Solving a Bi-Criterion Cutting Stock Problem with Open-Ended Demand: A Case Study
Author(s): A. W. J. Kolen and F. C. R. Spieksma
Source: *The Journal of the Operational Research Society*, Vol. 51, No. 11 (Nov., 2000), pp. 1238-1247
Published by: Palgrave Macmillan Journals on behalf of the Operational Research Society
Stable URL: http://www.jstor.org/stable/254209
Accessed: 13/07/2009 05:38

Your use of the JSTOR archive indicates your acceptance of JSTOR's Terms and Conditions of Use, available at http://www.jstor.org/page/info/about/policies/terms.jsp. JSTOR's Terms and Conditions of Use provides, in part, that unless you have obtained prior permission, you may not download an entire issue of a journal or multiple copies of articles, and you may use content in the JSTOR archive only for your personal, non-commercial use.

Please contact the publisher regarding any further use of this work. Publisher contact information may be obtained at http://www.jstor.org/action/showPublisher?publisherCode=pal.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

JSTOR is a not-for-profit organization founded in 1995 to build trusted digital archives for scholarship. We work with the scholarly community to preserve their work and the materials they rely upon, and to build a common research platform that promotes the discovery and use of these resources. For more information about JSTOR, please contact support@jstor.org.

Palgrave Macmillan Journals and *Operational Research Society* are collaborating with JSTOR to digitize, preserve and extend access to *The Journal of the Operational Research Society.*

http://www.jstor.org

Journal of the Operational Research Society (2000) 51, 1238–1247 ©2000 Operational Research Society Ltd. All rights reserved. 0160-5682/00 $15.00

www.stockton-press.co.uk/jors

Solving a bi-criterion cutting stock problem with open-ended demand: a case study

AWJ Kolen and FCR Spieksma*

Maastricht University, The Netherlands

We consider a real-life cutting stock problem with two types of orders. All orders have to be cut from a given number of raws (also known as stock unit, master reel or jumbo). For each order the width of the final (also known as reels or units) and the number of finals is given. An order is called an exact order when the given number of finals must be produced exactly. An order is called an open order when at least the given number of finals must be produced. There is a given maximum on the number of finals that can be produced from a single raw which is determined by the number of knives on the machine. A pattern specifies the number of finals of a given width that will be produced from one raw. A solution consists of specifying a pattern for each raw such that in total the number of finals of exact orders is produced exactly and at least the number of finals of open orders is produced. There are two criteria defined for a solution. One criterion is the cutting loss: the total width of the raws minus the total width of the produced finals. The second criterion is the number of different patterns used in the solution. We describe a branch-and-bound algorithm that produces all Pareto-optimal solutions.

Keywords: cutting stock problem; multi-objective; practice of OR

Introduction

Consider the following setting that is encountered in a plant of a medium sized producer of abrasives. Abrasives (like other material as paper or textiles) are manufactured in rolls of large width. These rolls, referred to as *raws* (also known as stock unit, master reel or jumbo) are cut into rolls of smaller width, referred to as *finals* (also known as reels or units). This is carried out by a cutting machine that is equipped with a number of knives. In the plant that we consider up to 200 types of abrasives are produced in each week. For each type of abrasives, a set of orders is given that should be produced in the upcoming week. An order consists of two numbers: a *width* of a final, and a *number* of finals (also referred to as demand). In addition, for each type of abrasives a set of widths is prespecified that are popular in demand. Producing too many finals of one of these widths causes no concern since these finals will eventually be sold at a later time. However, if a width does not belong to this set, then it is required that exactly the number of finals as given in the order must be produced. Therefore, in this manner, the plant distinguishes between so-called *open* orders and *exact* orders.

Obviously, in order to satisfy demand for a type of abrasives, raws of the appropriate type have to be cut into finals in a feasible way. Cutting a raw into finals is done according to a *pattern*, that is, a pattern specifies the number of finals of a given width that will be produced from one raw. A solution consists of specifying a pattern for each raw of the appropriate type such that in total exactly the number of finals of exact orders and at least the number of finals of open orders is produced. What determines the quality of such a solution? After some discussions with the management of the plant, it became clear that two criteria are relevant.

Firstly, as expected, minimising cutting loss is an important issue. This is especially true for the plant under consideration, since about 40% of the plant's turnover comes from material that is cut by cutting machines as described above. Therefore, minimising cutting loss simply minimises costs.

Secondly, minimising the number of different patterns is an important issue as well. The relevance of this objective stems from the fact that between any two consecutive raws that are cut according to different patterns, the machine undergoes a setup. More in particular, the knives of the machine have to be reset according to the new pattern. This task is performed manually, and takes a varying amount of time, up to three quarters of an hour. (In fact, this amount of time depends on how much two patterns differ. Of course, one could then conceive of a more sophisticated objective that takes into account a weight for a pair of patterns. For reasons of simplicity we did not follow such an approach). Therefore, minimising the number of different patterns

**Correspondence: Dr FCR Spieksma, Department of Mathematics, Maastricht University, PO Box 616 NL-6200 MD Maastricht, The Netherlands.* E-mail: spieksma@math.unimaas.nl

leads to gains in time needed to produce the orders, and to a more stable production process.

