Description

Muamer Hrncic

July 2020

1 Waste Minimization

1.1 Problem description

The problem is known as the one dimensional cutting stock problem. The problem belongs to the class of NP-Complete problems. A detailed description of the problem and its solution can be found e.g. in [1].

A sawmill has raw material of length L at its disposal. An order of made-to-measure wood contains data of wood products. Let $I = \{1, \ldots, k\}$ the indexing of the different lengths to be cut. Of each length l_i a certain quantity n_i is ordered. It is assumed that $l_i \leq L, \forall i \in I$. The aim is to find a cutting plan with minimum waste for the given order. There are no further restrictions for the cutting process.

1.2 Model

There are different modeling approaches and good heuristic methods for this problem. The following modeling approach has also proven itself for larger instances.

The model is implemented with the modeling language AMPL. The enumeration procedure of the patterns is implemented in C# and was prepared in such a way that it creates a finished .dat file for the AMPL model on the basis of the input data. The model was tested on the NEOS SERVER [2, 3, 4] with different input data using the solver framework SCIP [5].

First, all feasible cutting patterns are enumerated for a given order. The index set of the patterns is $J = \{1, \ldots, m\}$, $m \in \mathbb{N}$. A single pattern is a vector $a_{*,j} = (a_{1,j}, \ldots, a_{k,j})^T$ of positive integers. The length l_i in this pattern is cut $a_{i,j}$ times. A pattern $a_{*,j}$ is feasible, if

$$r_j = \sum_{i=1}^{k} a_{i,j} \cdot l_i$$

holds the limits $0 \leq r_j \leq L$. All feasible patterns are stored as the columns of a matrix A. The integer variable x_j indicates how often pattern $a_{*,j}$ is used in a cutting plan. The following table serves as an example, where the raw material length $L = 10$.

Cutting patterns part 1												
l_i	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}	a_{09}	a_{10}	a_{11}	a_{12}
4	2	2	1	1	1	1	1	1	1	0	0	0
3	0	0	2	1	1	0	0	0	0	3	2	2
2	1	0	0	1	0	3	2	1	0	0	2	1
r_j	0	2	0	1	3	0	2	4	6	1	0	2

Cutting patterns part 2												
l_i	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{22}	a_{23}
4	0	0	0	0	0	0	0	0	0	0	0	0
3	2	1	1	1	1	0	0	0	0	0	0	0
2	0	3	2	1	0	5	4	3	2	1	0	0
r_j	4	1	3	5	7	0	2	4	6	8	10	0

Cutting stock

Input data:

- $i \in I = \{1, \dots, k\}$...product i
- $m \in \mathbb{N}$
- $j \in J = \{1, \dots, m\}$...pattern j
- $l_i \in \mathbb{Z}_+$...length of product i
- $n_i \in \mathbb{Z}_+$...amount of product i
- $L \in \mathbb{Z}_+$...raw material length
- $A = (a_{ij}) \in \mathbb{Z}^{k \times m}$, where a_{ij}...occurrence of l_i in pattern j

Variables:

- $x_j \in \mathbb{Z}_+$...usage of pattern j

Model:
$\min \sum_{j=1}^{m} x_j$
s.t.

(1) $\sum_{j=1}^{m} x_j \cdot a_{i,j} \geq n_i, \forall i \in I$

To solve the problem, the number of patterns used is minimized. The constraint is to be understood as follows:

(1) The requirement n_i of length l_i must be met.

1.3 Heuristics

To determine an optimal cutting plan in a reasonable time on the basis of an inquiry or order, an order must not become too extensive with regard to the number of types and quantities. Time also plays an important role in this problem. With increasing diversity, the time required to find a exact solution also grows rapidly.

In order to keep the effort to find a solution and the benefit in this case reasonably balanced, it is sufficient to find good or appropriate approximate solutions for such inquiries and orders within a reasonable time. This can be achieved with heuristic methods.

For the Cutting-Stock-Problem there is a multiplicity of such procedures. The following two procedures are presented, which were programmed and tested in C#. The input data, the variables and the initialization part are the same for both algorithms.

Input data:

- $i \in I = \{1, ..., k\}$...product i
- $l_i \in \mathbb{Z}_+$...length of product i, where $l_1 \geq l_2 \geq \ldots \geq l_k$
- $n_i \in \mathbb{Z}_+$...amount of product i
- $\lambda = (l_{1_1}, \ldots, l_{1_{n_1}}, \ldots, l_{k_{n_k}})$
- $L \in \mathbb{Z}_+$...raw material length

Output:

- $z \in \mathbb{N}$...number of raw material bars used
- $p \in \mathbb{N}^m$...map of which l_i is cut from which raw material bar

Initialization:

$i^* \leftarrow 1$
$j \leftarrow 1$
$m \leftarrow \sum_{i=1}^{k} n_i$
$\lambda \leftarrow (l_{1_1}, \ldots, l_{1_{n_1}}, \ldots, l_{k_{n_k}})$
$l_{used} \leftarrow (0, \ldots, 0)$
$p \leftarrow (1, 0, \ldots, 0)$

The entries of the vector λ consist of the lengths l_i of the request or order. Every entry occurs n_i times. The j-th entry of l_{used} indicates the current length consumption of bar j. The algorithms differ in the assignment of the product lengths to the raw material bars. In this context a raw material bar l_{used_j} is called:

1. not open, if $l_{used_j} = 0$,
2. open, if $l_{used_j} > 0$ and
3. full, if $L - l_{used_j} < \min\limits_{1 \leq i \leq k} l_i$.

