MATEC Web of Conferences **117**, 00050 (2017) DOI: 10.1051/matecconf/201711700050
XXVI R-S-P Seminar 2017, Theoretical Foundation of Civil Engineering

Spreadsheet solutions of the one-dimensional cutting stock problem

Robert Gajewski[1,*]

[1] Warsaw University of Technology, Faculty of Civil Engineering, Al. Armii Ludowej 16, Warsaw 00-637, Poland

Abstract. The cutting stock problem encompasses cutting parts available in stock, which are called objects, to produce in specified quantities smaller pieces which are called items and optimizing an objective function. The cutting stock problem is common for many industrial processes – steel, paper tube and construction industries. One of the most important phases in solution of the one-dimensional cutting stock problem is a pattern generating procedure which can be performed in a spreadsheet.

1 Introduction and literature review

The cutting stock problem encompasses cutting parts available in stock, which are called objects, to produce in specified quantities smaller pieces which are called items and optimizing an objective function. Objective functions comprise minimizing the total waste, minimizing the cost of cutting the objects, minimizing the total number of objects cut, maximizing profit and minimizing production costs. Sample solution of this problem is called cutting plan. Such plan is provided by a set of cutting patterns and corresponding frequencies which describe how many times each pattern should be used to produce the items, see Fig. 1. Each cutting pattern defines a subset of items which will be cut from an object. More information about this can be found in surveys [1] and [2].

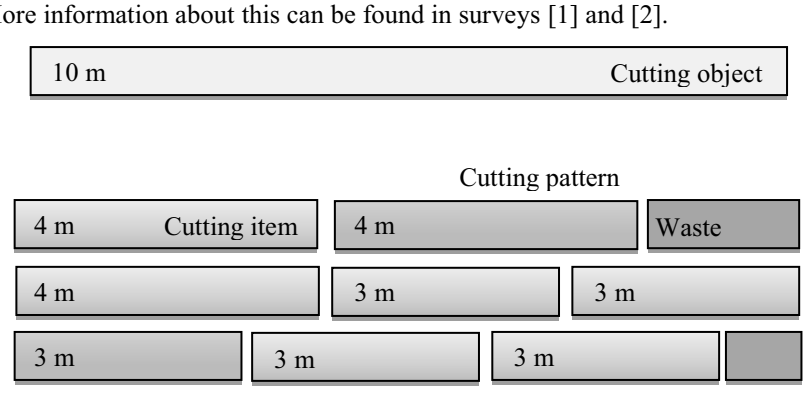

Fig. 1. Cutting plan: cutting object, items, patterns and wastes.

* Corresponding author: r.gajewski@il.pw.edu.pl

The cutting stock problem is common for many industrial processes. Armbruster described in [3] a solution procedure for a pattern sequencing problem as part of a one-dimensional cutting stock problem in the steel industry. In [4] Belov presented a cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. Cherri [5] introduced a heuristic approach to the one-dimensional cutting stock problem with usable leftover. Gau [6] created a problem generator for the standard one-dimensional cutting stock problem. Matsumoto described in [7] the one-dimensional stock cutting problem in the paper tube industry. Ogunranti discussed in [8] how to minimize waste (off-cuts) using cutting stock model for the case of one dimensional cutting stock problem in wood working industry. Poldi presented in [9] heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. In [10] Shahin used genetic algorithms to solve the one-dimensional cutting stock problem in the construction industry. In [11] Stadtler described the one-dimensional cutting stock optimization with usable leftover - a case of low stock-to-order ratio. In thesis [12] Tomat discussed one-dimensional cutting stock optimization with usable leftover - a case of low stock-to-order ratio. In [13] Umetani solved one-dimensional cutting stock problem to minimize the number of different patterns. In [14] Vance used branch-and-price algorithms for the one-dimensional cutting stock problem.

One of the most important phases in solution of the one-dimensional cutting stock problem is a pattern generating procedure presented by Suliman in [15]. Similar approach was presented by Goulimis [16]. This work was based on enumerating the possible cutting patterns, and solving the associated integer program by a combination of cutting planes and branch and bound.

