Journal of Computational Interdisciplinary Sciences (2009) 1(2): 159-164

ISSN 1983-8409
http://epacis.org

A pattern reduction procedure in a one-dimensional cutting stock problem by grouping items according to their demands

Gonçalo Renildo Lima Cerqueira[1] and Horacio Hideki Yanasse[2]

Manuscript received on November 25, 2008 / accepted on February 10, 2009

ABSTRACT

Cutting and packing problems appear in many industrial settings where larger pieces are cut into smaller ones in order to produce items or products that are required by other industries or customers, e.g., paper, steel and fiber industries. A solution for the cutting problem consists in determining a group of patterns and their frequencies, that is, the number of times that a pattern is cut to supply the demand. Cost of the production may be dependent on the pattern changes, because they might require time and people to prepare the equipment. This work introduces a heuristic method/approach that produces a solution to the one-dimensional cutting stock problem with a reduced number of different patterns in the solution. The heuristic method proposed begin separating the items in two disjointed groups, according to their demands. Patterns are generated with items of these groups and those with limited waste are accepted. A residual problem is solved with items whose demands are not satisfied and, with the solution obtained, we apply a pattern reducing procedure of the literature.

Keywords: cutting stock problem, computational heuristic method, pattern reduction procedure, linear programming.

Correspondence to: Gonçalo Renildo Lima Cerqueira

[1]Department of Exact Sciences, State University of the Southwest of Bahia (UESB) – Box 95, 45083-900 Vitória da Conquista, BA, Brazil. E-mail: goncalo@uesb.br

[2]National Space Research Institute (INPE/LAC) – Box 515, 12227-010 São José dos Campos, SP, Brazil. E-mail: horacio@lac.inpe.br

1 INTRODUCTION

Suppose that there exist n items with length l_i and demands d_i that should be cut from objects in stock with length L and cost equal to c_j for $j \in J$, where J is the set of indices of all the feasible patterns. We assume without loss of generality that $c_j = 1, \forall j \in J$. The problem consists of minimizing the amount of times that each pattern should be used so that all of the items have their demands completely satisfied. The cutting stock problem can be formulated as follows:

$$\text{MINIMIZE} : f(x) = \sum_{j \in J} x_j \tag{1}$$

$$\text{SUBJECT TO} : \sum_{j \in J} a_{ij} x_j \geq d_i \quad i = 1, \ldots n \tag{2}$$

$$\sum_{i=1}^{n} a_{ij} l_i \leq L, \quad \forall j \in J \tag{3}$$

$$a_{ij} \in N, x_j \in N, i = 1, \ldots, n, \quad \forall j \in J \tag{4}$$

The imposition of integrality on the variables x_j makes the problem difficult to solve compared to the case where such constraints are not present. In addition, the total number of different patterns increases greatly when n (the number of different item types) increases. The cardinality of set J can easily reach the order of millions already for a dozen of item types depending on the item sizes. Gilmore and Gomory [6], [7] propose to solve problem (1)-(4) by the Simplex Method with Column Generation after relaxing the integrality constraints on variables x_j. The columns that must satisfy (3), (4) are generated when required; there is no need to determine all the possible patterns. An integer solution for the problem is obtained afterwards by applying rounding techniques found in the literature (see Poldi [13]) to the solution obtained with the linear relaxed problem.

In this work we propose a heuristic method to solve the one-dimensional cutting stock problem considering also as objective, the minimization of the number of patterns in the solution. In real cutting settings, a reduced number of different patterns is often desired, mainly when this also implies reductions in costs. There is not much work in the literature that considers the pattern reduction problem.

Haessler [9] proposed the use of a SHP (*Sequential Heuristic Procedure*), a repeated pattern exhaustion technique, in which a pattern is selected if it satisfies aspiration levels for the waste and frequency. In each iteration, a pattern is selected if it satisfies the aspirations levels. Its frequency is determined by cutting it as much as possible without overproducing any of the items. The process is repeated until there are no more demands to be satisfied.

Farley and Richardson [4] consider a pattern minimization problem as a fixed charge problem, with an objective composed by the sum of the costs of the patterns and of the setups corresponding to the different patterns.

Foerster and Wascher [5] proposed the KOMBI, an extension of the pattern minimization method proposed by Diegel et al. [1] based on the combination of 2, 3, 4 or more patterns of a solution for the cutting problem. The procedure maintains constant the total number of objects processed by ensuring that the sum of the frequencies of the resulting patterns has to be the same as the sum of the frequencies of the initial combined patterns.

