April 13, 2005

RT0606
Operations Research 11 pages

Research Report

A More Flexible Algorithm for Two Dimensional Guillotine Cutting Stock Problem

Toshinari Itoko and Kaiyang Yang

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A More Flexible Algorithm for Two Dimensional Guillotine Cutting Stock Problem

Toshinari Itoko[1] and Kaiyang Yang[2]

[1] Tokyo Research Laboratory, IBM Japan. itoko@jp.ibm.com
[2] Research School of Information Sciences and Engineering, ANU, Australia.
Kaiyang.Yang@anu.edu.au

Abstract. In this paper, we present a novel algorithm for two dimensional guillotine cutting stock problem with application to plate design in steel industry. The algorithm is based on Column Generation idea developed by Gilmore and Gomory. In order to apply Column Generation methodology to our problem and make the algorithm more flexible, we propose a new pricing heuristic which generates feasible guillotine cutting patterns by Wang's algorithm. The main features of the algorithm are (1) it can solve the problems whose stock sizes are uncertain (variables) (2) it is independent of the definition of feasible patterns except for guillotine cutting constraint. The flexibility of the algorithm enables its broader applications to practical problems e.g. plate design in steel industry etc. The computational results show the effectiveness of the algorithm.

1 Introduction

The Cutting Stock Problem (CSP) is to cut or fill large materials (*stocks*) with all *demands* of small materials (*items*) meeting some optimization criteria, e.g. minimizing wastage etc. CSP has a wide variety of different formulations with corresponding applications in practice, see the survey [3] by Dyckhoff for details.

In this paper, we concentrate on two dimensional CSPs allowing guillotine cutting only i.e. edge-to-edge cuts parallel to two edges of stocks and allowing the dimensions of stock sizes to be variables i.e. dimensions of stocks are determined "after" designing the cutting patterns (i.e. layout of items). We assume here that all stocks and items are rectangles which is usually the case in the wood, glass and metal industry.

Usually, CSPs are formulated as an Integer Programming (IP) of the following form:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq d_i \quad (i = 1, \ldots, m), \\ & x_j \in Z_+ \ (j = 1, \ldots, n). \end{aligned}$$

c_j : cost of pattern j
x_j : # of pattern $j$applied
d_i : demand of item i
a_{ij} : # of item i in pattern j
m : # of item types
n : # of feasible patterns

Unfortunately, this formulation has an exponential number of variables x_j and columns $\boldsymbol{a_j} := (a_{1j}, \ldots, a_{mj})^T$ subject to m. In order to overcome this difficulty, Gilmore and Gomory developed the column generation approach which solved Linear Programming (LP) relaxation of the above-mentioned IP for one dimensional CSP [5]. Regarding the problem solved in [5], all the costs are fixed to 1 (i.e. minimizing the number of used stocks) and the feasible patterns are described by inequalities $\forall j, \sum_{i=1}^{m} a_{ij}\, l_i \leq L$ where l_i denotes the length of i-th item and L denotes the length of the stocks. This simple problem setting reduces the pricing problem to a Knapsack problem.

Farley presented improved algorithms based on the Gilmore and Gomory approach to three staged two dimensional CSP in [4]. The Farley's pricing algorithm is based on lexicographical search which is still hard to be applied to n-staged two dimensional CSP.

In this paper, we propose a more "flexible" algorithm which can handle n-staged cases. The algorithm is able to deal with problems whose stock sizes are variables and also it can be easily applied to complicated feasible cutting pattern definition such as certain two different items can not be cut from the same stock etc.

Our algorithm is based on column generation and includes a new pricing heuristic. The heuristic generates new candidate cutting patterns (candidate columns to be added) by employing Wang's algorithm [8].

Our algorithm does not solve pricing problem to optimality since that may not make any sense (= columns) in order to solve the original IP to optimality. We take the solution of LP as a guideline to collect "good" columns i.e. might be selected in the solution process of original IP. The computational results shows the effectiveness of the proposed ideas.

The structure of this paper is organized as the following. Section 2 shows the framework of our algorithm based on column generation and Section 3 presents the detail of the algorithm including the new pricing heuristic. Section 4 presents its application to the steel cutting industry. In Section 5, we show the results of experiments illustrating the effectiveness of the algorithm.

2 Column Generation Based Approach

Column generation is a method to solve huge LP assuming that constraints which define the columns are given. Column generation is often used to solve LP relaxation of IP as which combinatorial optimization problems are formulated. Vehicle routing and crew scheduling problems are well-known applications of column generation (See e.g. [6]).