First-Fit-Decreasing heuristic:

while $i^* \leq m$ **do**
  $j \leftarrow \min\{j \leq m | l_{used_j} + \lambda_{i^*} \leq L\}$
  $p_{i^*} \leftarrow j$
  $l_{used_j} \leftarrow l_{used_j} + \lambda_{i^*}$
  $i^* \leftarrow i^* + 1$
end while
$z \leftarrow \max_{1 \leq i^* \leq m} p_{i^*}$

Best-Fit-Decreasing heuristic:

while $i^* \leq m$ **do**
  $j \leftarrow \max\{j \leq m | l_{used_j} + \lambda_{i^*} \leq L\}$
  $p_{i^*} \leftarrow j$
  $l_{used_j} \leftarrow l_{used_j} + \lambda_{i^*}$
  $i^* \leftarrow i^* + 1$
end while
$z \leftarrow \max_{1 \leq i^* \leq m} p_{i^*}$

Each algorithm considers all opened raw material bars in each step of the loop. The currently considered product length λ_i^* is assigned as follows. In the First-Fit-heuristic it is assigned to the first bar into which it fits, and in the Best-Fit-heuristic it is assigned to the bar where the smallest remaining length in the bar results from the addition of λ_i^*. In case of ambiguity, the bar with the smallest index is selected.

If there is no space for λ_i^* in any of the opened raw material bars, or if all opened bars are full, a new bar is opened. The selected bar is saved for the current product length λ_i^* in the i^*-th entry of vector p.

After the last step p contains a suitable assignment for all i^* and the largest entry in p is the number of raw material bars to use. We now present a simple example of how the two heuristics work.
$l = (20, 17, 17, 14, 12, 7, 7, 7, 6, 6, 5, 3, 2)$ is an order of lengths that need to be cut. The sawmill has raw material with length $L = 31$.

Raw material	FFD Cutting Scheme					Rest
bar 1	20			7	3	1
bar 2	17			14		0
bar 3	17			12	2	0
bar 4	7	7	6	6	5	0

Raw material	BFD Cutting Scheme					Rest
bar 1	20			7	3	1
bar 2	17			14		0
bar 3	17			12	2	0
bar 4	7	7	6	6	5	0

Figure 1: Comparison of the FFD and BFD heuristic

The example shows, that both heuristics result in 4 bars. The cutting stock model also results in 4 bars. When testing with unsorted lengths, the BFD-heuristic delivered better results on average, but took more time. In the case that the lengths are sorted, the FFD-heuristic was beneficial.

1.3.1 Test Instances and Results

In this section a number of test instances and their results are presented. The instances are tested with the model and heuristics. The vector l denotes the lengths and c denotes the amounts. The instances look as follows:

Instance CS1:
$l = 6000, 5500, 5200, 4800, 4400, 3500, 2500, 2400, 1700, 1150$
$c = 200, 150, 220, 100, 100, 264, 80, 156, 50, 150$
$L = 12000;$

Instance CS2:
$l = 4200, 3900, 3650, 3200, 2980, 2700, 2200, 2050, 1890, 1560, 1200, 1100$
$c = 120, 120, 120, 120, 120, 120, 120, 120, 120, 120, 120, 120$
$L = 13800$

Instance CS3:
$l = 9000, 7800, 7600, 4200, 3900, 3650, 3200, 2980, 2800, 2700,$
$2200, 2050, 1890, 1750, 1560, 1480, 1450, 1300, 1200, 1100$
$c = 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,$
$300, 300, 300, 300, 300, 300, 300, 300$
$L = 15200$

Instance CS4:
$l = 9000, 7800, 7600, 4200, 3900, 3650, 3200, 2980, 2800, 2700,$
$2200, 2050, 1890, 1750, 1560, 1480, 1450, 1300, 1200, 1100$
$c = 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500,$
$500, 500, 500, 500, 500, 500, 500, 500$
$L = 17000$ The solving times are given in seconds.

Overview Results				
Instance	cutting patterns	used bars model	used bars FFD	used bars BFD
CS1	355	491	508	508
CS2	2222	267	273	273
CS3	11022	1260	1282	1282
CS4	14794	1877	1908	1908
Instance	solving time model	solving time FFD	solving time BFD	
CS1	0.030	0.010	0.25	
CS2	0.090	0.09	0.021	
CS3	11.68	0.032	0.239	
CS4	62.41	0.089	0.608	

The approximation quality of the heuristics strongly depends on the data. In instance CS4 the difference of used bars in the exact model and the heuristics is

about 31. In times of rising prices and the importance of protecting resources and the environment, this is not acceptable and is a fact against the use of heuristics.

Nevertheless the combination of exact model and heuristics offers some useful opportunities. The heuristics offer bounds for an order, and they are executed fast. This could be used for example in the planning phase for processing many orders.

References

[1] Paul C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting stock problem i. *Operations Research*, 9:849–859, 01 1961.

[2] Joseph Czyzyk, Michael P. Mesnier, and Jorge J. Moré. The neos server. *IEEE Journal on Computational Science and Engineering*, 5(3):68 — 75, 1998.

[3] Elizabeth D. Dolan. The neos server 4.0 administrative guide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[4] William Gropp and Jorge J. Moré. Optimization environments and the neos server. In Martin D. Buhman and Arieh Iserles, editors, *Approximation Theory and Optimization*, pages 167 – 182. Cambridge University Press, 1997.

[5] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin, July 2018.