Application of the Cutting Stock Problem to a Construction Company: A Case Study

Seda Alp, Gurdal Ertek*, S.Ilker Birbil

(*) Corresponding author

**Sabanci University, Faculty of Engineering and Natural Sciences,
Orhanli, Tuzla, 34956, Istanbul, Turkey**

**Tel: +90(216)483-9568
Fax: +90(216)483-9550
Email: ertekg@sabanciuniv.edu**

Application of the Cutting Stock Problem to a Construction Company: A Case Study

Seda Alp, Gurdal Ertek*, S.Ilker Birbil

Abstract

This paper presents an application of the well-known cutting stock problem to a construction firm. The goal of the 1Dimensional (1D) cutting stock problem is to cut the bars of desired lengths in required quantities from longer bars of given length. The company for which we carried out this study encounters 1D cutting stock problem in cutting steel bars (reinforcement bars) for its construction projects. We have developed several solution approaches to solving the company's problem: Building and solving an integer programming (IP) model in a modeling environment, developing our own software that uses a mixed integer programming (MIP) software library, and testing some of the commercial software packages available on the internet. In this paper, we summarize our experiences with all the three approaches. We also present a benchmark of existing commercial software packages, and some critical insights. Finally, we suggest a visual approach for increasing performance in solving the cutting stock problem and demonstrate the applicability of this approach using the company's data on two construction projects.

Submission areas: Decision Support System, Management Information Systems

Application of the Cutting Stock Problem to a Construction Company: A Case Study

Introduction

The cutting stock problem is encountered in many industries, including the construction industry. In constructions, steel bars (reinforcement bars) of certain lengths are needed in specified quantities. These bars are cut from long steel bars (with lengths as much as 1200 cms) according to the cutting patterns. The problem is to find the "optimal" cutting patterns, where the total number of long steel bars used is minimized, subject to the constraint that the desired shorter steel bars are cut in required quantities.

The Problem Setting

The Company

Mimag Makina Ltd, which is established in 1983, is a construction firm under the parent company, MIMAG Group[1]. Initial goal in founding the company group was to produce steel fabrication needs of the parent company, mainly formwork, scaffolding, and construction equipment. Later the products were diversified, and found a general market among contractors. At present, the company designs, produces, supervises formwork and scaffolding systems suitable for all kinds of structures such as dams, bridges, business centers, and industrial plants. Up to this date several projects were completed by Mimag, including housing projects (200 residential sites), industrial installations, railway electrification, silos, chimneys, and various civil, mechanical and structural installation jobs are executed.

Reinforced concrete, which is the combination of concrete and steel, is used in several projects of Mimag because of the quality of durability and for economic purposes. One of the projects that Mimag had dealt with was the “Diesel/Kerosene Hydroprocessing and CCR Reformer Project” implemented in Izmit Refinery of Tupras, Turkey (Figure 1a). Tupras is the top company in Turkey with respect to sales revenues. Vertical and horizontal reinforcements were built to replace some silos in the mentioned project. The other real data that we have used was from the project “Waste Water Treatment Plant” conducted in Adana, Turkey. The project was for the reinforcement of slab. To determine the amount of steel bars used in reinforced concrete, several methods are used to minimize the amount of scrap. The standard steel bars used in reinforced concrete steel bars are 1200 cm and have various diameters ranging from 6mm to 50mm (Figure 1b). These different steels with different lengths and diameters should be cut in an efficient manner such that the scrap is minimized (Figure 1c).