Column generation is one variant of simplex method. In order to handle the vast number of columns, a small subset of the columns is considered firstly and other columns, whose corresponding variables become new basis (i.e. can improve the objective value), are added in each iteration. The problem considered with subset of columns is called *restricted master problem*, and the original problem with all columns is called *master problem*. According to the duality theorem of

LP, only the columns with negative reduced costs will decrease the objective value.

Column generation based algorithm for IP can be described in the following general framework:

Step 1. Set up restricted master problem with initial columns.
Step 2. Solve LP relaxation of restricted master problem.
Step 3. Find columns with negative reduced cost.
Step 4. Add some/all of them to restricted master problem.
Step 5. If there are no columns to be added, go to Step 6.
Otherwise go back to Step 2.
Step 6. Obtain integer solution by some means.

Reduced cost of column j is defined as $\bar{c_j} := c_j - \sum_{i=1}^{m} \bar{y_i}\, a_{ij}$ where $\bar{y_i}$ is the i-th element of the dual optimal solution for the current LP relaxation of restricted master problem. $\bar{c_j} < 0$ means to cut off the dual optimal solution from the feasible region of the current LP dual problem. This is the outline of the proof that only the columns with negative reduced cost will decrease the objective value.

Searching for columns with negative reduced costs is called *pricing problem* or *subproblem*. Pricing problem of CSP can be described as the following optimization problem.

$$\begin{array}{ll} \text{Maximize} & \displaystyle\sum_{i=1}^{m} \bar{y_i} a_i - c \\ \text{subject to} & \boldsymbol{a} := (a_1, \ldots, a_m)^T \text{is a feasible column (pattern).} \end{array}$$

If the cost of pattern c is fixed to 1 or defined by wastage, this pricing problem becomes a CSP called *constrained CSP*. Constrained CSP is the problem of cutting one stock into items with values not exceeding demands and the total value maximized. By setting the value of item i to be $\bar{y_i}$ or $\bar{y_i} + v_i$ where v_i is the dimension of item i, we can verify the equality relationship between pricing problem and constrained CSP.

Dynamic programming [2] and recursive (tree search) algorithm [7] are two most effective methods to solve constrained CSP to optimality. That is to say, the corresponding pricing problem can be solved to optimality accordingly.

However, in our problem the cost c changes dynamically with the cutting pattern. So, in Step 3 instead of finding the exact column with least reduced cost, we propose a heuristic method to find columns with "near" least reduced costs.

3 Pricing Heuristics

The complete algorithm is described as the following, and the details for pricing heuristic are stated in Step 3a – 3d.

Step 1. Set up restricted master problem with initial columns.
Step 2. Solve LP relaxation of the restricted master problem.
Step 3. Find columns with negative reduced cost.
  a. Sort items (rows) in decreasing order of their dual solution of LP.
  b. [Pattern generation based on Wang's algorithm]:
     Add items to *seeds* in the order one by one and generate maximum number N_MAX_PATTERNS of candidate patterns (columns).
  c. Sort candidate patterns in increasing order of their reduced costs.
  d. [Optional] Sort candidate patterns with negative reduced costs in increasing order of their costs.
Step 4. Add new columns to restricted master problem in the order until there are no columns with negative reduced cost or the number of added columns reaches the maximum number N_ADD_MAX. If there are no further columns to be added, go to Step 5.
  Otherwise, go back to Step 2.
Step 5. Solve IP to get integer solution.

Since reduced cost $\bar{c_j}$ is defined as $c_j - \sum_{i=1}^{m} \bar{y_i}\, a_{ij}$ and a_{ij} (the number of item i in pattern j) is always a nonnegative integer, the larger dual solution $\bar{y_i}$ is, the less reduced cost $\bar{c_j}$ would be. In other words, patterns including items with larger dual solutions are more likely to have less reduced cost. So we choose the seeds (items) for generating patterns in decreasing order of their dual solutions in Step 3a.

In Step 3c, we collect columns with the smallest (negative) reduced costs to improve the LP solution as possible. In addition, in Step 3d, we suggest selecting columns with the least costs so that we may obtain better solutions of original IP. (The effectiveness is shown later in Sect. 5.4.)

By using seed items, we generate lots of feasible patterns in Step 3b. The generation process is based on Wang's algorithm, which produces guillotine feasible cutting patterns by successive horizontal or vertical builds of two smaller patterns. Since we build up two patterns in guillotine feasible way, all newly generated patterns are guillotine feasible. We introduce representation such as H(a, b) and V(a, b) which denote the pattern produced by horizontal and vertical build of item a and b relatively, e.g. Fig. 1 – 3.

