3 = 3880$ mm, the remainder of the rod becomes $L_r = 2115$ mm. The first item in S with length $l_i \leq L_r$ is 1675 mm and we cut it from the rod. Then the remainder becomes $L_r = 435$ mm. We cut one item with $l_6 = 70$ mm and so on. At the end for the first rod we obtain remainder $L_r = 45$ mm which is smaller than the smallest item length and therefore the wastage for this rod is $l_w = 45$ mm. For the considered example the obtained cutting patterns, the number of rods used for each pattern and the waste obtained in using that pattern are given in Table 4.

Total number of the utilized for the solution rods of section EQA 70×7 is 22. In this particular case we succeed in obtaining an exact solution. Indeed, as it can be seen from the values of k_1 and k_2 for this section in Table 5 the number of utilized rods can not be smaller than 22 and for our solution we utilize exactly 22 rods. In fact, for all sections in the sample data we obtain exact solution as it can be seen from Table 5.

The approximate solution obtained by the previously used by STOBET Ltd algorithm for this sample data utilized 30 rods.

Table 5. Utilized rods for the obtained by greedy algorithm solution for data in Table 1

section	k_1	k_2	number of utilized rods	$w, \%$
L 50 × 4	21	22	22	4,75
PLATE 6 × 80	2	0	2	46.17
PLATE 5 × 70	1	0	1	87.50
PLATE 5 × 180	1	0	1	87.50
PLATE 5 × 205	1	0	1	92.85
100 × 80 × 5	20	20	20	48.75
SHS 100 × 4	10	12	12	22.88
RHS 100 × 50 × 5	12	12	12	3.96
SHS 40 × 4	6	0	6	11.49
EQA 70 × 7	21	22	22	5.13



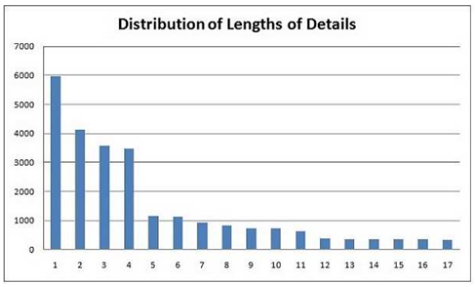


Fig. 1. Items quantities and length distribution

For solving (1D) cutting-stock problem we also applied the developed for the (2D) cutting-stock problem genetic algorithm. Doing this for the sample data the approximate solutions obtained from both algorithms (greedy and genetic) utilized the same number of rods but, of course, in some cases they differ in patterns of cuts.

Further Improvement of the Greedy Algorithm

A rod with a big remaining L_r will appear in few cases in the obtained by the greedy algorithm approximate solution in cutting rods with stock length L. If L_r of the last utilized rod is grater than $0,7L$ then it is relevant to try to permute the items in some of the patterns in order to cut the few items from the last rod from the other utilized rods.

For the obtained approximate solution, the waste from each utilized rod is known. The number and the lengths of details cut from the last rod are also known. Let their lengths be $d_1 > d_2 > d_3 > \cdots$.

Create a list $\overline{S}$ of all utilized rods (except the last one) ordered by their appearance in the solution.

Details are already sorted by length in descending order. For detail i with length l_i we seek a detail j with length $l_j > l_i$ such that the permutation of one item of length l_i and one item of length l_j between two different rods from $\overline{S}$ is possible. Set $\varepsilon = l_j - l_i$, and call a rod from where l_i is cut rod i, and a rod from where l_j is cut rod j. Let the waste of rod i be w_i, and the waste of rod j be w_j. If we permute l_i and l_j between rod i and rod j, then w_i will become $w_i + \varepsilon$, while w_j will become $w_j - \varepsilon$. The permutation (i, j) is possible if $w_j - \varepsilon \geq 0$. A possible permutation (i, j) is useful if $w_i \geq d_k$ for some d_k because after the permutation an item of this length (or smaller) can be cut from rod i instead from the last rod. If we succeed in finding enough useful permutations, the cut of the last rod will be saved.

Preliminary part:

Find the possible permutations:

For all detail i with length l_i find the maximal waste from all rods in which cutting item with length l_i is involved and denote it as $\overline{w}_i$.

For any detail i with length l_i set $j = i + 1$ and create the triple $(\overline{w}_i, \overline{w}_j, l_j - l_i)$. If $\overline{w}_j \geq l_j - l_i$ the permutation (i, j) is labeled as possible and the next couple $(i, j + 1)$ is considered, if not no more j are considered for possible permutation with i.

Find the useful permutations:

For any possible permutation (i, j) the smallest k with $\overline{w}_i + l_j - l_i \geq d_k$ will be used for labeled it as k-useful. If such k does not exists the permutation is useless.