Prior to our involvement, an experienced planner constructed manually a proposal how to cut raws into finals. This situation was considered unsatisfactory for a number of reasons:

- The dependence of the plant on the experience of a single planner. Of course, relying upon implicit knowledge of a single person is a situation that, if possible, one would want to avoid.
- Only a single solution of unknown quality was constructed. Whether improvements over the manually constructed solutions were possible, or whether alternative solutions of comparable quality were present, was not known.
- Some of the solutions constructed by the planner were infeasible. To explain this, consider the following: for reasons of quality, it is stipulated that 10 mm from the left side as well as 10 mm from the right side of each raw must be cut and considered as waste. So, for example from a raw with width 1400 mm, only 1380 mm could be used for producing finals. Some manually constructed solutions used parts of these 10 mm. Evidently, this was considered as something that should be avoided.

These reasons motivated our involvement. This paper describes our approach to the problem described above. A key feature in our approach is the decision not to combine both criteria into a single objective function but instead to solve a bi-criterion problem. This decision was also motivated by observing that management of the plant expressed the need for a *set* of high quality solutions (see above), one of which could be chosen for implementation depending upon the circumstances. The set of solutions that we generate is in fact the set of Pareto-optimal solutions. Another important aspect of our approach is that we treat the number of raws (for a type of abrasives) as a given parameter (unlike the classical cutting stock problem). Due to the type of instances that arise in our setting this seems a justified approach. We solve the bi-criterion problem by a branch-and-bound algorithm that outputs a set of feasible solutions. This set of solutions has the property that no other feasible solution can have a better value for any one of the two criteria than some solution in the set. The algorithm has been adopted by the plant and works in satisfactory manner: the production plan for a next week is constructed by solving an instance for each type of abrasives. This gives, for each type, a set of high quality solutions, one of which is chosen by the planner for implementation. When we compared our approach on historic data it turned out that in a significant number of cases, the solution constructed by the planner was not among the set of solutions the algorithm generated. This implies that in such a case we could provide a strictly better solution.

The paper is organised as follows. In following section we mention related literature, give a precise problem description, and perform a computational experiment. The next section describes the branch-and-bound algorithm, and in the following section we report practical experience with the algorithm. Finally, we state the conclusions.

Literature, problem description and an experiment

In this section we first discuss the differences with the classical cutting stock problem and mention related literature, then we give a precise problem description and finally we report on a computational experiment.

Related literature

The one-dimensional cutting stock problem is a classical problem in Operations Research. In its pure form it can be described as follows. Given a set of orders, each order described by a width and a number of finals requested exactly, how to cut all finals from raws with a given width using a minimum number of raws?

The problem is already discussed in Kantorovich,[1] and Gilmore and Gomory[2,3] described their famous column generation procedure in two celebrated papers. Recent approaches can be found in Degraeve and Peeters.[4] A classification of literature devoted to cutting and packing problems can be found in the book of Dyckhoff and Finke.[5]

From the discussion described in the introduction, one can deduce a number of characteristics that determine the precise cutting stock problem we deal with in this paper.

- We have to deal with a limited number of finals that can be cut from single raw. This is caused by the fact that there is a limited number of cutting knives available. In fact, this is a standard requirement: it is already discussed in Gilmore and Gomory[3] (see also Chapter 13 in Chvátal[6]).
- There are two types of orders. In some cutting stock problems it is customary to allow a percentage of deviation from the required amount, that is, for each order one is allowed to produce any amount between, say 90% and 110% of the required amount.[7] In our case we have a very different type of lower and upper bounds, and our algorithm will make use of this property.
- In our case the number of raws is given. As discussed in the previous section, this is an important aspect. It changes the problem from a bin packing/cutting stock type of problem to a problem that can be formulated as a so-called generalised assignment type of problem (GAP).
- We have to deal with an objective consisting of two criteria: minimizing total cutting loss and minimizing the number of different patterns. This objective received some attention in literature, see Chu and Antonio[8] and Diegel *et al.*[9] However, one often takes this into account

by summing both criteria using appropriate weight factors.[10] As explained before, we will not follow this approach but instead will enumerate a set of high quality solutions. In this way, the decision maker is not forced to decide explicitly beforehand how to weigh each criterium, but is instead allowed to choose in any way he sees fit between a number of alternatives.

The input

An instance of the problem that occurs for each type of abrasives is specified using the following parameters:

- n is the number of orders, so let us define $N \equiv \{1, 2, \ldots, n\}$,
- $S \subseteq N$ is the set of exact orders,
- $R \subseteq N$ is the set of open orders (with $R \cup S = N, R \cap S = \emptyset$),
- W is the width of a raw,
- C is the maximum number of finals that can be cut from a raw,
- K is the number of available raws,
  for $i = 1, 2, \ldots, n$:
- w_i is the width of a final of order i,
- d_i is the number of finals in order i (demand).

As explained before a *pattern* specifies how a raw is cut, that is, it specifies the number of finals of the given widths that will be produced from a raw. To describe a solution we must specify a pattern for each of the K raws in the instance. The *frequency* of a pattern is the number of raws that are cut according to that pattern.