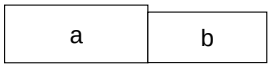

Fig. 1. H(a, b)

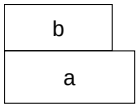

Fig. 2. V(a, b)

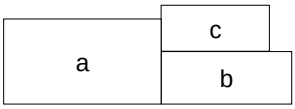

Fig. 3. H(a, V(b, c))

To reduce the number of generated patterns, we reject undesirable patterns based on cutoff ratio β, which is the maximum ratio of inner waste area to the

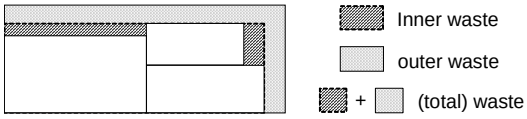

Fig. 4. Inner and outer waste

product area (product area = stock area − outer waste area), see Fig. 4. Also we reject duplicated patterns. For example, H(H(a, a), H(b, b)) and H(H(H(a, a), b), b) are the same patterns generated in the second and third iteration respectively, so we need to reject this pattern after at least the third iteration.

Incorporating these devices, our pattern generation algorithm is described in Fig. 5.

L: List of generated patterns.
s_i: Item with i-th largest dual solution.
n_{last}: Size of L at the end of last iteration.
$L := \emptyset,\ n_{last} := 0,\ N := 0.$
For i: 1 to m,
  $L \leftarrow$ single plate pattern of item s_i,
  While $N <$ size of L,
    $N :=$ size of L,
    For j: 1 to N,
      For k: $n_{last} + 1$ to N,
        For each direction D (i.e. D := H and V),
          check feasibility of D($L[j]$, $L[k]$),
          check desirability of D($L[j]$, $L[k]$),
          If D($L[j]$, $L[k]$) is feasible and desirable,
            $L \leftarrow$ D($L[j]$, $L[k]$).
    $n_{last} := N$.
Return.
Note: Whenever $n =$ N_MAX_PATTERNS, return at the moment.

Fig. 5. Pattern generation based on Wang's algorithm

4 Application to Plate Design in Steel Industry

In this part, we shows the application of the algorithm to plate design in steel industry.

Plate design is a CSP which has the following three difficulties. First of all, stocks are not given in advance; whereas stocks ("products" may be more suitable) are designed in the algorithm. Secondly, the definition of feasible cutting

patterns is very complicated since products must satisfy many quality requirements and only the existing machines are available and so on. Third, the costs of products are not simple since not only the calculation of waste is difficult but also soft constraints have to be considered. We have already shown that our algorithm could handle these difficulties.

In the rest of this paper, we show the effectiveness of the algorithm by computational experiments. Before presenting the results of the experiments, we give the problem settings.

Since the real problem setting is too complicated to be explicitly presented in this paper, we have the simplified setting as the following. Maximum width and length of products are set to 3,500 and 50,000 respectively. These constraints must be satisfied strictly. Also minimum width and length of products are 2,500 and 25,000 respectively. These values are only used to define waste area, see Fig. 6. The costs are defined by waste area. Only the guillotine cuts are allowed and

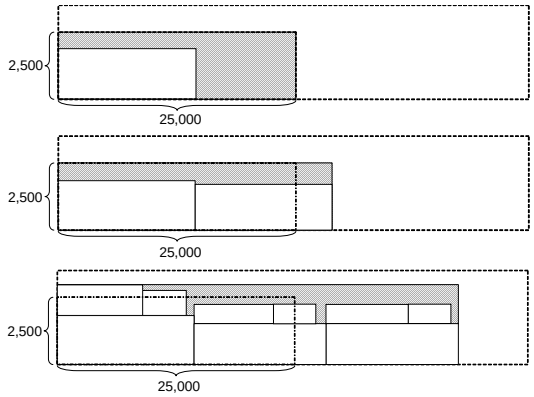

Fig. 6. Dimension of product and waste area

only one vertical cut is permitted to imitate practical setting. By one vertical cut constraint, at most 4-staged cutting patterns are generated.

5 Computational Experiment

Two kinds of experiments were carried out for different purposes. One is *property test* to show the characteristics of our algorithm including the following four experiments:

(1) Relationship between LP and IP optimal values,
(2) Adding multiple columns,
(3) Starting with more initial columns,
(4) Sorting candidate columns by their costs.

The other one is *comparison test* to examine the performance of our algorithm comparing with Wang's original algorithm and its simple extension.

We have three input data S, M, L which have small, medium and large number of items respectively (Table 1). We used data M for the first three property tests and all data for the other tests.

Table 1. Data size

	data S	data M	data L
types	16	46	212
items	51	335	587

We implemented our algorithm in C++ language with open source library developed in COIN-OR project [1]. We use COIN/Clp and COIN/Cbc modules to solve LP and IP respectively. We run the program on the personal computer (CPU 1.6GHz, RAM 1.0GB).

