Mathematical Programming Models for the one-dimensional cutting stock problem with usable leftover

Santiago Valdés Ravelo
Instituto de Informática
Universidade Federal de Goiás
`santiago@inf.ufg.br`

Cláudio Nogueira de Meneses
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
`claudio_n_meneses@yahoo.com`

Maristela Oliveira dos Santos
Departamento de Matemática Aplicada e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
`mari@icmc.usp.br`

Abstract

We consider the one-dimensional cutting stock problem in which the non-used material in the cutting patterns may be used in the future, if large enough. This variant is more complicated to solve than the traditional one because it introduces a new objective to be minimized besides minimizes the loss material, being a multiobjective optimization problem and also an NP-hard problem. We analyze the existing mathematical models and propose new models, comparing them by solving instances from the literature and practical instances with the CPLEX solver. The computational experiments show that we proposed a model quite superior to the existing models for this problem.

Keywords: One-dimensional Cutting Stock Problem, Mathematical Modeling, Multiobjective Optimization. Combinatorial Optimization

1 Introduction

Cutting stock problems arise from many applications in the industry, they consist in cutting a set of items from a stock of larger objects, in order to satisfy the item demands, minimizing the loss material, the cost of the objects to cut or maximizing the profit of the produced items, being in general all these problems computationally difficult to solve. In this article we consider the one-dimensional cutting stock problem with usable leftover, in which if the non-used material in the cutting patterns is large enough it may be used in the future. This feature introduces more difficulties to the problem and has received attention in the literature, creating mathematical models and several algorithms to get *good* solutions (Gradisar et al. (1997), Gradisar et al. (1999), Gradisar and Trkman (2005), Ababuara and Morabito (2008), Cherri et al. (2008)).

Cherri et al. (2008) gave the following definition to the problem:

Definition 1. The one-dimensional cutting stock problem with usable leftover (CSPUL) is defined as:

A set of pieces (items) must be produced by cutting large units (objects) of standard sizes (objects bought from suppliers) or non-standard (objects that are leftover of previous cuts). The demand of the items and the availability of the objects are given. Demand must be met by cutting the available objects such that the leftovers are "small" (denoted by scrap) or "sufficiently large" (denoted by retails) to return to stock, but in a reduced number.

In this work we begin with Section 2 analyzing the existing models, and in Section 3 are presented new mathematical models to the problem. Section 4 presents computational tests and finally the conclusions are in Section 5.

2 Existing Mathematical Models

To the best of our knowledge Gradisar et al. (1997) was the first one proposing a model to the cutting stock problem that considers the existence of usable leftover (retails). Lets see that model:

- m: number of objects
- d_j: size of object j $(1 \leq j \leq m)$
- n: number of items
- s_i: length of item i $(1 \leq i \leq n)$
- b_i: demand of item i $(1 \leq i \leq n)$
- UB: minimum size of a leftover material to be considered as retail
- Y, M: constants to determine the maximum number of different items cut out of one object (used $Y = M = 4$)

Variables:

- x_{ij}: number of items type i having being cut from object j $(1 \leq i \leq n$ and $1 \leq j \leq m)$
- δ_j: leftover material of the object j $(1 \leq j \leq m)$
- Δ_i: uncut order of items type i $(1 \leq i \leq n)$
- t_j: loss of the object j $(1 \leq j \leq m)$

- y_{ij}: indicates if an item type i is being cut from object j ($1 \le i \le n$ and $1 \le j \le m$)
- z_j: indicates if object j is used in the cutting plan ($1 \le j \le m$)