2 Search tree

In order to generate feasible cutting patterns a search tree can be used [15]. This procedure is described by Suliman as follows: “The search tree is constructed by traversing first from bottom root to top and then from left to right. The construction starts at the root (first level of the tree), and continues to move upward in the tree by adding additional sizes to the combinations already specified by the previous branches. While traversing along a specific path on the search tree, the feasibility of that path is maintained by ensuring a non-negative end terminal node which implies adequate material to satisfy the requirements of the path. Each node at any level has as many branches as the maximum number of units that can be cut from the remaining width.”

All calculations for a simple case can be performed by hand. Let us consider element which length is 9 units. We want to cut from it pieces which lengths are 5, 4, 3 and 2 units respectively. For this case search tree can be constructed manually.

- How many 5 unit’s pieces can be cut? Only one. The remaining part is 4 units.
- How many 4 unit’s pieces can be cut from the remaining part which has 4 units? Exactly one and remaining part will be 0.
- How many 3 unit’s pieces can be cut from the remaining part which has 4 units? Also one and remaining part will be 1.
- How many 2 unit’s pieces can be cut from the remaining part which has 4 units? Exactly two and remaining part will be 0.
- How many 4 unit’s pieces can be cut? Two. Remaining part is 1 unit.
- When we start from cutting one 4 unit’s piece we have remaining part which has 5 units.

We can continue this and all results can be summarized in a table (see Fig. 2).

5	1	1	1	0	0	0	0	0	0	0
4	1	0	0	2	1	1	0	0	0	0
3	0	1	0	0	1	0	3	2	1	0
2	0	0	2	0	1	2	0	1	3	4
	0	1	0	1	0	1	0	1	0	1

Fig. 2. Cutting object, items, pattern, plan and wastes.

This solution can also be presented as a tree (see Fig. 3).

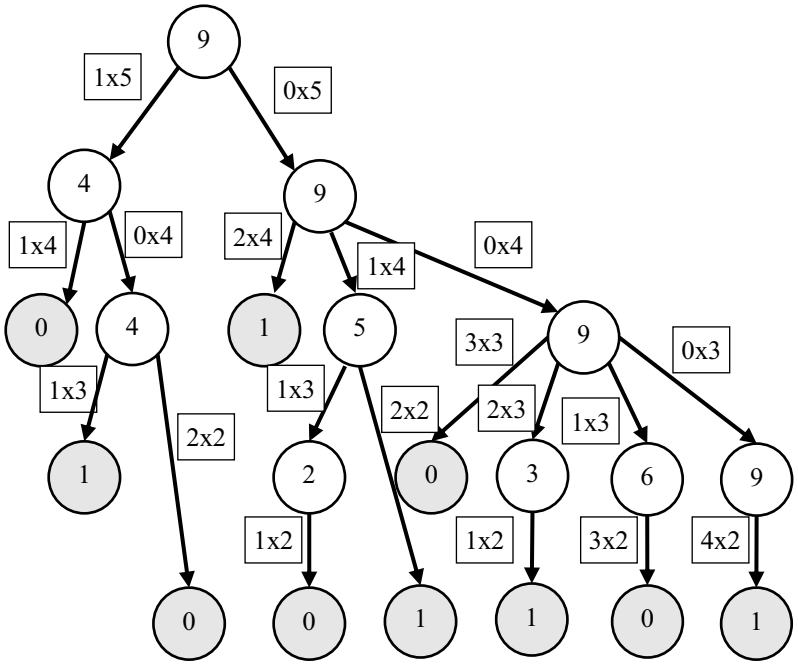

Fig. 3. Sample search **tree**.

It is hardly possible to perform all calculations even for slightly more complicated example. When we want to cut pieces which lengths are 5, 4, 3 and 2 units respectively from element which length is 18 units number of different cutting patterns is 40 and algorithmic approach is necessary.