We are interested in finding each feasible solution that is not dominated by another feasible solution with respect to the both criteria: minimising total cutting loss, and minimising the number of different patterns. Let us here more formally express this notion, where $g_1(\sigma)$ denotes the cutting loss of solution σ and $g_2(\sigma)$ denotes the number of different patterns in σ.

Definition 1 A feasible solution σ is called *Pareto-optimal* if there is no feasible solution π such $g_1(\pi) \leqslant g_1(\sigma)$, and $g_2(\pi) \leqslant g_2(\sigma)$, where at least one of the inequalities is strict.

Thus, we will describe an algorithm that finds all Pareto-optimal solutions.

To illustrate all this, consider as an example the following input (which reflects an instance we encountered in practice).

Example 1 We have $n = 5$, $S = \{1, 2, 4\}$, $R = \{3, 5\}$, $W = 1380$, $C = 36$, and $K = 67$; widths and demands of the orders are given in Table 1. Now we have completely specified an instance.

A feasible solution to this instance is described in Table 2. The column 'Loss' gives the cutting loss with respect to each pattern, and the column 'Freq.' gives the corresponding frequencies. The cutting loss associated with this solution equals 170, and the solution uses four different patterns.

Table 1 Width and demand of the orders

Order	*1*	*2*	*3*	*4*	*5*
Width w	305	200	115	110	95
Demand d	15	135	470	25	40

Table 2 A solution

	No. of finals order 1	*No. of finals order 2*	*No. of finals order 3*	*No. of finals order 4*	*No. of finals order 5*	*Loss*	*Freq.*
Pattern 1	0	0	3	5	5	10	2
Pattern 2	1	1	5	1	2	0	15
Pattern 3	0	0	12	0	0	0	20
Pattern 4	0	4	5	0	0	5	30
Total	15	135	471	25	40	170	67

Table 3 Another solution

	No. of finals order 1	*No. of finals order 2*	*No. of finals order 3*	*No. of finals order 4*	*No. of finals order 5*	*Loss*	*Freq.*
Pattern 1	0	5	0	0	4	0	4
Pattern 2	3	0	4	0	0	5	5
Pattern 3	0	4	5	0	0	5	10
Pattern 4	0	0	12	0	0	0	23
Pattern 5	0	3	5	1	1	0	25
Total	15	135	471	25	41	75	67

Table 3 contains another solution; it has a cutting loss of 75, and uses five different patterns.

In fact, any feasible solution to the instance of Example 1 is dominated by at least one of the solutions given; in other words, these solutions are all Pareto-optimal solutions.

An experiment

A very straightforward approach to the problem described above would be the following: write down an integer linear programming (ILP) formulation, and next try to solve it using a standard IP solver. In fact, that is what we did in some preliminary experiments, with the modification that we only considered cutting loss as an objective function. Using integer decision variables t_{ij} = the number of finals of order i in raw j, a straightforward model looks as follows:

$$\text{(ILP)} \min \sum_{j=1}^{K}\left(W - \sum_{i=1}^{n} w_i t_{ij}\right)$$

subject to

$$\sum_{j=1}^{K} t_{ij} = d_i \qquad \text{for } i \in S; \tag{1}$$

$$\sum_{j=1}^{K} t_{ij} \geqslant d_i \qquad \text{for } i \in R; \tag{2}$$

$$\sum_{i=1}^{n} w_i t_{ij} \leqslant W \qquad \text{for } j = 1, \ldots, K; \tag{3}$$

$$\sum_{i=1}^{n} t_{ij} \leqslant C \qquad \text{for } j = 1, \ldots, K; \tag{4}$$

$$t_{ij} \text{ integer} \qquad \text{for } i \in N, j = 1, \ldots, K. \tag{5}$$

Using this model to solve instances of the problem can be very time-consuming: consider the—not atypical—instance described in Example 1. Cplex (version 6.5), using default settings was stopped after 1 hour of computing time on a Pentium without having found a feasible solution. The size of the search tree had grown to more than 400 Mb. And notice that this approach completely neglects the second criterion. Of course, this experiment does not imply that finding a feasible solution to our problem is a time-consuming task, however, it lends credibility to the claim that linear programming based approaches may not be among the most suited approaches to solve instances of this bi-criterion problem. This experiment motivated us to develop another approach.

The algorithm

This section develops a branch-and-bound algorithm for the problem described in the previous section which outputs all Pareto-optimal solutions. Let us first introduce some additional notation that we need to describe a solution.

- m denotes the number of different patterns, and let $M \equiv \{1, 2, \ldots, m\}$,
- f_j is the frequency of pattern j for $j = 1, 2, \ldots, m$,
- $X = (x_{ji})$ is the number of finals of order i occurring in pattern j, $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$.

