AIMMS Modeling Guide - Cutting Stock Problem

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

This documentation was typeset by AIMMS B.V. using LATEX and the LUCIDA font family.

Chapter 20

A Cutting Stock Problem

This chapter

This chapter applies a *delayed column generation* technique to find a set of optimum cutting patterns for a class of cutting stock problems. Each pattern is essentially a column of the underlying linear program. In practical applications, the number of cutting patterns can be extremely large. However, instead of considering the millions of possible cutting patterns, a submodel of the cutting stock problem is systematically built up with new patterns until it contains the optimum solution. The new patterns are added to the submodel by solving an auxiliary integer program, referred to as the *column pattern generation* model. The chapter begins with a basic model which is then extended.

References

The methodology for cutting stock problems dates back to work of Gilmore and Gomory ([Gi61, Gi63]). A good exposition on this subject and its underlying theory can also be found in [Ch83].

Keywords

Linear Program, Integer Program, Simplex Method, Column Generation, Mathematical Derivation, Customized Algorithm, Auxiliary Model, Worked Example.

20.1 Problem description

This section

This section introduces a class of *cutting stock problems*, which are typically encountered in the paper and textile industries.

Raws and finals

Materials such as paper and textiles are often produced in long rolls, the length of a truck trailer for instance. These long rolls are referred to as *raws*. These raws are subsequently cut into smaller portions called *finals*, with their sizes specified by customers.

Slicing raws

A raw can be sliced all the way through so that the diameter of each final has the same diameter as the original raw. This is usually what happens when rolls of paper are cut. The slicing of a raw is illustrated in Figure 20.1.

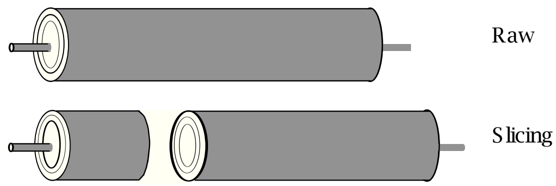

Figure 20.1: Slicing raws of paper and textile

Demand requirements

Assume that a production scheduler has a list which specifies the required number of each final size. He must then develop a production schedule detailing the number of rolls and how they should be cut to meet demand.

Objective

The objective of the scheduler is to determine the most economical way to meet the demand. It is assumed that there are no storage space constraints. The objective becomes to minimize the total number of rolls required to make the finals. In this chapter, the more general multi-period inventory case is not addressed. When time periods are considered, the objective is not just to minimize the number of raws used, but also to consider the storage costs involved.

20.2 The initial model formulation

Investigating the structure

A natural inclination when initially constructing a model for the cutting stock problem is to consider two sets, namely 'Raws' and 'Finals'. However, on closer inspection, you will note there is only one kind of raw in the problem description. The question then arises: does the set 'Raws' contain only one element (reflecting that only one kind exists), or does this set contain an undetermined number of elements (one for each raw to be cut)? Similarly, should the set 'Finals' contain each final or just the possible sizes of finals? The answer to these questions is not immediately apparent, and further analysis is required.

Cutting patterns

If you have to write down a model for a small example, it is likely you will develop the concept of a cutting pattern. A *cutting pattern* is a specific recipe stating for each size of final how many finals are cut from a single raw. Of course, there are several such recipes. For small examples the size of the set of cutting patterns is not exorbitant, but in most real problems the number of possible cutting patterns could be in the millions.

The set 'Finals'

When you have adopted the concept of a cutting pattern as a building block for the model, it is also clear that the set 'Finals' should contain the possible sizes of finals (and not each individual final that is demanded by the customers). This is because the cutting pattern is defined in terms of possible sizes of finals.

Verbal model description

Assume for the duration of this section that the size of the set of cutting patterns is workable. A verbal statement of the model is then as follows.

Minimize: *the number of raws to be used*

Subject to:

for all possible sizes of finals: the number of finals produced from cutting raws according to the set of allowable cutting patterns must meet the demand.

Mathematical description

The following integer program is a mathematical representation of the verbal model in the previous paragraph.

Indices:

p	*cutting patterns*
f	*finals*

Parameters:

d_f	*demand for final f*
a_{fp}	*number of finals f in cutting pattern p*

Variable:

x_p	*number of raws cut with cutting pattern p*

Minimize:

$$\sum_p x_p$$

Subject to:

$$\sum_p a_{fp} x_p \geq d_f \qquad \forall f$$

$$x_p \geq 0 \quad \textit{integer} \qquad \forall p$$

Relax integer requirement

As the number of cutting patterns increases, the solution time of the underlying integer programming solver will also increase. In which case, an attractive alternative may be to drop the requirement that x_p is integer, and just solve the corresponding linear programming model. A straightforward rounding scheme may then be quite sufficient for all practical purposes. The rounded integer solution may not be optimal, but it can easily be made to satisfy the demand requirements.