3 Pattern generation algorithm

All details of the pattern generation algorithm can be found in [15]. Let us denote by w the length of the element and by w_1, w_2,…w_n lengths of pieces which should be cut from it. We assume that $w_1>w_2>\ldots>w_n$. For each problem matrix **A** with number of elements nxm should be calculated, where m denotes number of different cutting patterns. The first element is given by:

$$a_{11} = \mathrm{div}\left(\frac{w}{w_1}\right) \tag{1}$$

The second element in the first column is:

$$a_{21} = \mathrm{div}\left(\frac{w - a_{11} \cdot w_1}{w_2}\right) \tag{2}$$

The *i*-th element in the first column is:

$$a_{i1} = \mathrm{div}\frac{w - \sum_{k=1}^{i-1} a_{k1} \cdot w_k}{w_i} \tag{3}$$

Each element in the matrix is given by:

$$a_{ij} = \mathrm{div}\frac{w - \sum_{k=1}^{i-1} a_{kj} \cdot w_k}{w_i} \tag{4}$$

We can check these formulas for the given example and compare them with Fig. 2.

$$\begin{aligned} a_{11} &= \mathrm{div}\left(\frac{\mathrm{w}}{\mathrm{w}_1}\right) = \mathrm{div}\left(\frac{9}{5}\right) = 1 \\ a_{21} &= \mathrm{div}\left(\frac{w - a_{11} \cdot w_1}{w_2}\right) = \mathrm{div}\left(\frac{9 - 5 \cdot 1}{4}\right) = 1 \\ a_{31} &= \mathrm{div}\frac{w - \sum_{k=1}^{2} a_{k1} \cdot w_k}{w_i} = \mathrm{div}\frac{9 - 5 \cdot 1 - 4 \cdot 1}{3} = 0 \\ a_{41} &= \mathrm{div}\frac{w - \sum_{k=1}^{3} a_{k1} \cdot w_k}{w_i} = \mathrm{div}\frac{9 - 5 \cdot 1 - 4 \cdot 1 - 3 \cdot 0}{2} = 0 \end{aligned} \tag{5}$$

More information about algorithm and its flowchart can be found in [15].

4 From algorithm to program

In [15] a special computer routine for generating cutting patterns was develop which requires knowledge of programming language. This problem can be solved directly inside spreadsheet without programming in Visual Basic for Application. Such approach is not known in literature. It enables fast and robust solution of the whole problem in a spreadsheet. Spreadsheet formulas were tested on many examples. Solution of one of them is presented in Fig. 4. Length of element is 18 units and it is cut into pieces which length is 5, 4, 3 and 2 units respectively. In this case, there are 40 different cutting patterns. This table can also be calculated by hand but this process is time consuming and can be a source of mistakes.

5	3	3	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	2	1	1	0	0	0	3	2	2	1	1	1	1	0	0	0	0	0
3	1	0	0	1	0	2	1	0	0	1	0	3	2	1	0	4	3	2	1	0
2	0	1	0	0	2	1	2	4	0	1	2	0	1	3	4	0	2	3	5	6
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	3	3	3	2	2	2	2	1	1	1	1	1	0	0	0	0	0	0	0
	0	2	1	0	3	2	1	0	4	3	2	1	0	6	5	4	3	2	1	0
	1	0	1	3	0	2	3	5	1	2	4	5	7	0	1	3	4	6	7	9

Fig. 4. Cutting plan.

5 Solver solution

In the next step Linear Programming problem is formulated and solved by spreadsheet solver. Two models can be built for such problem. We will denote by x_i number of elements cut according to *i*-th pattern.

In the first model, we minimize only wastes – we do not take into account that number of cut pieces can be greater than required so in all constraints we will have condition ≥. In the second model, we will use additional decision variables y_j denoting number of elements having length w_j cut above the requirements. In this case, we will subtract value of y_j in each constraint and use equality condition. Moreover, in the goal function we will add the total length of these additional elements.

Example which is solved has element which length is 13 units cut into pieces which have length 5, 4, 3 and 2 units. Data from cutting plan can be directly used to build spreadsheet model of the problem.