Now a solution is given by an integer m, a positive integer m-vector $f = (f_1, f_2, \ldots, f_m)$, and a nonnegative integer $m \times n$-matrix $X = (x_{ji})$. A solution (f, X) is feasible if and only if it satisfies the following constraints

$$\sum_{j=1}^{m} f_j = K; \tag{6}$$

$$\sum_{j=1}^{m} f_j x_{ji} = d_i \qquad \text{for } i \in S; \tag{7}$$

$$\sum_{j=1}^{m} f_j x_{ji} \geqslant d_i \qquad \text{for } i \in R; \tag{8}$$

$$\sum_{i=1}^{n} w_i x_{ji} \leqslant W \qquad \text{for } j \in M; \tag{9}$$

$$\sum_{i=1}^{n} x_{ji} \leqslant C \qquad \text{for } j \in M. \tag{10}$$

Constraint (6) specifies that exactly K raws are used. Constraints (7) and (8) are the *demand constraints* that specify that for an exact order precisely its demand and for an open order at least its demand must be produced. Constraints (9) are the *width constraints* that specify that the total width of the finals produced from one raw must not exceed the width of the raw. Constraints (10) are the *cardinality constraints* that specify that the total number of finals produced from one raw must not exceed the given upper bound.

We follow the following solution approach (Figure 1). Firstly, we compute a lower bound for the minimum total cutting loss that is valid for any number of different patterns. Then, starting with $m = 1$, we do the following. We generate all possible frequency vectors f, that is, all elements of the set $F \equiv \{(f_1, f_2, \ldots, f_m) | 0 < f_1 \leqslant f_2 \leqslant \cdots \leqslant f_m, \sum_{j=1}^{m} f_j = K\}$. (Indeed, notice that

```
The Algorithm:
Step 1: Compute a lower bound on the minimum total cutting loss, say $lb$.
Step 2: $m := 1$, $ub := \infty$.
Step 3: while $m \leq K$ and $ub > lb$ do
    begin
        compute $F$;
        for each $f \in F$ do compute mincutloss$_f$ (by branch-and-bound));
        mincutloss[$m$]:=min$_{f \in F}$ mincutloss$_f$;
        if mincutloss[$m$] $< ub$ then
        begin
            output this solution;
            $ub$ :=mincutloss[$m$];
        end;
        $m := m + 1$;
    end.
```

Figure 1 The algorithm.

without loss of generality we can assume that $f_1 \leqslant f_2 \leqslant \cdots \leqslant f_m$). Then, we find for each of these frequency vectors f a feasible solution (f, X) of minimum total cutting loss (denoted by 'mincutloss$_f$' in Figure 1). This yields a solution with minimum total cutting loss for a specific value of m (denoted by 'mincutloss[m]' in Figure 1). If this minimum total cutting loss is less than the best known minimum cutting loss (ub in Figure 1), it is a Pareto-optimal solution and outputted by the algorithm. Otherwise, m is raised to $m+1$ and the algorithm continues. It should be clear that this approach yields all Pareto-optimal solutions.

Therefore, the algorithm consists of three basic ingredients: firstly, it computes a lower bound, secondly it computes—by simple enumeration—all frequency vectors (the elements of the set F), and finally, for each element of the set F, a branch-and-bound approach is applied to find the minimum cutting loss associated to this frequency vector. Notice that, in the above, a frequency vector is computed and used as input for the branch-and-bound part. Therefore, in the following we demonstrate how to compute a solution with minimum total cutting loss for model (6)–(10) given a vector f. More explicitly, the model we solve is given by the following constraints:

$$\sum_{j=1}^{m} f_j x_{ji} = d_i \qquad \text{for } i \in S; \tag{11}$$

$$\sum_{j=1}^{m} f_j x_{ji} \geqslant d_i \qquad \text{for } i \in R; \tag{12}$$

$$\sum_{i=1}^{n} w_i x_{ji} \leqslant W \qquad \text{for } j \in M; \tag{13}$$

$$\sum_{i=1}^{n} x_{ji} \leqslant C \qquad \text{for } j \in M. \tag{14}$$

Summarising, we determine for each number of patterns what the minimum cutting loss is, and whether this corresponds to a Pareto-optimal solution. This process continues until either $m = K$ or a solution is found with a cutting loss that is minimum for any number of patterns. All Pareto-optimal solutions are presented to the user. The remainder of this section explains in detail how the branching and the bounding of the algorithm work. Finally we illustrate the algorithm on a small example which features all the essentials.

The branching

The root node corresponds to all nonnegative integer solutions X satisfying the constraints (11)–(14).

A node of the branch and bound tree not corresponding to the root node is defined by a subset $S^* \subseteq S$, a subset $R^* \subseteq R$, and vectors $(x^*_{1i}, x^*_{2i}, \ldots, x^*_{mi})$, $i \in S^* \cup R^*$ satisfying the following four constraints:

$$\sum_{j=1}^{m} f_j x^*_{ji} = di \qquad \text{for } i \in S^*; \tag{15}$$

$$\sum_{j=1}^{m} f_j x^*_{ji} \geqslant di \qquad \text{for } i \in R^*; \tag{16}$$

$$\sum_{i \in S^* \cup D^*} w_i x^*_{ji} \leqslant W \qquad \text{for } j \in M; \tag{17}$$

$$\sum_{i \in S^* \cup D^*} x^*_{ji} \leqslant C \qquad \text{for } j \in M. \tag{18}$$

The node represents all nonnegative integer solutions X that satisfy the constraints (11)–(14) given above, and the constraints given below

$$x_{ji} = x^*_{ji} \qquad \text{for } j \in M, i \in S^*; \tag{19}$$

$$x_{ji} \geqslant x^*_{ji} \qquad \text{for } j \in M, i \in R^*. \tag{20}$$

to branch we select an order $i \in S \setminus S^*$ or an order $i \in R \setminus R^*$.