Largest In Least Empty

A simple and useful heuristic is 'Largest In Least Empty' (LILE). This heuristic is loosely described in the following four steps.

1. Round the fractional solution values downwards, and determine the unmet demand.
2. Sort the finals in the unmet demand from largest to smallest.
3. Place the largest final from the unmet demand in the least empty raw that can contain this final. If this is not possible, an extra raw must be added.

4. Continue this process until the sorted list of finals from the unmet demand is completely allocated.

It is possible that a pattern generated by this algorithm is one of the patterns used in the relaxed integer programming solution (see Table 20.2 in which pattern 12 is generated again). The LILE algorithm tends to minimize the number of extra raws required, and turns out to work quite well in practice.

Example data and ...

Consider an example where the raws are ten meters long, and there are four sizes of finals, namely, 450 cm, 360 cm, 310 cm and 140 cm. The raws can be cut using thirty-seven cutting patterns as shown in Table 20.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
450 cm	2	1	1	1	1	1	1	1	1										
360 cm		1	1							2	2	2	1	1	1	1	1	1	1
310 cm				1	1								2	1	1	1			
140 cm		1		1		3	2	1		2	1			2	1		4	3	2

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
450 cm																		
360 cm	1	1																
310 cm			3	2	2	2	1	1	1	1	1							
140 cm	1			2	1		4	3	2	1		7	6	5	4	3	2	1

Table 20.1: Thirty-seven cutting patterns to cut a raw of size 1000

... the solution

With the use of all thirty-seven cutting patterns, the minimum number of raws required according to the linear programming solution is $452\frac{1}{4}$. As expected, the number of times a cutting pattern is used in the optimal solution is fractional. One of the optimal solutions is listed in Table 20.2. Rounding this optimal fractional linear programming solution, using the LILE heuristic, gives an integer objective function value of 453 required raws. This number of 453 is optimal, because the linear programming objective function value of $452\frac{1}{4}$ is a lower bound for the integer objective function value, and 453 is the first integer in line.

20.3 Delayed cutting pattern generation

Cutting stock submodel approach

Problems commonly encountered in the paper industry involve raws and finals of arbitrary sizes. The number of possible cutting patterns can then grow into the millions and the approach of the previous section is no longer workable. This section describes an algorithm that makes the problem workable by limiting the number of possible cutting patterns. Instead of going explicitly through millions of cutting patterns, a *cutting stock submodel* is systematically

Finals	**Optimal Patterns**				**LILE Patterns**			**Demand**
	01	10	12	13	02	12	30	
450 cm	2				1			97
360 cm		2	2	1	1	2		610
310 cm				2			1	395
140 cm		2			1			211
Fractional Solution	$48\frac{1}{2}$	$105\frac{1}{2}$	$100\frac{3}{4}$	$197\frac{1}{2}$				
LILE Solution	48	105	100	197	1	1	1	

Table 20.2: The solutions: linear programming versus LILE

built up to contain the optimum solution by adding patterns identified by solving an auxiliary integer program, referred to as the *column pattern generation model*.

Algorithm description

The first step of the algorithm is to create a submodel of the cutting stock problem which contains a set of cutting patterns which will satisfy the requirements. Clearly, this initial set will not (necessarily) be optimal. This submodel is then solved. Using the resulting shadow prices in conjunction with simplex method theory, it is possible to formulate an auxiliary model (cutting pattern generation model) integer program. Solving this model identifies one cutting pattern which is then added to the cutting stock submodel to improve its objective (i.e. to reduce the number of raws). The cutting stock submodel with this extra pattern, is then solved. The process is repeated (using updated shadow prices) until the submodel contains the set of optimum cutting patterns. In practice, the total number of new cutting patterns generated by the cutting pattern generation model is quite small and so the overall algorithm is very efficient.

New approach

Like in Chapter 19, this algorithm is based on the simplex method. However, the approach differs because it takes advantage of the fact that all objective function coefficients in the cutting stock model are identical, while the ones in the file merge model are not. You are referred to Chapter 19 for a discussion of the simplex method and the application of shadow prices and reduced costs.

Iteration between two models

The iterative solving of the cutting stock submodel and the cutting pattern generation program is summarized below.

```
Initialize cutting stock submodel
WHILE progress is being made DO
    Solve cutting stock submodel
    Solve cutting pattern generation model
    IF new cutting pattern will lead to improvement
        THEN add it to cutting stock submodel
ENDWHILE
```

Cutting stock model initialization

The cutting stock submodel can be initialized in many ways. The simplest option is to include one pattern for each final size. With each pattern consisting of the maximum number of finals that can be cut from the raw. For example, for the model in the previous section, you could use patterns 1, 12, 22 and 31. By selecting patterns which include all final sizes, then the first solution will be feasible (but not necessarily optimal).