Model:

$$\min \quad \sum_{i=1}^{n} \Delta_i \tag{1}$$

$$\min \quad \sum_{j=1}^{m} t_j \tag{2}$$

$$\begin{aligned}
s.t: \quad & \sum_{i=1}^{n} s_i x_{ij} + \delta_j = d_j \quad \forall j \quad \text{(knapsack constraints)} \\
& \sum_{j=1}^{m} x_{ij} + \Delta_i = b_i \quad \forall i \quad \text{(demand constraints)} \\
& \sum_{i=1}^{n} y_{ij} \le Y \le M \quad \forall j \quad \text{(maximum number of different items for an object)} \\
& t_j = \begin{cases} \delta_j, & \text{if } z_j = 1 \text{ and } \delta_j < UB \\ 0, & \text{otherwise} \end{cases} \quad \forall j \\
& y_{ij} = \begin{cases} 0, & \text{if } x_{ij} = 0 \\ 1, & \text{otherwise} \end{cases} \quad \forall i, j \\
& z_j = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} x_{ij} = 0 \\ 1, & \text{otherwise} \end{cases} \quad \forall j \\
& Integers: x_{ij} \ge 0,\ \delta_j \ge 0,\ t_j \ge 0,\ \Delta_i \ge 0,\ y_{ij}, z_j \in \{0,1\} \\
& (1 \le i \le n \text{ and } 1 \le j \le m)
\end{aligned}$$

The *knapsack constraints* guarantee that the sum of lengths of the items cut from an object is not larger than the length of the object, and the *demand constraints* guarantee that the number of item of a type is no greater than the demand of that type of item.

The above model does not solve our problem, but solves a very similar other one, so making some modifications we get a model that actually solves CSPUL. In first place we do not need the variables y_{ij}, because in the formulation of our particular problem there is no restriction to the number of types of items that can be produced by one object. Also we can drop the variables z_j and redefine the variables t_j as:

$$t_j = \begin{cases} \delta_j, & \text{if } \delta_j < d_j \text{ and } \delta_j < UB \\ 0, & \text{otherwise} \end{cases}$$

Adding the variables $r_j = \begin{cases} 1, & \text{if } \delta_j < d_j \text{ and } \delta_j \ge UB \\ 0, & \text{otherwise} \end{cases}$

And dropping the variables Δ_i because we are considering that all demands can be satisfied, we replace the objective function (1) by:

$$\min \quad \sum_{j=1}^{m} r_j \tag{3}$$

Finally, we get a mathematical model for CSPUL:

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{m} r_j \\
\min \quad & \sum_{j=1}^{m} t_j \\
s.t: \quad & \sum_{i=1}^{n} s_i x_{ij} + \delta_j = d_j \quad \forall j \quad \text{(knapsack constraints)} \\
& \sum_{j=1}^{m} x_{ij} = b_i \quad \forall i \quad \text{(demand constraints)} \\
& t_j = \begin{cases} \delta_j, \text{ if } \delta_j < d_j \text{ and } \delta_j < UB \\ 0, \text{ otherwise} \end{cases} \quad \forall j \\
& r_j = \begin{cases} 1, \text{ if } \delta_j < d_j \text{ and } \delta_j \geq UB \\ 0, \text{ otherwise} \end{cases} \quad \forall j \\
& Integers: x_{ij} \geq 0, \ \delta_j \geq 0, \ t_j \geq 0, \ r_j \in \{0,1\} \\
& (1 \leq i \leq n \text{ and } 1 \leq j \leq m)
\end{aligned}
$$

In Gradisar et al. (1999) and Gradisar and Trkman (2005) the same model of Gradisar et al. (1997) is analyzed by cases and are added a new variable and a constraint associated to the maximum length of retails (there must not be more than one retail larger than the largest item):

$$
\sum_{j=1}^{m} u_j \leq 1, \text{ where: } u_j = \begin{cases} 1, \text{ if } \delta_j < d_j \text{ and } \delta_j > \max_i s_i \\ 0, \text{ otherwise} \end{cases}
$$

We do not consider this constraint because it could be interesting having few but larger retails.