Let us first discuss the branching when an order $i \in S \setminus S^*$ is selected. In that case we generate the set F_i of all solutions $(x^*_{1i}, x^*_{2i}, \ldots, x^*_{mi})$ that satisfy constraint (11) and we add a branch for each solution for which constraints (17) and (18) are satisfied whenever i is added to S^*. The subsets of feasible solutions corresponding to the generated nodes form a partition of the set of feasible solutions corresponding to the node for which the branching was defined.

Let us now discuss the case that an order $i \in R \setminus R^*$ is selected. In that case we enumerate all solutions $(x^*_{1i}, x^*_{2i}, \ldots, x^*_{mi})$ that satisfy constraint (12) and are minimal, that is, it is not possible to reduce one of the positive x^*_{ji} without violating constraint (12). A solution is minimal if and only if it satisfies constraint (16) and the following constraint:

$$\sum_{j=1}^{m} f_j x^*_{ji} < d_i + \min\{f_j | x^*_{ji} > 0\}. \tag{21}$$

We call constraint (21) the *minimality* constraint. We add a branch for each solution for which constraints (17) and (18) are satisfied whenever i is added to R^*. The union of the subsets of feasible solutions corresponding to the generated nodes is equal to the set of feasible solutions corresponding to the node for which the branching was defined. To see this assume we have a solution X that satisfies (11)–(14) and (19)–(20). If it also satisfies the minimality constraint (21), then it is one of the solutions generated and is an element of the subset corresponding to the node generated for this solution. If it does not satisfy (21), then if j^* is such that $f_{j^*} = \min\{f_j | x_{ji} > 0\}$, then we can reduce x_{j^*i} by one without violating constraint (12). Repeating this process we can construct a solution $(x^*_{1i}, x^*_{2i}, \ldots, x^*_{mi})$ that satisfies (16) and (21) and for which $x^*_{ji} \leqslant x_{ji}, j = 1, \ldots, m$.

The question remains how to generate the set F_i of all solutions, for an open order i with width $d_i > 0$, that satisfies constraints (16) and (21). If we realize that $f_1 \leqslant f_2 \leqslant \cdots \leqslant f_m$, then F_i is partitioned into the subsets $F_i^k, k = 1, \ldots, m$, where $F_i^k = \{(x^*_{1i}, x^*_{2i}, \ldots, x^*_{mi}) | x^*_{ji} = 0, j = 1, \ldots, k-1, x^*_{ki} > 0, \sum_{j=k}^m f_j x^*_{ji} \in \{d_i, d_i+1, \ldots, d_i + f_k - 1\}\}$. We generate F_i for an open order i by enumerating the solutions in each subset $F_i^k, k = 1, \ldots, m$.

We use a fixed branching order of the orders. For each order i we generate the corresponding set F_i of all feasible branches with respect to the root node. The orders are selected for branching in nondecreasing order of the cardinality of this set of feasible branches. We use a depth first search strategy in which the nodes with a common predecessor are visited in nondecreasing order of their lower bound.

The bounding

In a node of the branch and bound tree as defined in the previous section we solve the following optimisation problem.

$$\min \sum_{j=1}^m f_j\left(W - \sum_{i=1}^n w_i x_{ji}\right) \tag{22}$$

subject to

$$\sum_{j=1}^m f_j x_{ji} = d_i \qquad \text{for } i \in S^*; \tag{23}$$

$$\sum_{j=1}^m f_j x_{ji} \geqslant d_i \qquad \text{for } i \in R^*; \tag{24}$$

$$\sum_{j=1}^m f_j x_{ji} \geqslant 0 \qquad \text{for } i \in (S \setminus S^*) \cup (R \setminus R^*); \tag{25}$$

$$\sum_{i=1}^n w_i x_{ji} \leqslant W \qquad \text{for } j \in M; \tag{26}$$

$$\sum_{i=1}^n x_{ji} \leqslant C \qquad \text{for } j \in M; \tag{27}$$

$$x_{ji} = x^*_{ji} \qquad \text{for } j \in M, i \in S^*; \tag{28}$$

$$x_{ji} \geqslant x^*_{ji} \qquad \text{for } j \in M, i \in R^*; \tag{29}$$

$$x_{ji} \geqslant 0 \text{ and integer} \qquad \text{for } j \in M, i = 1, \ldots, n. \tag{30}$$

This is a relaxation of the optimisation problem we have to solve in the corresponding node because we have relaxed the constraints (11) for $i \in S \setminus S^*$, and the constraints (12) for $i \in R \setminus R^*$ by replacing them by the constraints $\sum_{j=1}^M f_j x_{ji} \geqslant 0$.

Constraints (23) are redundant because constraints (15) hold and together with constraints (28) they imply (23). Constraints (24) are redundant because constraints (16) hold and together with constraints (29) they imply (24). Constraints (28) are redundant because they are implied by the nonnegativity constraints (30).