Additional patterns

In addition to the four initial patterns, patterns 10 and 13 were generated by the cutting pattern generation program. These six patterns were sufficient to determine an optimal solution of the cutting stock model.

Mathematical formulation

Assume there is some cutting pattern y which is not part of the cutting stock submodel. Let y_f be a component of this vector. Each such component corresponds to the number of finals of size f used in the cutting pattern. In addition to y_f, let λ_f denote the shadow price associated with each demand requirement f in the cutting stock submodel. Then cutting pattern y should be added to the submodel whenever

$$1 - \sum_f \lambda_f y_f < 0$$

This condition is the reduced cost criterion in the simplex method when applied to the cutting stock model.

An auxiliary model

An auxiliary cutting pattern generation model can now be proposed based on the following three observations. First of all, the numbers y_f making up a cutting pattern must be nonnegative integers. Secondly, their values must be such that the cutting pattern does not require more than the total width of a raw. Thirdly, the new cutting pattern values should offer the opportunity to improve the objective function value of the reduced cutting stock problem as discussed in the previous paragraphs. Let w_f denote the required width of final f, and let W denote the total width of a raw. Then the three observations can be translated into the following model constraints.

$$\begin{array}{rcll} y_f & \geq & 0, \quad \text{integer} \quad \forall f & (1) \\ \sum_f w_f y_f & \leq & W & (2) \\ 1 - \sum_f \lambda_f y_f & < & 0 & (3) \end{array}$$

Transformation into a MIP model

The above model formulation contains a strict inequality and it must be manipulated before using a solver which is based on inequalities. There is one observation that makes it possible to rewrite the above system as a mixed-integer linear programming model. Whenever the term $\sum_f \lambda_f y_f$ is greater than one, the last inequality is satisfied. You could write this term as an objective function to be maximized subject to the first two constraints. Whenever the optimal value of this mathematical program is greater than one, you have found an interesting cutting pattern. Whenever this optimal value is less than or equal to one, you know that there does not exist a cutting pattern that can improve the objective value of the reduced cutting stock problem expressed as $\sum_f \lambda_f y_f$. This observation results in the following cutting pattern generation model.

Cutting pattern generation model

Maximize:

$$\sum_f \lambda_f y_f$$

Subject to:

$$\sum_f w_f y_f \leq W$$

$$y_f \geq 0, \quad \textit{integer} \qquad \forall f$$

Allowing inaccuracies

The implementation of this model in AIMMS is straightforward since the λ_f's are calculated during each solve iteration and can be directly accessed. It is important to allow numerical inaccuracies in the computed shadow prices. For this reason, it is generally advisable to use a small tolerance $\delta > 0$ when verifying whether a new patterns will lead to improvement. The mathematical condition to be verified for progress then becomes

$$\sum_f \lambda_f y_f \geq 1 + \delta$$

The value of δ is typically in the order of 10^{-4}. When δ is too small, the overall algorithm may not converge. In that case the cutting pattern generation model produces the same new pattern every time it is solved.

Identical objective coefficients

The transformation of the initial auxiliary model into a MIP model is strongly dependent on the property that all objective function coefficients are identical. Without this property, it is not meaningful to translate a strict inequality of the form $c_y - \sum_f \lambda_f y_f < 0$ into an objective function of the form $\sum_f \lambda_f y_f$ as has been done. Without identical coefficients, the stopping criterion in the delayed column generation algorithm is no longer correct. The reason is that when $c_{y^*} - \sum_f \lambda_f y_f^* \geq 0$ holds for an optimal solution y^* (indicating termination), it is still possible that $c_{\hat{y}} - \sum_f \lambda_f \hat{y}_f < 0$ holds for some other solution $\hat{y}$ due to a smaller value of $c_{\hat{y}}$.

20.4 Extending the original cutting stock problem

This section

In this section three possible extensions to the original cutting stock model are introduced and subsequently incorporated into a single new cutting stock model.

Multiple types of raws

One extension is to include several types of raws. Each type with its own length. This will result in a large increase in the overall number of cutting patterns to be considered when solving the underlying model.

Purchase cost

Another extension is to include the purchase cost of each type of raw. This changes the objective function of the underlying model. Rather than minimizing the number of raws to be used, the objective is to minimize the cost of raws.