In Ababuara and Morabito (2008), the model of Gradisar et al. (1997) is modified and they proposed the following two models:

Model 1:

Adds the variables: w_j $(1 \leq j \leq m)$

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{m} t_j \\
s.t: \quad & \sum_{i=1}^{n} s_i x_{ij} + \delta_j = d_j \quad \forall j \quad \text{(knapsack constraints)} \\
& \sum_{j=1}^{m} x_{ij} = b_i \quad \forall i \quad \text{(demand constraints)} \\
& z_j \leq \sum_{i=1}^{n} x_{ij} \quad \forall j \quad \text{(object constraints)} \\
& \sum_{i=1}^{n} x_{ij} \leq M z_j \quad \forall j \quad \text{(object constraints)}
\end{aligned}
$$

$$
\begin{aligned}
&\delta_j - UB \geq -Mw_j \quad \forall j \quad \text{(lost constraints)}\\
&\delta_j - UB \leq (M+\epsilon)(1-w_j) - \epsilon \quad \forall j \quad \text{(lost constraints)}\\
&t_j - Mw_j \leq 0 \quad \forall j \quad \text{(lost constraints)}\\
&t_j - Mz_j \leq 0 \quad \forall j \quad \text{(lost constraints)}\\
&t_j - \delta_j \leq 0 \quad \forall j \quad \text{(lost constraints)}\\
&\delta_j - t_j + Mw_j + Mz_j \leq 2M \quad \forall j \quad \text{(lost constraints)}\\
&-z_j + u_j \leq 0 \quad \forall\ j \quad \text{(retail constraints)}\\
&w_j + u_j \leq 1 \quad \forall\ j \quad \text{(retail constraints)}\\
&z_j - w_j - u_j \leq 0 \quad \forall\ j \quad \text{(retail constraints)}\\
&\sum_{j=0}^{m} u_j \leq RUB \quad \text{(maximum number of retails)}\\
&\delta_j, t_j, x_{ij} \geq 0,\ x_{ij} \in \mathbb{Z},\ z_j, u_j, w_j \in \{0,1\} \qquad (4)\\
&(1 \leq i \leq n \text{ and } 1 \leq j \leq m)
\end{aligned}
$$

Where are defined: $UB = \min_i s_i$, $M = \max_j d_j - \min_i s_i$, RUB is an upper bound for the number of retails in the solution (the proposal is $RUB = 1$), and ϵ is a small positive value to change the signal of the inequalities (when all values are integers $\epsilon = 1$).

On this model the *object constraints* set z_j to 1 if object j is used in the solution and to 0 otherwise. The *lost constraints* are to guarantee that if the object is used the leftover from that object is not a retail, then the leftover of that object is loss material. The *retail constraints* are to guarantee that if an object is used and it leftover is not loss material, then it must be a retail.

Model 2:

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{m} t_j\\
s.t: \quad & \sum_{i=1}^{n} s_i x_{ij} \leq d_j \quad \forall j \quad \text{(knapsack constraints)}\\
& \sum_{j=1}^{m} x_{ij} = b_i \quad \forall i \quad \text{(demand constraints)}\\
& \sum_{j=1}^{m} u_j \leq RUB \quad \text{(maximum number of retails)}\\
& UBu_j \leq d_j z_j - \sum_{i=1}^{n} s_i x_{ij} \quad \forall j \quad \text{(leftover constraints)}\\
& d_j z_j - \sum_{i=1}^{n} s_i x_{ij} \leq t_j + Mu_j \quad \forall j \quad \text{(leftover constraints)}\\
& t_j, x_{ij} \geq 0,\ x_{ij} \in \mathbb{Z},\ u_j \in \{0,1\}\\
& (1 \leq i \leq n \text{ and } 1 \leq j \leq m)
\end{aligned}
$$

The *leftover constraints* collapse all the *object constraints*, *lost constraints* and *retail constraints* from the previous model.

The second model is a simplification of the first one, but the two models solve the same problem, and their objective is to minimize the leftover material with a limited number of retails. They are approximated models to CSPUL, but their solutions are not necessarily the best solutions we can get because the original problem is multiobjective and those models reduce it to a single objective problem fixing the number of retails with a constraint.

For the CSPUL the Pareto optimal set is finite so solving the models of Ababuara and Morabito (2008) with $RUB = 0...K$, with K large enough, it is possible to get all non-dominated solutions of the multiobjective problem.

3 New Mathematical Models

The models of Ababuara and Morabito (2008) allow symmetry arising identical solutions to the problem by swapping the objects j_1 and j_2 if they are of same type. So it could be interesting to create models breaking the symmetry.

Ordering the objects by their lengths and adding the constraints: $z_{j+1} \leq z_j \; \forall j$ if objects j and $j+1$ are the same type.