If we define the variables $y_{ji} = x_{ji}, j \in M, i \in (S \setminus S^*) \cup (R \setminus R^*)$ and $y_{ji} = x_{ji} - x^*_{ji}, j \in M, i \in R^*$, then the minimisation problem is equivalent to the following maximisation problem.

$$\max \sum_{j=1}^n f_j\left(\sum_{i \in (S \setminus S^*) \cup R} w_i y_{ji}\right)$$

subject to

$$\sum_{i \in (S \setminus S^*) \cup R} w_i y_{ji} \leqslant W - \sum_{i \in S^* \cup R^*} w_i x^*_{ji} \qquad \text{for } j \in M;$$

$$\sum_{i \in (S \setminus S^*) \cup R} y_{ji} \leqslant C - \sum_{i \in S^* \cup R^*} x^*_{ji} \qquad \text{for } j \in M;$$

$$y_{ji} \geqslant 0 \text{ and integer} \qquad \text{for } j \in M, i \in (S \setminus S^*) \cup R.$$

Note that this maximisation problem decomposes into m independent optimisation problems, one for each pattern, of the following type.

(P1) $$\max \sum_{i \in N \setminus S^*} w_i y_i$$

subject to

$$\sum_{i \in N \setminus S^*} w_i y_i \leqslant W^*;$$

$$\sum_{i \in N \setminus S^*} y_i \leqslant C^*;$$

$$y_i \geqslant 0 \text{ and integer} \qquad \text{for } i \in N \setminus S^*,$$

where $0 \leqslant W^* \leqslant W$ and $0 \leqslant C^* \leqslant C$.

Since a problem of type (P1) has to be solved m times in each node of the branch and bound tree we will do some preprocessing which allows us to solve this optimisation problem by a simple table lookup.

In order to define the preprocessing phase consider for a given value $w, 0 \leqslant w \leqslant W$, the following optimisation problem.

(P2) $$\min \sum_{i \in N \setminus S^*} y_i$$

subject to

$$\sum_{i \in N \setminus S^*} w_i y_i = w;$$

$$y_i \geqslant 0 \text{ and integer} \qquad \text{for } i \in N \setminus S^*.$$

For each value of $w, 0 \leqslant w \leqslant W$, we store whether or not this problem is feasible, and if it is feasible we store its optimal value and optimal solution.

In order to solve (P1) we look for the largest value $w, w \leqslant W^*$ for which problem (P2) has an optimal solution with an optimal value less than or equal to C^*. The optimal value of (P1) is given by w and the optimal solution of (P1) is the optimal solution of (P2) for the given w.

We solve problem (P2) for all possible values of $w, 0 \leqslant w \leqslant W$, and for all possible sets S^* that can occur in our branch and bound algorithm by dynamic programming in the following way.

Let us assume without loss of generality that $R = \{1, \ldots, s\}$ and $S = \{s+1, \ldots, n\}$. We can also assume without loss of generality that the exact orders are indexed in the reverse order in which they are selected in the branching, that is, exact orders are selected in the branching in the order $n, n-1, \ldots, s+1$.

Let us define the optimisation problem $P(k, w)$, $1 \leqslant k \leqslant n, 0 \leqslant w \leqslant W$, by

$$P(k, w) \qquad \min \sum_{i=1}^{k} y_i$$

subject to

$$\sum_{i=1}^{k} w_i y_i = w;$$

$$y_i \geqslant 0 \text{ and integer} \qquad \text{for } i = 1, \ldots, k.$$

We define $A[k][w]$ to be equal to -1 if problem $P(k, w)$ is infeasible, and equal to the optimal value of $P(k, w)$ if this problem is feasible. Then the following relation holds.

$$A[1][w] = \begin{cases} -1 & \text{if } w \bmod w_1 \neq 0 \\ w/w_1 & \text{if } w \bmod w_1 = 0, \end{cases}$$

for $k > 1$

$$A[k][w] = \begin{cases} -1 \text{ if} A[k-1][w - qw_k] = -1 \\ \qquad \text{for all } q, 0 \leqslant q \leqslant w/w_k \\ \min\{A[k-1][w - qw_k] + q | q = 0, \ldots, w/w_k\} \\ \qquad \text{otherwise,} \end{cases}$$

where we use integer division.

Since we are not only interested in the optimal value of problem $P(k, w)$ but also in its optimal solution we store for each value of $A[k][w]$ which is nonnegative its optimal solution as a k-vector. If $A[k][w]$ is nonnegative and equal to $A[k-1][w - qw_k] + q$ for some value of q, $0 \leqslant q \leqslant w/w_k$, then an optimal solution of $P(k, w)$ is the k-vector for which the first $k-1$ components are equal to the optimal solution of $P(k-1, w - qw_k)$ and the kth component is equal to q.

An optimal solution of (P2) for a given value of w exists if and only if $A[s][w] \neq -1$ and in that case the optimal solution of (P2) is equal to the optimal solution of problem $P(s, w)$.

We use A and the solutions that correspond to each nonnegative cell of A to solve problem (P2) for a given set S^* as follows. If S^* is empty, then the optimal value of (P2) for a given value of w is given by $A[n][w]$. If S^* is not empty, then by definition $S^* = \{k+1, \ldots, n\}$ for some $k, s \leqslant k \leqslant n-1$. Hence $N \setminus S^* = \{1, \ldots, k\}$ and therefore the optimal solution of problem (P2) for a given value of w is given by $A[k][w]$.