Throughout property tests, we fix the maximum number of patterns allowed to be $\mathsf{N\_MAX\_PATTERNS} = 5000$ and cutoff ratio $\beta = 0.1$. Throughout all tests, IP solving process is always limited to maximum 3 minutes.

5.1 Relationship between LP and IP Optimal Values

Fig. 7 shows the relationship between LP optimal value and IP optimal value in each iteration of our algorithm (maximum number of columns to be added $\mathsf{N\_ADD\_MAX} = 1$). If LP optimal value decreases rapidly, IP optimal value

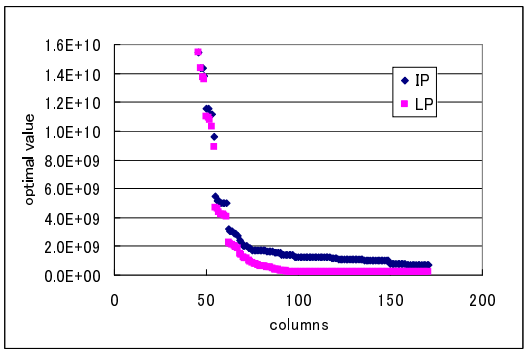

Fig. 7. Correlation of LP and IP optimal values

also decreases in such way. However, the gap between LP optimal value and IP optimal value remains large at the end.

5.2 Adding Multiple Columns

Fig. 8 and 9 are the plots of IP best objective values within 3 minutes and LP optimal value in the cases of single column adding (N_ADD_MAX = 1) and multiple column adding (N_ADD_MAX = 10) respectively. The variation of IP

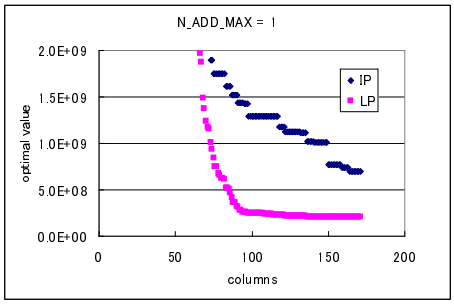

Fig. 8. Single column addition

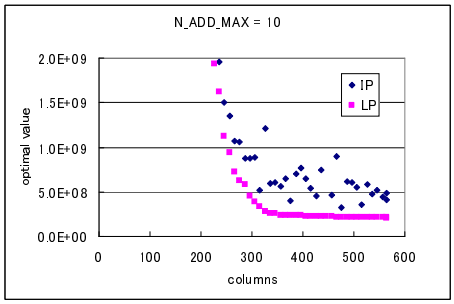

Fig. 9. Multiple columns addition

best objective value is from 3 minutes running time limit.

Adding multiple columns in each iteration greatly reduces the gap between LP and IP near the end. More precisely, adding multiple columns improves the final IP solution because LP optimal values are not very different in both cases: 2.06e+8 in the single case and 2.11e+8 in the multiple case.

By increasing maximum number of columns to be added N_ADD_MAX; $1 \rightarrow 10 \rightarrow 40 \rightarrow 100$, it is found that adding more columns improves the quality of IP solution as shown in Fig. 10 and 11. Also adding more columns increases the

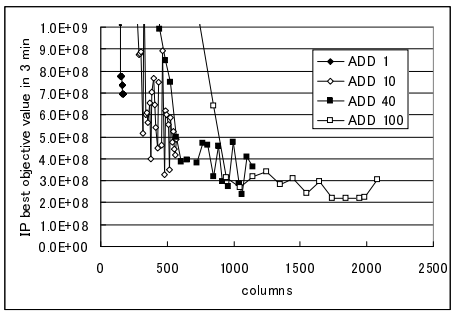

Fig. 10. Effect of multiple columns (1)

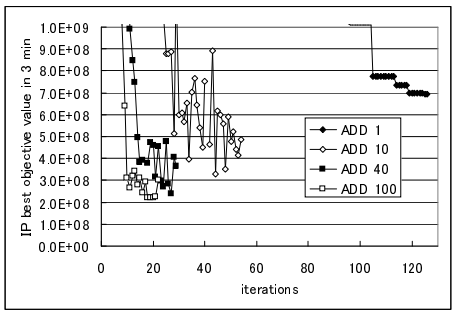

Fig. 11. Effect of multiple columns (2)

columns to be considered (Fig. 10), however, it decreases the total number of iterations (Fig. 11).