Capacity limitation

The third extension is to introduce machine capacity restrictions. It is assumed that there is an upper bound on the number of raws of each type that can be cut on the available machines during a fixed period of time. It is assumed that these upper bounds are not dependent on each other.

Model formulation

The resulting extended cutting stock problem can be translated into the following mathematical model.

Indices:

r	*types of raws*
p	*cutting patterns*
f	*finals*

Parameters:

c_r	*unit cost of raws of type r*
d_f	*required demand for final f*
a_{fpr}	*number of finals f in pattern p for raws of type r*
k_r	*available capacity for raws of type r*

Variable:

x_{pr}	*number of raws of type r cut with pattern p*

Minimize:

$$\sum_{p,r} c_r x_{pr}$$

Subject to:

$$\sum_{pr} a_{fpr} x_{pr} \geq d_f \qquad \forall f$$

$$\sum_{p} x_{pr} \leq k_r \qquad \forall r$$

$$x_{pr} \geq 0, \quad \textit{integer} \qquad \forall (p,r)$$

Delayed pattern generation . . .

With the extension of multiple raws and varying cost coefficients, it is no longer clear whether the delayed column generation algorithm of the previous section is applicable. The previous auxiliary model, finds a cutting pattern for just a single size of raw, and the contribution of a new cutting pattern is compared to the constant value of 1.

. . . can still be applied

Observe, however, that the cost coefficients are constant for all patterns belonging to a single type of raw. This implies that the idea of cutting pattern generation can still be applied as long as each type of raw is considered separately. The resulting generalized delayed pattern generation algorithm is summarized below.

```
WHILE progress is being made DO
    Solve cutting stock submodel
    FOR each type of raw DO
        Solve cutting pattern generation model
        IF new cutting pattern will lead to improvement
            THEN add it to cutting stock submodel
    ENDFOR
ENDWHILE
```

Capacity constraint modification

As a result of the extra capacity constraints, the condition to check whether a new pattern will lead to improvement needs to be modified. Let π_r denote the shadow price associated with the capacity constraint for raws of type r obtained after solving the cutting stock submodel. Then any cutting pattern y^r produced by the auxiliary pattern generation model for raws of type r will lead to an improvement only if

$$c_r - \pi_r - \sum_f \lambda_f y_f^r < 0$$

This condition is the reduced cost criterion in the simplex method applied to the extended cutting stock model developed in this section.

Allow for inaccuracies

Recall from the previous section that the inaccuracies in the shadow price computation need to be taken into account. By again introducing a $\delta > 0$, the above condition can be rewritten as

$$\sum_f \lambda_f y_f^r \geq c_r - \pi_r + \delta$$

Alternative solution sequence

In the above delayed pattern generation algorithm summary, the auxiliary model is solved for every type of raw r before solving the next cutting stock submodel. An alternative approach is to solve the cutting stock model as soon as one new interesting pattern has been found. You might want to investigate this alternative when the time required to solve the cutting pattern generation model is large relative to the time required to solve the cutting stock submodel.

A worked example

Consider three types of raws (600 cm, 800 cm and 1000 cm) and the same four final sizes as in Section 20.2 (140 cm, 310 cm, 360 cm and 450 cm). The corresponding demand for these finals is 100, 300, 500 and 200 respectively. The unit cost and the available capacity associated with each raw type is presented in Table 20.3.

Raw	c_r	k_r
600 cm	25	200
800 cm	30	200
1000 cm	40	300

Table 20.3: Raw type data

20.5 Summary

In this chapter a cutting stock problem was translated into a mathematical formulation based on the concept of cutting patterns. Due to the large number of cutting patterns in practical applications, a delayed column generation approach using a cutting stock submodel was introduced. This approach solves an auxiliary integer programming model to produce a single new cutting pattern which is then added to the cutting stock submodel. The auxiliary model has been developed in detail, and the overall solution approach has been outlined. The algorithm can easily be implemented in Aimms.

Exercises

20.1 Implement the cutting stock model described in Section 20.2 using the example data presented in Table 20.1. Write a small procedure in Aimms to round the optimal linear programming solution using the Largest-In-Least-Empty heuristic.

20.2 Implement the delayed cutting pattern generation approach described in Section 20.3 in Aimms as an iteration between two models. Check whether the optimal solution found is the same as the one found previously.

20.3 Implement the extension of the initial cutting stock model, which is described in Section 20.4. Verify that the optimal objective function value equals 15,600 using the example data from Section 20.4.

Bibliography

[Ch83] V. Chvátal, *Linear programming*, W.H. Freeman and Company, New York, 1983.

[Gi61] P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem part i*, Operations Research **9** (1961), 849–859.

[Gi63] P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem part ii*, Operations Research **11** (1963), 863–888.