Remark The dynamic program described here to solve problem (P2) is also used to compute a lower bound for the minimum total cutting loss for any number of patterns. Indeed, by computing the 'free width' denoted by $Z := K \cdot W - \sum_{i=1}^{n} d_i w_i$ and solving (P2) for each value of w, $0 \leqslant w \leqslant Z$, we find the largest value of $w \leqslant Z$, say w^*, for which problem (P2) has a feasible solution. The lower bound is then equal to $Z - w^*$.

An example

Consider the following instance. We have $n = 3$, $S = \{1\}, R = \{2, 3\}, W = 27, C = 6$, and $K = 6$; widths and demands of the orders are given in Table 4.

The algorithm starts by checking whether a solution using only a single pattern exists (the case $m = 1$). Since $d_1 = 8$ and hence not a multiple of $K = 6$, no solution with $m = 1$ exists. Set $m = 2$, and compute the set F of all possible pairs of frequencies, that is $F = \{(f_1, f_2) | 0 < f_1 \leqslant f_2, f_1 + f_2 = 6\} = \{(1, 5), (2, 4), (3, 3)\}$. For each of these three frequency vectors, the algorithm computes a solution with minimum total cutting loss as follows. Start with frequency vector (1, 5). Let us compute F_1, the set of pairs that indicate how the 8 requested number of finals of order 1 are distributed over two patterns with multiplicities 1 and 5:

$$F_1 = \{(x_{11}, x_{12}) \in \mathbf{Z}^2 | x_{11} + 5x_{12} = 8, 0 \leqslant x_{11}, x_{12} \leqslant 2\}$$

Observe that the first constraint follows from the value of the frequency vector (constraint (11) for $i = 1$), while the upper bound constraints follow from the width constraint. Since $F_1 = \emptyset$, no feasible solution exists in case $(f_1, f_2) = (1, 5)$. Let us proceed with the second element of f, $(2, 4)$. We now have:

$$F_1 = \{(x_{11}, x_{12}) \in \mathbf{Z}^2 | 2x_{11} + 4x_{12} = 8, 0 \leqslant x_{11}, x_{12} \leqslant 2\} \\ = \{(2, 1), (0, 2)\}.$$

In the same fashion:

$$F_2 = (x_{21}, x_{22}) \in \mathbf{Z}^2 | 2x_{21} + 4x_{22} \geqslant 5, 0 \leqslant x_{21}, x_{22} \leqslant 6, \\ \text{for } x_{21} > 0: 2(x_{21} - 1) + 4x_{22} < 5, \\ \text{for } x_{22} > 0: 2x_{21} + 4(x_{22} - 1) < 5\}.$$

Observe that the upper bound constraints follow from the width constraint. Since order 2 is an open order, the minimality constraints (21) have been added. By the process described earlier we conclude that $F_2 = \{(3, 0), (1, 1), (0, 2)\}$. Similarly:

$$F_3 = (x_{31}, x_{32}) \in \mathbf{Z}^2 | 2x_{31} + 4x_{32} \geqslant 8, 0 \leqslant x_{31}, x_{32} \leqslant 5, \\ \text{for } x_{31} > 0: 2(x_{31} - 1) + 4x_{32} < 8, \\ \text{for } x_{32} > 0: 2x_{31} + 4(x_{32} - 1) < 8\}.$$

We conclude that $F_3 = \{(4, 0), (2, 1), (0, 2)\}$.

Table 4 An instance

Order	*1*	*2*	*3*
Width w	10	3	5
Demand d	8	5	8

Table 5 Information to derive lower bounds

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*
{3}					1					2					3					4					5		
{2, 3}			1		1	2		2	3	2	3	4	3	4	3	4	5	4	5	4	5	6	5	6	5	6	7
{1, 2, 3}			1		1	2		2	3	1	3	4	2	4	2	3	5	3	4	2	4	5	3	3	3	4	6

The number of levels in the branch-and-bound tree equals the number of orders, being three here. The branching sequence is determined by the cardinality of the sets F_i, in this case the branching sequence is 1, 2 and 3.

In the tree we make extensive use of Table 5 to get lower bounds. We have already described how this table can be computed efficiently.

To explain Table 5: if there is an entry in this table in row a and column b it means that with the set of finals in row a, an integer combination can be constructed with a total width of b. The value of the entry denotes the minimum number of finals needed to accomplish this width. In fact, in our algorithm we not only stored this number but also the corresponding solution. For instance, consider column 19 and row {2, 3}. The entry 5 indicates that one can precisely fill a (remaining) width of 19 using 5 finals of orders 2 and 3. The corresponding solution, not shown in the table, is (0, 3, 2).

Let us now sketch the branch-and-bound tree. At the first level there is a branch for each element of F_1 (Figure 2).