Umetami et al. [15] proposed the *Iterated Local Search Algorithm* (ILS), which that fixes the total number of different patterns and searches for a solution that minimizes the quadratic deviation of the items produced in relation to the demand. The algorithm generates a solution in the neighborhood of an initial solution obtained by perturbing one pattern at a time of the current set of patterns. The frequency of the new pattern is obtained by solving an auxiliary linear programming problem.

Yanasse and Limeira [17] proposed a hybrid procedure composed of three phases. In the first phase, patterns are generated and the "good" ones are selected and used to reduce the problem; in the second phase, the residual problem is solved, if it exists; and in the third, a pattern reducing technique of the literature is applied [5]. A "good" pattern is one that completes totally the demand of at least two items, and presents an acceptable waste.

Vanderbeck [16] proposes an exact algorithm by formulating the problem as an integer quadratic linear programming problem solved using a branch-and-bound column generation technique. Computational tests performed by the author showed a deficiency of his algorithm in solving large sized instances. On the other hand, his algorithm was able to solve exactly several small sized instances.

Cui et al. [1] present a sequential heuristic method for the one-dimensional cutting stock problem minimizing waste and a reduced number of patterns. It uses some parameters that modify the waste and number of patterns in the solution. Different values for these parameters are tested and the best cutting plan is kept. According to the authors the performance of their heuristic method is better than the others in the literature with the advantage of being simpler to implement.

2 PATTERN REDUCTION PROCEDURE

In this section, we describe the Pattern Reduction Procedure Grouping by Demand (PRPGD). In this procedure, we try to generate patterns that contain items whose demands are close, therefore, with these patterns, we can produce a "reasonable" amount of these items, with the possibility of fulfilling their demands.

Let $(D_{\max})$ and (d_{mn}) be the largest and the smallest demand, respectively, in the original cutting stock problem. If $(d_{mn}) < 3$ then make $(d_{mn}) = 3$. Consider the following two disjoint intervals, $I_1 = [d_{mn}, D_{\max}/2]$ and $I_2 = (D_{\max}/2, D_{\max}]$.

Let $r_i > 0$ be the residual demand of item i and $(d_{\min})$ the minimum residual demand. If $(d_{\min}) < 3$ then make $(d_{\min}) = 3$. Initially, $r_i = d_i$ for $i = 1, \ldots, n$.

Initially, only the items with large demands are considered, that is, items i, $i = 1, \ldots, n$, such that $r_i \in I_2$. Then, items with demands in the interval I_1 are considered. Afterwards, all the items whose demands were not yet satisfied are considered.

For the set of items satisfying I_1 and I_2, patterns are generated by solving the following bounded knapsack problem

$$\text{MAXIMIZE}: \quad \sum_{i=1}^{n} C_i y_i \tag{5}$$

$$\text{SUBJECT TO}: \sum_{i=1}^{n} l_i y_i \leq L \tag{6}$$

$$0 \leq y_i \leq H_i, y_i \in N, i = 1, \ldots, n \tag{7}$$

where $C_i = l_i^2$ if $l_i > L/2$ and $C_i = l_i$, otherwise; y_i is the number of item type i in the pattern, and $(H_i = \lfloor r_i/d_{\min} \rfloor)$. For items not in the set considered, $H_i = 0$.

The frequency (f_P) of a generated pattern $P_j = [y_1, \ldots, y_n]$ that satisfies the aspiration levels for the efficiency is given by $f_P = \min\lfloor r_i/y_i \rfloor, r_i \geq y_i > 0; i = 1, \ldots, n$. Every time a pattern is accepted, the residual demand of the items is updated and the process of pattern generation is repeated.

When a pattern is not accepted, the process of generation of patterns is repeated with the next set of items.

Patterns are generated until one is not accepted. If there are still unsatisfied demands, a residual problem, based on the ideas in Cui et al. [1], is solved where all the patterns generated are accepted.

The aspiration level for the efficiency is determined taking as reference the overall waste (α) of the patterns of the solution obtained using Gilmore and Gomory's procedure. The aspiration level for the efficiency of a pattern is given by $(naep = 1 - \mu)$ where $\mu = 0.8\alpha$.

A PRPGD heuristic method is summarized in the Appendix.

3 IMPLEMENTATION AND COMPUTATIONAL RESULTS

The heuristic method was implemented in C++, and the computational tests were performed in an Intel Core 2 Duo computer, 1.50GHz with 1GB RAM.