[1] http://www.mimag.com.tr

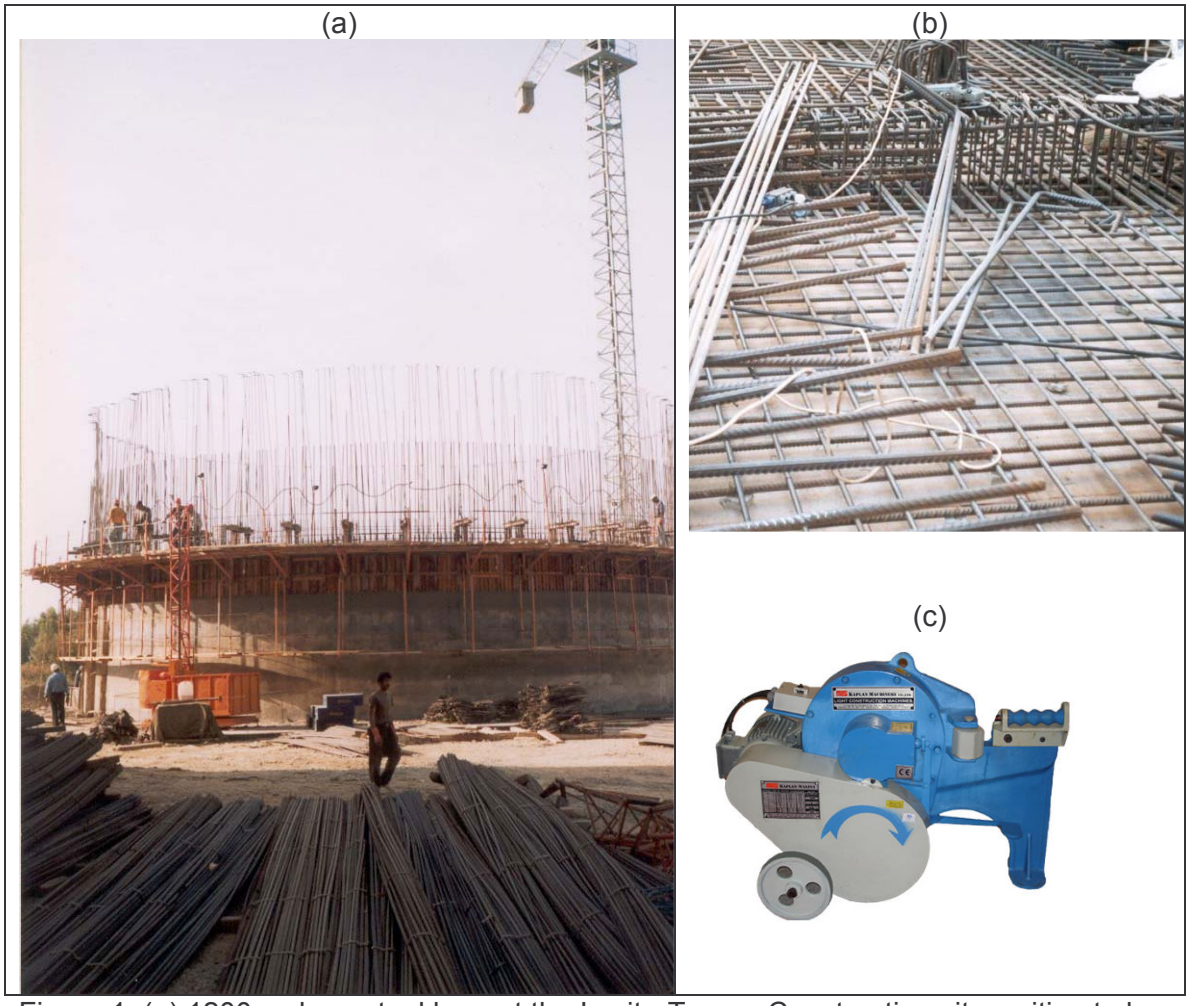

Figure 1. (a) 1200cm long steel bars at the Izmit - Tupras Construction site waiting to be cut. (b) The cut steel bars with different lengths and diameters. (c) Equipment used to cut the steel bars. Source: http://www.kaplanmakina.com

The 1D Cutting Stock Problem

The mathematical model for the 1D cutting stock problem is presented below:

SETS:
I: set of patterns
J: set of lengths

DECISION VARIABLES:
x_i : number of bars cut according to pattern i.