We reduce the number of symmetric solutions allowed by the **Model 2**, and we get the **Model 3**:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{m} t_j \\
\min \quad & \sum_{j=1}^{m} u_j \\
s.t: \quad & \sum_{i=1}^{n} s_i x_{ij} \leq d_j \quad \forall j \quad \text{(knapsack constraints)} \\
& \sum_{j=1}^{m} x_{ij} = b_i \quad \forall i \quad \text{(demand constraints)} \\
& UBu_j \leq d_j z_j - \sum_{i=1}^{n} s_i x_{ij} \quad \forall j \quad \text{(leftover constraints)} \\
& d_j z_j - \sum_{i=1}^{n} s_i x_{ij} \leq t_j + M u_j \quad \forall j \quad \text{(leftover constraints)} \\
& z_{j+1} \leq z_j \quad \forall j \text{ if objects } j \text{ and } j+1 \text{ are the same type} \\
& t_j, x_{ij} \geq 0, \;\; x_{ij} \in \mathbb{Z}, \;\; u_j \in \{0,1\} \\
& (1 \leq i \leq n \text{ and } 1 \leq j \leq m)
\end{aligned}$$

But **Model 3** still allows symmetric solutions and that is the reason we decided to introduce a totally different model:

- m: number of types of objects
- C_j: number of objects of type j $(1 \leq j \leq m)$
- n: number of items
- b_i: demand of item i $(1 \leq i \leq n)$
- K_j: number of possible cut patterns for objects of type j $(1 \leq j \leq m)$

- a_{ijk}: number of items i in cut pattern k of objects of type j ($1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq K_j$)
- u_{jk}: indicates if cut pattern k of objects of type j generates a retail ($1 \leq j \leq m$, $1 \leq k \leq K_j$)
- t_{jk}: loss of cut pattern k of objects of type j ($1 \leq j \leq m$, $1 \leq k \leq K_j$)

Variables:

- x_{jk}: number of times that cut pattern k of objects of type j has been included in the solution

Model 4:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{m}\sum_{k=1}^{K_j} t_{jk}x_{jk} \\
\min \quad & \sum_{j=1}^{m}\sum_{k=1}^{K_j} u_{jk}x_{jk} \\
s.t: \quad & \sum_{j=1}^{m}\sum_{k=1}^{K_j} a_{ijk}x_{jk} = b_i \quad \forall i \quad \text{(demand constraints)} \\
& \sum_{k=1}^{K_j} x_{jk} \leq C_j \quad \forall j \quad \text{(stock constraints)} \\
& x_{jk} \geq 0,\ x_{jk} \in \mathbb{Z} \\
& (1 \leq i \leq n,\ 1 \leq j \leq m \text{ and } 1 \leq k \leq K_j)
\end{aligned}$$

The *stock constraints* are to guarantee that there are not used more objects of a type than the amount of them in the stock.

4 Computational Tests

In this section we present tests with instances used by Cherri et al. (2008), some of them taken from Gradisar and Trkman (2005) (the numerical instances) and others from Ababuara and Morabito (2008) (the practical instances).

To compare the models we fix a maximum number of retails and solve a single objective model using the CPLEX solver, on a PC Intel Core 2 Duo, 2.8GHz and 3.25GB of RAM, under Microsoft Windows XP Professional operating system.

The following subsections discuss the instances and the obtained results. The results are given in tables in which are used Var.Numb. to denote the number of variables, Const.Numb. to denote the number of constraints, RUB to denote the number of retails, Init.Sol. to denote the initial integer solution, Init.GAP to denote the gap calculated by the CPLEX for the initial solution, Last.Sol. to denote the last integer solution, GAP to denote the gap calculated by the CPLEX for the last solution, Nod.Numb. to denote the number of nodes, Iter. to denote the number of iterations, Stop.Crit. to denote the stop criterion being: OPT by optimal solution and OME by out of memory exception, and Time to denote the execution time in seconds.

4.1 Numerical Instances

For these instances there is just one object in stock for each type of object, so models 2 and 3 result the same model.