The computational experiment consisted in applying the heuristic method to eighteen classes of instances, each one of them with 100 test problems obtained using CUTGEN1, a random generator of one-dimensional cutting stock problems developed by Gau and Wascher [7], with the same seed (1994) adopted by Foerster and Wascher [4] and Umetami et al. [15]. In these test problems, item l_i, $i = 1, \ldots, n$, is generated randomly in the interval $[v_1 L, v_2 L]$ for different values of v_1 and v_2; the standard length L of the object is 1000 and the average demand db is equal to 10 or 100. The classes combine different values of n, v_1, v_2 and db, as shown in Table 1.

The results of the computational tests are presented in Table 2. In columns YLP and FWP the average number of objects and patterns obtained using the procedures of Yanasse and Limeira [17] and Foerster and Wascher [5], respectively, are presented. In columns PRPGD, PRPGD 1 and PRPGD 2 the average number of objects and patterns obtained with the heuristic method proposed and two other variants are presented. In the first variant, in the first phase, patterns are generated only with demands in set I_1; in the second variant, in the first phase, patterns are generated only with demands in set I_2.

The computational results in Table 2 suggest that PRPGD, PRPGD1 and PRPGD2 are competitive comparative to the procedures proposed by Foerster and Wascher [5] and Yanasse and Limeira [17], since the solutions they generate are not dominated. In terms of the average number of objects, Foerster and Wascher [5] procedure presented the best results. However, considering classes 9, 11 and 17, the PRPGD heuristic method has obtained better solutions than that obtained by Yanasse and Limeira [17].

The average computational times of the heuristic method (t_{PGPGD}) are shown in Table 3. The execution times, t_{YL} and t_{FW}, of the Yanasse and Limeira [17] and Foerster and Wascher [5] procedures, respectively, were reproduced from their works. The direct comparison of these times is not possible since they come from different machines (the computational tests of Yanasse and Limeira [17] were performed in a microcomputer Intel Celeron, 266 MHz, 128 MB RAM and a workstation Sun Ultra 30,

Table 1 – Characteristics of the classes.

class	n	v_1	v_2	db	class	n	v_1	v_2	db
1	10	0.01	0.2	10	10	20	0.01	0.8	100
2	10	0.01	0.2	100	11	40	0.01	0.8	10
3	20	0.01	0.2	10	12	40	0.01	0.8	100
4	20	0.01	0.2	100	13	10	0.2	0.8	10
5	40	0.01	0.2	10	14	10	0.2	0.8	100
6	40	0.01	0.2	100	15	20	0.2	0.8	10
7	10	0.01	0.8	10	16	20	0.2	0.8	100
8	10	0.01	0.8	100	17	40	0.2	0.8	10
9	20	0.01	0.8	10	18	40	0.2	0.8	100

Table 2 – Average values of the solutions.

class	YLP		FWP		PRPGD		PRPGD 1		PRPGD 2	
	objects	patterns	objects	patterns	objects	patterns	objects	patterns	objects	patterns
1	**11.56**	**3.31**	**11.49**	**3.40**	**11.58**	**3.36**	**11.58**	**3.36**	**11.58**	**3.26**
2	**110.4**	**6.95**	**110.25**	**7.81**	**111.85**	**5.30**	**111.88**	**5.28**	**111.82**	**5.41**
3	**22.17**	**4.96**	**22.13**	**5.89**	**22.20**	**5.22**	**22.27**	**5.10**	**22.27**	**5.36**
4	**215.98**	**10.32**	**215.93**	**14.26**	**217.73**	**8.04**	**218.01**	**7.95**	**218.17**	**7.90**
5	**42.99**	**7.63**	**42.96**	**10.75**	**43.06**	**7.69**	**43.15**	**8.29**	**43.19**	**8.16**
6	**424.89**	**13.31**	**424.71**	**25.44**	**427.40**	**12.79**	**428.31**	**12.64**	**428.41**	**12.34**
7	**51.69**	**7.66**	**50.21**	**7.90**	**51.68**	**7.63**	**52.24**	**7.56**	**51.80**	**7.68**
8	**502.23**	**9.62**	**499.52**	**9.96**	**516.44**	**9.82**	**519.84**	**9.75**	**520.00**	**9.77**
9	99.49	13.64	**93.67**	**15.03**	**98.04**	**13.41**	**99.41**	**13.36**	**99.08**	**13.52**
10	**948.41**	**18.21**	**932.32**	**19.28**	**990.04**	**17.44**	**1002.31**	**17.14**	**1003.03**	**17.16**
11	195.67	24.60	**176.97**	**28.74**	**185.77**	**24.25**	**191.45**	**23.35**	**190.91**	**23.55**
12	**1847.42**	**33.23**	**1766.20**	**37.31**	**1841.55**	**31.92**	**1942.35**	**30.06**	**1944.66**	**29.99**
13	**64.20**	**8.93**	**63.27**	**8.97**	**64.33**	**8.86**	**65.37**	**8.72**	**65.28**	**8.80**
14	633.26	10.51	**632.12**	**10.32**	638.89	10.46	656.44	10.61	652.23	10.58
15	**123.90**	**16.28**	**119.43**	**16.88**	**123.36**	**16.21**	**126.18**	**15.76**	**125.84**	**15.73**
16	**1197.66**	**19.89**	**1191.80**	**19.91**	**1226.89**	**19.52**	**1271.14**	**19.40**	**1270.76**	**19.51**
17	244.02	29.76	**224.68**	**31.46**	**235.51**	**29.36**	**241.46**	**28.11**	**241.34**	**28.09**
18	**2268.3**	**37.90**	**2242.4**	**38.28**	**2374.4**	**35.77**	**2459.86**	**35.65**	**2458.71**	**37.41**