PARAMETERS:
a_{ij} : number of pieces of length j within one bar cut according to pattern i
b_j : required number of pieces of length j

THE MODEL

$$min \sum_{i \in I} x_i \qquad (1)$$

$$\text{s.t} \quad \sum_{i \in I} a_{ij} x_i \geq b_{j_i}, \forall j \in J \qquad (2)$$

$$x_i \geq 0 \text{, integer} \qquad (3)$$

The goal of the model is to minimize the objective function (1), which consists of the total number of long steel bars used. Constraint set (2) ensures that enough number of shorter bars are cut from long bars in required quantities. The decision variables are restricted to non-negative values (3).

One can obtain the optimal solution to the cutting stock problem by using an IP/MIP solver. For solving large instances of the problem, the reader is referred to Gilmore and Gomory (1961).

The Solution Approaches

Solving within the GAMS Modeling System

The first solution approach that we followed to tackle with Mimag's 1D cutting stock problem was building and solving an IP formulation within the GAMS Modeling System[2]. We have implemented our GAMS model based upon information provided in Kalvelagen (2003). We initially built the IP model for one of the subprojects within the Tupras Project, consisting of 8 different lengths (Appendix A). One advantage of using GAMS is being able to represent the data in matrix and vector forms. Another advantage is being assured that the optimal solution is found. However, there are major drawbacks of this approach:

- The patterns cannot be generated by the GAMS modeling language; they have to be generated through a separate program written in a general purpose programming language, such as C++.
- Adding a new pattern requires "hacking" into the GAMS model and adding a new row to the exact appropriate place. GAMS modeling system is very sensitive to incorrect syntax, and can give hard-interpret error messages when the required syntax is not followed.

Developing an IP-based Optimization Program

The shortcomings of using a system such as GAMS have led us to development of an IP-based optimization program for solving the problem. We had written a C++ program to provide the pattern data to be input into the GAMS model. However, this early implementation was employing nested loops (one loop for each additional length) to generate the feasible patterns. Such an implementation required modification of the source code for every new subproject data, since each subprojects can have different number of lengths required. Yet another issue was the fact that GAMS is a commercial

[2] www.gams.com

software, which requires purchase of a new license for each installation. We clearly needed a free-of-charge open-source modeling system or software library to use as the optimization engine.

lp_solve[3] is a free solver based on revised simplex algorithm for solving linear programming (LP) problems and the Branch-and-bound technique for solving IP and MIP problems. Even tough it is developed in ANSI C, it can be called from programs written in other programming languages, such as C++, Java, C#, and Visual Basic, as well. We have selected to code our program in Java[4], since one of the authors possesses proficiency in this language. We have used a "Java wrapper" that enables the communication of the Java code with the lp_solve optimization engine, and carried out our code development within the Eclipse IDE[5] (Integrated Development Environment).

Our Java program reads data from a text file, generates patterns efficiently through an array based representation of patterns, calls lp_solve to build and solve the IP model, and prints out the solution. Currently, our program has minimal GUI (Graphical User Interface), since GUI can consume great amounts of CPU time, and slow down the time to solve the cutting stock problem. The bottleneck of our current implementation is the generation of the patterns taking up to 1 minute for small-size instances with up to 17 lengths. One larger-size instance with 25 lengths took more than 10 minutes to solve. One possible solution to this problem is cutting some of the lengths from the next longer length. We utilized this approach for solving the mentioned instance.

We validated the correctness of our program by comparing its solution to the GAMS solution for three different instances.

Using Commercial Software

Our next approach to solving the Mimag's problem was to test commercial 1D cutting stock software available on the internet. The software packages were searched using the key word "1 dimensional cutting stock program" at www.google.com. The first ~30 pages of the search results were scanned to retrieve applicable software. Only seven packages were found to have demo versions available that can handle the largest instance (with 25 lengths) that we had. These packages and their web sites are listed in Appendix B.

Benchmarking Results

The Performance Data

Some important questions that we had in mind -upon completion of our Java program- were the following;

Q1. How effective is our program compared to the commercial packages that we downloaded?
Q2. How do the commercial packages compared amongst them?

[3] http://lpsolve.sourceforge.net/5.5/
[4] htpp://java.sun.com
[5] www.eclipse.org

Q3. Would it be considerably more effective to use two different software packages at the same time for each subproject and implement the better of the two solutions that they provide?
Q4. Does batching of two or more subprojects and determining the best solution for the batch increase performance (decrease the cost)?