Main part:

For item with length d_1, if there is no 1-useful permutation the algorithm stops. If not, 1-useful permutation (i, j) is done. As a result one item with length d_1 is cut from rod i. Rods i and j have to be deleted from $\overline{S}$. Maximal

Table 6. Initial data for RHS example with $L = 6000$ mm and $l_c = 5$ mm

section	n_i	l_i, mm
RHS $100 \times 50 \times 4$	2	5975
RHS $100 \times 50 \times 4$	4	4135
RHS $100 \times 50 \times 4$	6	3575
RHS $100 \times 50 \times 4$	2	3473
RHS $100 \times 50 \times 4$	6	1153
RHS $100 \times 50 \times 4$	2	1127
RHS $100 \times 50 \times 4$	14	910
RHS $100 \times 50 \times 4$	26	810
RHS $100 \times 50 \times 4$	108	716
RHS $100 \times 50 \times 4$	36	715
RHS $100 \times 50 \times 4$	288	612
RHS $100 \times 50 \times 4$	486	365
RHS $100 \times 50 \times 4$	54	350
RHS $100 \times 50 \times 4$	54	340
RHS $100 \times 50 \times 4$	18	334
RHS $100 \times 50 \times 4$	54	330
RHS $100 \times 50 \times 4$	36	320

wastes $\overline{w}_i$ and $\overline{w}_j$ have to be recalculated if necessary and the labels of (i, j) have to be corrected if necessary. This is repeated for all items with d_1, then for items with d_2, etc.

At the end, either there are left some items with length d_k and no k-useful permutations are available in which case we keep the former approximate solution, or all items with lengths d_k are cut in result of permutations and the obtained in that way approximate solution is better than the former one (number of utilized rods is decreased by 1).

In place of the main part, the genetic algorithm can be used. When the initial set of useful permutations (i, j) is identified, then the index set of details I can be divided in two parts – I_1 will be the indexes of details involved in useful permutations, and I_2 will be $I \setminus I_1$.

Then the utilized rods $\overline{S}$ can be divided into two parts: $\overline{S}_1$ consists of all rods from which is cut at least one detail i involved in useful permutation and $\overline{S}_2 = \overline{S} \setminus \overline{S}_1$. We need only to permute items which are in rods from $\overline{S}_1$. Hence, input of the genetic algorithm will be all items cut from rods from $\overline{S}_1$ as well as the items cuts from the last rod. The rods from $\overline{S}_2$ and the rods obtained after the greedy algorithm is applied will form the new and possibly better approximate solution. If the result obtained by the genetic algorithm is not satisfactory one

Table 7. Input data for genetic algorithm used for re-optimization in RHS example

section	n_i	l_i, mm
RHS 100 × 50 × 4	5	3575
RHS 100 × 50 × 4	2	3473
RHS 100 × 50 × 4	2	1127
RHS 100 × 50 × 4	14	910
RHS 100 × 50 × 4	26	810
RHS 100 × 50 × 4	108	716
RHS 100 × 50 × 4	36	715
RHS 100 × 50 × 4	2	612
RHS 100 × 50 × 4	4	365
RHS 100 × 50 × 4	2	320

can enlarge the input adding in relevant way some possible permutations.

To illustrate the above, consider RHS example. Data are given in Table 6.

The greedy algorithm obtains approximate solution for which the number of utilized rods is 109, and the total waste is 18836 mm, or 2.88%. The solution is such that from the last rod only two items with length 320 mm are cut.

Permutations $(1127, 910)$, $(910, 810)$, $(910, 716)$, $(910, 715)$, $(810, 716)$, $(810, 715)$ are identified as useful. This means that as input for the genetic algorithm we have to consider the items cut from rods numbered from 8 to 36 in the former solution and the two items with length 320 mm. The input for the genetic algorithm for RHS example is given in Table 7. The expected result of 29 rods is achieved and the re-optimization was successful in this case. After re-optimization the number of rods utilized for the new approximate solution is 108, the total waste is 12836 mm, or 1.96%.

6. Feedback from STOBET Ltd.

The company has already successfully applied the proposed algorithms and solutions. STOBET Ltd reported:

- Task 1: The Genetic type algorithm is very powerful. For short time we can obtain sufficiently good results. We will built-in this method in our practice.

- Task 2: The greedy type algorithm is very fast and it is sufficiently good for many cases. Since the proposed greedy algorithm finds approximate solution it is possible in few cases to get not the most economical result, i.e. the exact solution. Therefore we can apply proposed re-optimization or we can use memoization in dynamic optimization.

Acknowledgements. Thanks are due to Prof. Nikola Yanev for presenting to the group a thorough review of the scientific achievements in the field.

References

[1] IICT-BAS, ESGI113, problem 3, a genetic algorithm solver, `https://github.com/TodorBalabanov/ESGI113Problem3GeneticAlgorithmSolver`

[2] D. Goldfarb, M. Todd. Chapter II. Linear programming, In: Handbooks in OR & MS, vol. 1 (Eds. G. L. Nemhauser et al.), Elsevier, 1989.

[3] D. Whitley, T. Starkweather, C. Bogart, Genetic algorithms and neural networks: optimizing connections and connectivity, *Parallel Computing* Volume 14, Issue 3 (1990) 347-361.