296 MHz 384 MB RAM; all procedures of Foerster and Wascher [5] were implemented on an IBM-compatible 486/66 personal computer using MODULA-2, under MS-DOS 6.0).

4 CONCLUSIONS

We presented a new heuristic method to solve the cutting stock problem with a reduced number of different patterns. The heuristic method it proposed tries to generate patterns with limited waste, with different items with similar demands. The solutions obtained with the new heuristic method were, in general, not dominated by the solutions obtained by using two other usual methods. Changes can be introduced in the proposed heuristic method aiming to improve its performance. One possible change is to define the intervals for the demands dynamically, in each iteration, instead of in the beginning of the process and varying their lengths.

ACKNOWLEDGMENTS

This work was partially financed by FAPESP, CAPES and CNPq to whom the authors would like to express their gratitude.

Table 3 – Execution times.

class	t_{YL}	t_{FW}	t_{PGPGD}	class	t_{YL}	t_{FW}	t_{PGPGD}
1	0.23	0.35	0.10	10	3.36	3.40	3.00
2	0.48	1.26	0.12	11	7.17	36.98	4.82
3	0.12	2.10	73.58	12	44.62	77.41	39.58
4	2.75	16.41	8.57	13	0.13	0.13	0.12
5	3.43	40.03	3.01	14	0.25	0.18	0.18
6	7.81	383.30	6.54	15	0.97	1.92	0.37
7	0.11	0.11	0.12	16	2.46	2.71	3.36
8	0.60	0.24	0.20	17	15.46	51.31	7.97
9	0.49	1.47	0.40	18	50.61	71.34	60.87

REFERENCES

[1] CUI Y, ZHAO X, YANG Y & YU P. 2008. A heuristic for the one dimensional cutting stock problem with pattern reduction. Proceedings of the Institution of Mechanical Engineers, v. 222, part B, Journal of Engineering Manufacture, doi:http://dx.doi.org/10.1243/09544054JEM966.

[2] DIEGEL A, CHETTY M, VAN SCHALKWYCK S & NAIDOO S. 1993. Setup combining in the trimloss problem – 3-to-2 & 2-to-1. Durban: Business Administration, University of Natal. Working Paper. First Draft.

[3] DYCKHOFF H. 1990. A typology of cutting and packing problems. European Journal of Operational Research, 44: 145–159.

[4] FARLEY AA & RICHARDSON KV. 1984. Fixed charge problems with identical fixed charges. European Journal of Operational Research, 18: 245–249.

[5] FOERSTER H & WÄSCHER G. 2000. Pattern reduction in one-dimensional cutting stock problem. International Journal of Production Research, 38: 1657–1676.

[6] GILMORE P & GOMORY R. 1961. A linear programming approach to the cutting-stock problem. Operations Research, 9: 849–859.

[7] GILMORE P & GOMORY R. 1963. A linear programming approach to the cutting-stock problem II. Operations Research, 11(6): 863–888.

[8] GAU T & WÄSCHER G. 1995. CUTGEN1: a problem generator for the standard one-dimensional cutting stock problem. European Journal of Operational Research, 84: 572–579.

[9] HAESSLER RW. 1975. Controlling cutting pattern changes in one-dimensional trim problems. Operations Research, 23: 483–493.

[10] HINXMAN AI. 1980. The trim loss and assortment problem: a survey. European Journal of Operational Research, 5: 8–18.

[11] JOHNSON DS. 1973. Near-optimal bin packing algorithms Cambridge, MA: Massachusetts Intitute of Technology. Tecnical Report MAC TR-109, Project MAC.