We have found the answers to the first three questions based upon Mimag's data. For answering the fourth question, we propose a visual approach that requires drawing a chart to determine which subprojects to merge. We present our visual approach in the next section.

We have carried out a performance benchmark of our program and the commercial packages by using the data sets from Izmit (Tupras) and Adana projects of Mimag. The total number of bars used in our program's solution and the commercial packages' solutions are shown in decreasing order in Table 1. The software marked with a (*) is the one which Mimag was planning to purchase.

Table 1. The total number of bars found by the benchmarked software

Software Name	Total number of Bars
Java Program	10399
Software 1	10400
Software 2	10402
Software 3	10403
Software 4	10410
Software 5*	10419
Software 6	10577
Software 7	11858

Insights

As can be seen from Table 1, our Java program outperformance all the commercial packages. This is naturally expected, since we know from optimization theory that IP solution gives the optimal objective function value. The next thing that we noticed is that none of the commercial packages have achieved the optimal total number of bars. This indicates that the software packages are not implementing an optimal algorithm, but are implementing heuristic techniques.

We noticed that the first five software packages give solutions very close to the optimal. So a construction company can select any of these five software packages, based on other criteria such as price, on-line support, usability, data import/export capabilities, and reporting quality (including visualization of optimal cutting patterns and their quantities). One very important observation is that Software 6 and especially Software 7 perform poorly. A company should definitely avoid using Software 7, which also happens to be the most expensive package amongst all. Software 7 is distinguished from all others by its capability to import/export to and from MS Excel. Meanwhile, Software 6 has the best GUI amongst all packages. Thus, the software selection should be based on performance tests, rather than price, data import/export capabilities, or GUI. We have so far answered Q1 and Q2. Since the software packages that perform well are extremely close to the optimal there is no need to use two of those software packages, thus answering Q3.

A Visual Approach for Determining How to Batch Subprojects

We now present a visual approach that enables a decision maker to select which subproject-diameter combinations can be batched (combined) to be cut simultaneously. This approach is an answer to Q4 posed in the previous section. We project that the approach we describe below can reduce the number of long bars cut (which is the objective function of 1D cutting stock problem), and suggest that it can be implemented in cutting stock software packages.

Figure 2 shows the chart that we propose[6], where each point represents a subproject-diameter combination. The x-axis shows values of "Optimal number of bars used", the y-axis shows values of "Diameters" (in cm), and the color of the points shows "Optimal usage" (as a percentage) for that subproject, where dark colors denote lower usages (bad performance). To generate this chart, the cutting stock problem has to be solved for each subproject, and the three values that will be displayed in the chart have to be calculated. This is a fairly easy thing to do, since steel bar requirements for each subproject of a project is determined apriori and stays fixed through out the construction project.

The approach we suggest is the following:

1. Start with the highest "Diameter" value on the y-axis and examine the points that reside on that row. We start searching from the biggest diameter since the cost of steel bars is depended upon the weight, which is a quadratic function of the diameter.
2. If there is a point with a dark color on that row which has a large "Optimal number of bars used" value on the x-axis, then consider batching that with another point (subproject-diameter combination). For example in Figure 2, we first consider batching points A and B.
3. Test using software whether the batching possibility in Step 2 brings any improvements. If batching brings significant improvements go ahead and carry out the batching while cutting the steel bars. If batching does not bring significant improvement carry out the cutting separately.
4. Return back to Step 1, considering the next smaller diameter, until you cannot find any batching possibilities that could bring significant savings. For example, the next batching possibility in Figure 2 is combining points C and D.

When we applied the proposed visual approach, we observed that the two considered projects (Izmit and Adana) did not include great improvement opportunities through batching. We even observed that batching could result in a worse performance. Therefore, the decision maker should be aware that batching is not always a good option. Even though we have not gained meaningful improvements by batching, when using our data, we believe that for other projects there can be significant savings. One could especially expect good opportunities when there are a large number of projects/subprojects managed simultaneously, and a large number of steel bars to be cut. We suggest carrying out more extensive tests as a future research area.