Instance 1: with 20 types of objects with lengths between 2200 and 6000 *cm*; and availability of one unit of each object type. Table 1 shows size and demand of each item. Table 2 shows the CPLEX solutions for this instance with the different models.

Item	Length (cm)	Demand
1	235	4
2	200	51
3	347	42
4	471	16
5	274	37

Table 1: Item lengths and demands of instance 1

	Model 1		Model 2 and Model 3		Model 4	
Var.Numb.	200		160		228928	
Const.Numb.	246		66		26	
RUB	0	1	0	1	0	1
Init.Sol.	812	666	1612	706	212	128
Init.GAP	100	100	100	100	100	100
Last.Sol.	412	56	212	2	12*	0*
GAP	100	100	100	100	100	0
Nod.Numb.	12311549	13982753	299588	2816951	159119	14800
Iter.	246490262	51903397	63764569	72973808	326807	68008
Stop.Crit.	OME	OME	OME	OME	OME	OPT
Time	81131.97	99449.89	54980.63	224907.99	851.56	5239.51

Table 2: CPLEX solutions for each model with instance 1. The * indicates that the solution is optimal.

Instance 2: with 20 types of objects with lengths between 2100 and 5000 *cm*; and availability of one unit of each object type. Table 3 shows size and demand of each item. Tables 4, 5 and 6 show the CPLEX solutions for this instance with the different models.

Item	Length (cm)	Demand
1	549	39
2	433	27
3	207	43
4	308	30
5	583	2

Table 3: Item lengths and demands of instance 2

From the tables it is evident that even when the **Model 4** has much more variables than the rest of the models, it seems to be much better than the other proposed models, reaching an optimal solution in each case and on almost all the tests stopping by optimality in a very short time while for the other models the CPLEX solver takes much more time to stop, and when it does is by out of memory exception without reaching optimal values.

	Model 1				
Var.Numb.	200				
Const.Numb.	246				
RUB	0	1	2	3	4
Init.Sol.	1523	1144	1044	747	1319
Init.GAP	100	100	100	100	100
Last.Sol.	763	412	588	263	335
GAP	100	100	100	100	100
Nod.Numb.	13319148	16888927	13788840	14090112	13225677
Iter.	133503248	274466473	127304689	166485902	129522082
Stop.Crit.	OME	OME	OME	OME	OME
Time	13213.92	20635.11	15711.78	34251.25	56088.31

Table 4: CPLEX solutions for model 1 with instance 2.

	Model 2 and Model 3				
Var.Numb.	160				
Const.Numb.	66				
RUB	0	1	2	3	4
Init.Sol.	6359	2480	2704	2988	1256
Init.GAP	100	100	100	100	100
Last.Sol.	639	322	99	11	46
GAP	100	100	100	100	100
Nod.Numb.	11246014	31677069	21057922	32281347	28203188
Iter.	113908074	172418711	121279869	221740224	186045954
Stop.Crit.	OME	OME	OME	OME	OME
Time	12484.94	233656.75	79663.91	32032.25	30844.71

Table 5: CPLEX solutions for models 2 and 3 with instance 2.

	Model 4				
Var.Numb.	44389				
Const.Numb.	26				
RUB	0	1	2	3	4
Init.Sol.	183	92	2*	1*	1
Init.GAP	98	97.2	61.96	100	100
Last.Sol.	31*	3*	2*	1*	0*
GAP	79.15	0	0	0	0
Nod.Numb.	561368	9	8	57	260
Iter.	3012034	229	226	778	5025
Stop.Crit.	OME	OPT	OPT	OPT	OPT
Time	1793.03	210.8	183.02	60.67	111.02

Table 6: CPLEX solutions for model 4 with instance 2. The * indicates that the solution is optimal.

4.2 Practical Instances

Instance 3: with *one* object type of length 3000 *cm*, and availability of 10 in stock. Table 7 shows size and demand of each item. Table 8 shows the CPLEX solutions for this instance with the different models.