[12] McDIARMID C. 1999. Pattern minimisation in cutting stock problems. Discrete Applied Mathematics, 98: 121–130.

[13] POLDI KC. 2003. Algumas extensões do problema de corte de estoque. MS Dissertation, ICMC – USP, São Carlos, SP, Brazil.

[14] STADTLER H. 1990. A one-dimensional cutting stock problem in the aluminium industry and its solution. European Journal of Operational Research, 44: 209–223.

[15] UMETAMI S, YAGIURA M & IBARAKI T. 2003. One-dimensional cutting stock problem to minimize the number of different patterns. European Journal of Operational Research, 146: 388–402.

[16] VANDERBECK F. 2000. Exact algorithm for minimizing the number of setups in the one-dimensional cutting stock problem. Operations Research, 48: 915–926.

[17] YANASSE HH & LIMEIRA MS. 2006. A hybrid heuristic to reduce the number of different patterns in cutting stock problems. Computers & Operations Research, 33: 2744–2756.

[18] WALKER WE. 1976. A heuristic adjacent extreme point algorithm for the fixed charge problem. Management Science, 22: 587–596.

Appendix

Heuristic (PRPGD)

Begin

Input: one-dimensional cutting stock problem
Output: solution for the problem with reduced number of patterns

(First phase)

do $r_i = d_i$ for $i = 1, \ldots, n$.

Step 1: Solve the relaxed cutting problem using CPLEX 10 and determine the overall waste (α) of the patterns in the solution.

Step 2: Compute the aspiration level for the efficiency of a pattern: $naep = 1 - \mu$, where $(\mu = 0.8\alpha)$.

Step 3: Compute the largest $(D_{\max})$and smallest(d_{mn}) demands for the problem. If $(d_{mn}) < 3$ do $(d_{mn}) = 3$. Do $d_{\min} = (d_{mn})$.

Step 4: Define the intervals of demands, $I_1 = [d_{mn}, D_{\max}/2]$ and $I_2 = (D_{\max}/2, D_{\max}]$ with $I = I_1 \cup I_2$.

Step 5: For $i = 1, \ldots, n$, $r_i \in I_2$, make $(H_i = \lfloor r_i/d_{\min} \rfloor)$, otherwise, $H_i = 0$.

Step 6: Solve the bounded knapsack problem and obtain the current pattern : $P_j = [y_1, \ldots, y_n]$.

Step 7: **if** (pattern satisfies aspiration level) **then** determine its frequency $f_P = \min \lfloor r_i / y_i \rfloor, r_i \geq y_i > 0$. Update the demands, determine $(d_{\min} = \min\{r_i\}; i = 1, \ldots, n)$, if $(d_{\min}) < 3$ do $(d_{\min}) = 3$, and return to *Step 5*.

Step 8: For $i = 1, \ldots, n$, $r_i \in I_1$, make $(H_i = \lfloor r_i/d_{\min} \rfloor)$, otherwise, $H_i = 0$.

Step 9: Solve the bounded knapsack problem and obtain the current pattern : $P_j = [y_1, \ldots, y_n]$.

Step 10: **if** (pattern satisfies aspiration level) **then** determine its frequency $f_P = \min \lfloor r_i / y_i \rfloor, r_i \geq y_i > 0$. Update demands, determine $(d_{\min} = \min\{r_i\}; i = 1, \ldots, n)$, if $(d_{\min}) < 3$ do $(d_{\min}) = 3$, and return to *Step 8*.

(Second phase)

Step 11: **if** ($r_i > 0$ for some $i = 1, \ldots, n$.) **then**

call the procedure to solve the residual problem

else go to *step 12*

(Third phase)

Step 12: Establish the solution of the problem

Step 13: Apply pattern reducing techniques to current solution

End

Box 1 – Procedure to solve the residual problem.

Begin

while ($r_i > 0$ for some $i, i = 1, \ldots, n$)

Step 1: Determine $(d_{\max} = \max\{r_i\})$ for $i = 1, \ldots, n$

Step 2: Determine $H = \max\{k\}, k = d_{\max}, \ldots, 2$, such that $\sum_{i=1}^{n} G_i l_i \geq L \cdot naep$, where $(G_i = \lfloor r_i / H \rfloor)$. **If** (no value of H is determined) **then** $H = 1$ and $G_i = r_i$. Generate a pattern with these items types with $H_i = G_i$

Step 3: Accept the cutting pattern and determine its maximum frequency

Step 4: Add the pattern to the solution

Step 5: Update the demands and return to *Step 1*

End