[6] The chart is generated using Miner3D Software (http://www.miner3d.com)

One important issue in batching is the additional inventory of cut steel bars that will have to be stored. This translates into additional inventory holding costs, which are incurred due to time value of money. Any analysis of batching should also consider this additional cost.

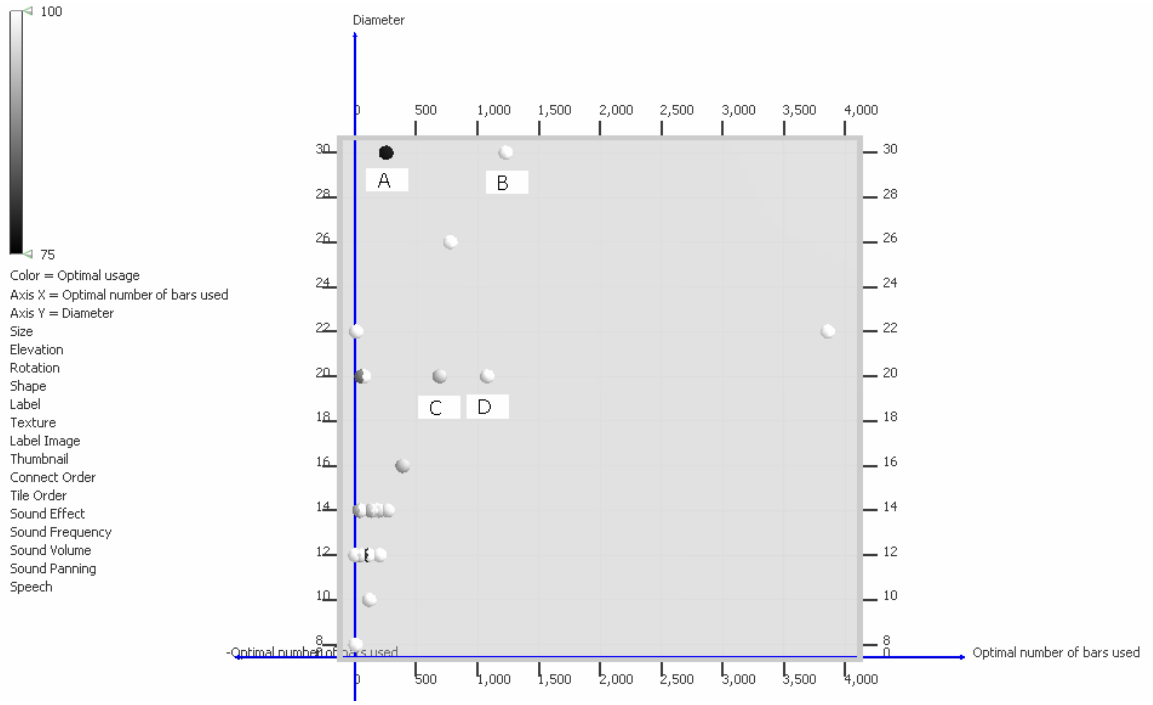

Figure 2. The chart that displays performance data on subprojects.

Conclusions

Motivated by a real world problem encountered by a construction firm, we have tested and benchmarked commercial software packages for solving the 1D cutting stock problem. Our findings suggest that available software packages should be tested with sample project data before adoption. We observed that software packages with the best import/export capabilities and/or GUI can perform very poorly. Besides, a higher price does not imply a good performance at all, as the worst software in our study was also the most expensive one. We concluded our paper with a visual approach that can be implemented in software packages to enable interactive decision making with respect to batching. We believe that our study will offer the decision makers with general guidance on software selection and the software companies with a feature that they can use to enhance the functionality of their products.

Acknowledgement

We sincerely thank Necati Alp at Mimag for motivating our study, providing the real world data, and continuously supporting us. We also thank Sevket Atilla at Kaplan Machinery for allowing us to use their equipment's photograph in our paper.