	_1	_2	_3	_4	_5	_6	_7	_8	_9	10	11	12	13	14	15	16	17	18	19	20	Su	No
5	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	12	12
4	0	0	2	1	1	0	0	0	3	2	2	1	1	1	1	0	0	0	0	0	19	13
3	1	0	0	1	0	2	1	0	0	1	0	3	2	1	0	4	3	2	1	0	14	14
2	0	1	0	0	2	1	2	4	0	1	2	0	1	3	4	0	2	3	5	6	15	15
W	0	1	0	1	0	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1	0	
R	2	0	8	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	3	0	24	

Fig. 5. Solution of the first model.

	_1	_2	_3	_4	_5	_6	_7	_8	_9	10	11	12	13	14	15	16	17	18	19	20	y1	y2	y3	y4	Su	No
5	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	12	12
4	0	0	2	1	1	0	0	0	3	2	2	1	1	1	1	0	0	0	0	0	0	-1	0	0	13	13
3	1	0	0	1	0	2	1	0	0	1	0	3	2	1	0	4	3	2	1	0	0	0	-1	0	14	14
2	0	1	0	0	2	1	2	4	0	1	2	0	1	3	4	0	2	3	5	6	0	0	0	-1	15	15
W	0	1	0	1	0	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1	5	4	3	2	11	
R	4	1	1	1	1	0	0	0	0	3	0	1	0	0	2	0	1	0	0	0	1	0	0	1		

Fig. 6. Solution of the second model.

Results for the first and second model are different. In the first model value of goal function is 11 but total length of not necessary elements is 24. In the second model toal number of unnecessary elements is 11.

6 Final conclusions

This paper presents spreadsheet implementation of pattern generation algorithm not known in the literature of the subject. This approach does not require knowledge of programming languages. All operations on matrix elements are performed directly in a spreadsheet.

Excel solver has well known limitations. Number of decision variables cannot exceed 200 for both linear and nonlinear problems. Moreover, using basic Excel Solver one can place constraints on up to 100 cells which are not decision variables. In such a case one can buy Frontline's Premium Solver or use Open Solver, the Open Source linear, integer and nonlinear optimizer for Microsoft Excel.

This research was not sponsored by any grant or authorities. This research was done just for pleasure and satisfaction.

References

1. A. C. Cherri, M. N. Arenales, H. H. Yanasse, K. C. Poldi, and A. C. Gonçalves Vianna, Eur. J. Oper. Res. **236**, 395 (2014)
2. C. H. Cheng, B. R. Feiring, and T. C. E. Cheng, Int. J. Prod. Econ. **36**, 291 (1994)
3. M. Armbruster, Eur. J. Oper. Res. **141**, 328 (2002)
4. G. Belov and G. Scheithauer, Eur. J. Oper. Res. **141**, 274 (2002)
5. A. C. Cherri, M. N. Arenales, and H. H. Yanasse, Eur. J. Oper. Res. **196**, 897 (2009)
6. T. Gau and G. Wäscher, Eur. J. Oper. Res. **84**, 572 (1995)
7. K. Matsumoto, S. Umetani, and H. Nagamochi, J. Sched. **14**, 281 (2011)
8. G. A. Ogunranti and A. E. Oluleye, J. Ind. Eng. Manag. **9**, 834 (2016)
9. K. C. Poldi and M. N. Arenales, Comput. Oper. Res. **36**, 2074 (2009)
10. A. A. Shahin and O. M. Salem, Can. J. Civ. Eng. **31**, 321 (2004)
11. H. Stadtler, Eur. J. Oper. Res. **44**, 209 (1990)
12. L. Tomat, *One-Dimensional Cutting Stock Optimization with Usable Leftover: A Case of Low Stock-to-Order Ratio* (Doctoral Dissertaton, University of Lublijana, Faculty of Economics, 2013)
13. S. Umetani, M. Yagiura, and T. Ibaraki, Eur. J. Oper. Res. **146**, 388 (2003)
14. P. H. Vance, Comput. Optim. Appl. **9**, 211 (1998)
15. S. M. A. Suliman, Int. J. Prod. Econ. **74**, 293 (2001)
16. C. Goulimis, Eur. J. Oper. Res. **44**, 197 (1990)