Item	Length (cm)	Demand
1	250	2
2	273	2
3	285	4
4	525	4
5	1380	4

Table 7: Item lengths and demands of instance 3

	Model 1		Model 2		Model 3		Model 4	
Var.Numb.	100		80		80		257	
Const.Numb.	126		36		45		7	
RUB	1	2	1	2	1	2	1	2
Init.Sol.	574	244	574	264	529	70	240*	0*
Init.GAP	100	100	100	100	100	100	0	0
Last.Sol.	240*	0*	240*	0*	240*	0*	240*	0*
GAP	0	0	0	0	0	0	0	0
Nod.Numb.	24639	30	64134	135	331	69	1	0
Iter.	116008	179	239880	509	1385	354	26	10
Stop.Crit.	OPT	OPT	OPT	OPT	OPT	OPT	OPT	OPT
Time	143.5	0.92	35.49	1.05	1.13	1	0.75	0.31

Table 8: CPLEX solutions for each model with instance 3. The * indicates that the solution is optimal.

Instance 4: with *one* object type of length 6000 *cm*, and availability of 10 in stock. Table 9 shows size and demand of each item. Tables 10 and 11 show the CPLEX solutions for this instance with the different models.

Item	Length (cm)	Demand
1	370	5
2	905	5
3	910	5
4	930	5

Table 9: Item lengths and demands of instance 4

As in the previous subsection, the CPLEX solver reaches with our **Model 4** the optimal values in less time, less iterations and using less nodes than the other models. In fact for these instances the CPLEX solver with **Model 4** reaches optimal solutions as initial feasible solutions, proving their optimality in few steps.

	Model 1			Model 2		
Var.Numb.	90			70		
Const.Numb.	125			35		
RUB	1	2	3	1	2	3
Init.Sol.	435	110	0*	3875	2034	0*
Init.GAP	100	100	0	100	100	0
Last.Sol.	250*	70*	0*	250*	70*	0*
GAP	0	0	0	0	0	0
Nod.Numb.	316596	2200604	0	270638	805423	0
Iter.	923668	7593572	14	576347	2231684	12
Stop.Crit.	OPT	OPT	OPT	OPT	OPT	OPT
Time	203.74	1872.75	0.75	149.39	82.53	0.74

Table 10: CPLEX solutions for models 1 and 2 with instance 4. The * indicates that the solution is optimal.

	Model 3			Model 4		
Var.Numb.	70			343		
Const.Numb.	44			6		
RUB	1	2	3	1	2	3
Init.Sol.	6060	190	5615	250*	70*	0*
Init.GAP	100	100	100	18.46	16.67	0
Last.Sol.	250*	70*	0*	250*	70*	0*
GAP	0	0	0	0	0	0
Nod.Numb.	1888	5072	18	10	0	0
Iter.	6872	16508	151	75	26	27
Stop.Crit.	OPT	OPT	OPT	OPT	OPT	OPT
Time	1.88	3.91	1.13	0.81	0.83	0.70

Table 11: CPLEX solutions for models 3 and 4 with instance 4. The * indicates that the solution is the optimal value.

5 Conclusions

We have shown new mathematical models for the One-dimensional Cutting Stock Problem with Usable Leftover, which belongs to the Cutting Problems class that are generally NP-hard problems. We also compare our models with the existing models for these problem using instances from the literature being some of them practical instances. For all experiments we have performed, our **Model 4** was quite superior to the other ones.

References

Ababuara, A. and Morabito, R. (2008). Cutting optimization of structural tubes to build agricultural light aircrafts. *Annals of Operation Research*, 169:149–165.

Cherri, A., Arenales, M., and Yanasse, H. (2008). The one-dimensional cutting problem with usable leftover - A heuirstic approach. *European Journal of Operational Research*, 196:897–908.

Gradisar, M., Jesenco, J., and Resinovic, G. (1997). Optimization of roll cutting in clothing industry. *Computers and Operational Research*, 10:945–953.

Gradisar, M., Kljajic, M., Resinovic, G., and Jesenco, J. (1999). A sequential heuristic procedure for one-dimensional cutting. *European Journal of Operational Research*, 114:557–568.

Gradisar, M. and Trkman, P. (2005). A combined approach to the solution to the general one-dimensional cutting stock problem. *Computers and Operational Research*, 32:1793–1807.