References

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting stock problem. *Operations Research*, vol: 9, 848--859

Kalvelagen, E. (2003). *Column Generation with Gams*. Gams Development Corp. Washinton DC.

APPENDIX A.

```
Sets
i widths /w1*w8/              *different lengths of the steel bars.
j pattern/p1*p38/;            *patterns in which combination they are going to be used.

scalar r raw width /1200/;    *bar length is set to 1200.

table demanddata(i,*)         *demand table for the lengths and their quantities.
         width   demand
w1       100     100
w2       200     2
w3       300     2
w4       493     250
w5       590     2
w6       630     2
w7       780     2
w8       930     2;

table patterndata(j,i)        *pattern table with respect to lengths

          w1       w2       w3       w4       w5       w6       w7       w8
p1        2        2        2        0        0        0        0        0
p2        4        1        2        0        0        0        0        0
p3        5        2        1        0        0        0        0        0
p4        6        0        2        0        0        0        0        0
p5        7        1        1        0        0        0        0        0
p6        8        2        0        0        0        0        0        0
p7        9        0        1        0        0        0        0        0
p8        1        0        1        0        0        0        0        0
p9        1        2        0        0        0        0        0        0
p10       0        2        1        1        0        0        0        0
p11       1        0        2        1        0        0        0        0
p12       2        1        1        1        0        0        0        0
p13       3        2        0        1        0        0        0        0
p14       4        0        1        1        0        0        0        0
p15       5        1        0        1        0        0        0        0
p16       7        0        0        1        0        0        0        0
p17       0        0        2        0        1        0        0        0
p18       1        1        1        0        1        0        0        0
p19       2        2        0        0        1        0        0        0
p20       3        0        1        0        1        0        0        0
p21       4        1        0        0        1        0        0        0
p22       6        0        0        0        1        0        0        0
p23       0        1        0        2        0        0        0        0
p24       2        0        0        2        0        0        0        0
p25       1        0        0        1        1        0        0        0
p26       0        0        0        0        2        0        0        0
p27       0        2        0        0        0        0        1        0
p28       1        0        1        0        0        0        1        0
p29       2        1        0        0        0        0        1        0
p30       4        0        0        0        0        0        1        0
p31       0        1        0        0        0        0        0        1
p32       0        1        1        0        0        1        0        0
p33       1        2        0        0        0        1        0        0
p34       2        0        0        0        0        0        0        1
p35       2        0        1        0        0        1        0        0
p36       3        1        0        0        0        1        0        0
p37       5        0        0        0        0        1        0        0
p38       0        0        0        1        0        1        0        0;
```

```
parameter d(i);                                 *demand in taken from the table as input.
d(i) = demanddata(i, 'demand');

parameter a(i,j);                               *the amount of the patterns that are going
                                                to be used.
a(i,j) = patterndata(j,i);

integer variable x(j) 'pattern used';
variable z 'objective';

x.up(j) = sum(i, d(i));

equations
numpat 'number of patterns used(objective)'
demand(i) 'meet demand';

        numpat.. z=e=sum(j, x(j));              *minimization of the patterns that are going
                                                to be used
demand(i).. sum(j, a(i,j)*x(j))=g=d(i);         *demand should be satisfied.

model cutting /all/;
solve cutting using mip minimizing z;
display x.l, x.m;
```

Appendix B

The commercial software packages benchmarked in this paper are listed in random order below. The numbering of the software packages within the paper is not with respect to this order.

http://www.astrokettle.com/pr1dsc.html
(1D Stock Cutter)

http://www.downloadjunction.com/product/software/30836/
(Optimumcut)

http://www.rrdrummond.com/demo1.htm
(The Itemizer 7.0)

http://www.softplatz.com/Soft/Graphics/CADs/Real-Cut-1D.html
(Real Cut 1d)

http://www.shareup.com/Pipe_Cutting_Suite-download-7887.html
(Pipe Cutting Suite 3.11)

http://www.softpicks.net/software/CutLogic-1D-9970.htm
(CutLogic)

http://tucows.menanet.net/preview/333725.html
(Plus1D)