5.3 Starting with More Initial Columns

We generate more initial columns (than only the identical columns) by using the algorithm shown in Fig. 5 with three different small cutoff ratios; $\beta = 0.001, 0.005, 0.01$, and run our algorithm under N_ADD_MAX $= 250$. Table 2 shows that with more initial columns, it needs less iterations to get the results. IP optimal values may not become better because LP optimal values (i.e. the

Table 2. Effectiveness of more initial columns

	cutoff ratio	initial columns	iterations	IP best objective	LP optimal
default	—	46	20	2.06e+08	2.05e+08
case 1	0.001	655	10	2.53e+08	2.04e+08
case 2	0.005	2338	6	2.51e+08	2.09e+08
case 3	0.01	5000	3	2.12e+08	2.11e+08

lower bound of IP optimal values) are slightly increasing.

5.4 Sorting Candidate Columns by their Costs

The effectiveness of sorting candidate columns by their costs are examined for three cases: N_ADD_MAX $= 100$, 250, 625.

As shown in Table 3, there is not so much difference for data S and M no matter if sorting columns by their costs or not. However, looking at the result for data L, it seems very effective for handling large number of items.

Table 3. Effectiveness of sorting candidate columns by their costs

		IP best objective value (lower bound)		
ADD	sort by	data S	data M	data L
100	RC	1.41e+07	3.03e+08 (2.15e+08)	2.15e+09 (8.70e+08)
	RC&C	4.35e+07	4.62e+08 (optimal)	7.70e+08 (4.55e+08)
250	RC	5.91e+06	2.06e+08 (2.06e+08)	———— (8.31e+08)
	RC&C	5.33e+06	2.08e+08 (2.05e+08)	7.72e+08 (5.58e+08)
625	RC	5.44e+06	2.12e+08 (2.07e+08)	———— (4.67e+08)
	RC&C	4.58e+06	2.27e+08 (2.16e+08)	1.54e+09 (4.25e+08)

(RC: Reduced Cost, C: Cost, ADD: N_ADD_MAX.)

5.5 Comparison with Wang's Algorithm

In order to examine the performance of our algorithm, we compare with other two algorithms. They are Wang's Algorithm and its simple extension, which separates given items into subsets of items and apply Wang's Algorithm for each the subsets. In this examination, our algorithm starts with more initial columns ($\beta = 0.005$) and sorts candidate columns by both reduced costs and costs. The comparison of these three algorithms are shown in Table 4.

Table 4. Comparison of three algorithms

	parameters		IP best objective value (lower bound)		
	ADD	cutoff	data S	data M	data L
CG	100	—	4.58e+06	2.34e+08 (optimal)	6.18e+08 (4.40e+08)
	250	—	4.58e+06	2.35e+08 (2.02e+08)	5.43e+08 (4.58e+08)
EW	—	0.05	5.33e+06	2.00e+08	5.31e+08
	—	0.1	6.61e+06	2.33e+08	9.60e+08
WA	—	0.01	6.64e+07	2.74e+08 (2.71e+08)	———— (7.59e+09)
	—	0.05	4.54e+06	———— (1.08e+09)	———— (9.17e+09)

(CG: Column Generation based algorithm of ours, EW: Extended Wang's algorithm, WA: Wang's Algorithm, ADD: N_ADD_MAX.)

Wang's algorithm performs very well for small data, however it does not work well for large data i.e. no feasible solutions are obtained within 3 minutes for data L.

Extended Wang's algorithm and our algorithm have about even performance enough to apply to the practical problems. More tuning up of both algorithms will improve the performance.

For data L, our algorithm takes at most 315 seconds and extended Wang's algorithm takes at least 733 seconds. However this means nothing because the running time is highly dependent on the performance of IP solver and the complexity of checking pattern feasibility, where Extended Wang's algorithm solves more IPs and our algorithm generates more patterns respectively.

6 Conclusion

In this paper, we have presented a new algorithm for multiple staged two dimensional guillotine cutting stock problem. The algorithm is based on Column Generation approach by Gilmore and Gomory and applies a new heuristic method. Therefore, it can deal with very complicated problems in steel industry. Computational results shows the algorithm is well-behaved in application to practical problems.

References

1. COIN-OR: http://www.coin-or.org/.
2. N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44, 1977.
3. H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.
4. A. A. Farley. Practical adaptation of the Gilmore-Gomory approach to cutting stock problems. *OR Spektrum*, 10:113–123, 1988.
5. P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.
6. M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, to appear.
7. K. V. Viswanathan and A. Bagchi. Best-first search methods for constrained two-dimensional cutting stock problems. *Operations Research*, 41(4):768–776, 1993.
8. P. Y. Wang. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, 31(3):573–586, 1983.