In each of the leaf nodes of Figure 2 we use Table 5 to compute a lower bound on the minimum cutting loss in the following manner. Consider the left-branch in Figure 2. The remaining width of the first pattern is $27-(2\times 10)=7$, and of the second pattern is $27-10=17$. Moreover, the remaining number of finals that can be cut in the first pattern is 4, and in the second pattern is 5. Row {2, 3} of Table 5 tells us that from a remaining width of 7 at most 6 can be used with 2 additional finals, implying a lower bound of 1 for this pattern. Similarly, we deduce that a remaining width of 17 can be filled completely using 5 additional finals. Therefore, for the second pattern we have a trivial lower bound of 0. Concluding, we can associate a lower bound of $(2, 4)\cdot(1, 0)=2$ to the left branch. Doing similar computations for the right branch in Figure 2 gives a lower bound of $(2, 4)\cdot(1, 1)=6$. We proceed to expand the branch with a minimum lower bound. Therefore, the tree develops as follows.

Computing the lower bounds corresponding to each of the three leaf nodes in the tree in Figure 3 yields the

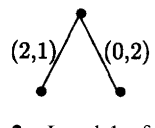

Figure 2 Level 1 of the tree.

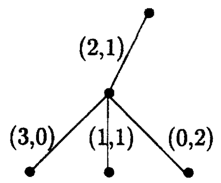

Figure 3 A part of level 2 of the tree.

following: for the left node we have a violation of the width constraint of pattern 1 (since $(2\times 10)+(3\times 3)>27$), for the other two nodes we compute (since finals 2 and 3 are open orders we again use row {2, 3} of Table 5) lower bounds of 2 and 2 respectively.

To further expand, we arbitrarily select the middle leaf node which yields the tree depicted in Figure 4.

The left leaf node and the middle leaf node each correspond to infeasible solutions due to violation of the width constraint. For the right leaf node we compute, using Table 5, a lower bound of 6: indeed, the remaining width for pattern 1 is 4 which can be filled by up to 1 by (0, 1, 0) and the remaining width for pattern 2 is 4 as well, which also can be filled up to 1 by (0, 1, 0). Therefore the solution we arrive at consist of the patterns described in Table 6.

Now we backtrack to deal with nodes that are still open. First we expand the right leaf node in Figure 3, giving us Figure 5.

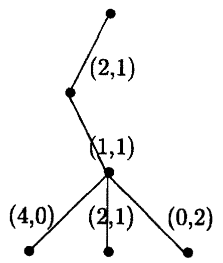

Figure 4 A part of level 3 of the tree.

Table 6 A solution

	No. of finals order 1	*No. of finals order 2*	*No. of finals order 3*	*Freq.*
Pattern 1	2	2	0	2
Pattern 2	1	2	2	4

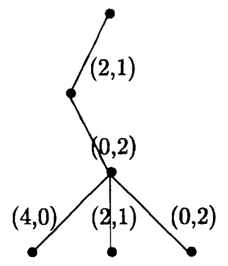

Figure 5 Another part of level 3 of the tree.

Again, the left leaf node and the middle leaf node each correspond to infeasible solutions due to violation of the width constraint, and the right leaf node gives a lower bound of 6. We conclude that no better solution can be found here. Similarly, we can fathom the right leaf node in Figure 2, thereby establishing that the solution described above has minimum cutting loss for $m = 2$.

In a similar fashion one can deal with the cases $m \geqslant 3$. In fact, it turns out that a cutting loss of 6 is minimal for any number of patterns. Therefore, the set of Pareto-optimal solutions is the single solution given in Table 6.

Practical experience with the algorithm

In this section we report some of the issues that are specific for the setting that motivated this study and that influence the performance of the algorithm. Instances that we encountered have the following characteristics.

- Any width of a final is less than $\frac{1}{3}$ of the width of a raw. This property implies that a trivial lower bound on the number of raws needed almost always admits a feasible solution. More specifically, given a set of orders for some type of abrasives, one easily verifies that the following expression yields a lower bound for the number of raws needed to produce all orders:

$$lbraw = \max\left\{\left\lceil\left(\sum_{i=1}^{n} d_i w_i\right)/W\right\rceil, \left\lceil\left(\sum_{i=1}^{n} d_i\right)/C\right\rceil\right\}.$$

(For instance, in Example 1 (see Table 1),

$$lbraw = \max\left\{\left\lceil\frac{92175}{1380}\right\rceil, \left\lceil\frac{685}{36}\right\rceil\right\} = 67.)$$

For all instances that we encountered in practice there was always a feasible solution using *lbraw* raws. This property motivates our choice to treat the number of raws as given. In fact, in our implementation the initial value of K is set equal to *lbraw*. If no feasible solution exists for the given number of raws, then the number of raws is increased by one (this has never happened in practice). If a feasible solution exists for a given value of K, then the user of the program can also set the value of K equal to any value which is greater than the given initial value.

- The number of raws required varies between 2 and 80 (but most instances require less than 10 raws), and the number of different widths varies between 2 and 8 (but most instances have less than 5 different widths). This characteristic is of eminent importance for the running time of the algorithm. Almost all instances that we encountered are solved within a few seconds, and this is due to the fact that these instances admit solutions where only a small number of patterns is sufficient to yield an overall minimum total cutting loss. And this is caused by the fact that almost all the instances have a relatively small number of different widths. In occasions (that are rare in practice) where the number of raws is large and the number of different widths is large, our algorithm can take a lot of time. Then generating all solutions satisfying (1) (the sets F_i) may take a lot of time. Summarising this discussion: our empirical evidence suggests that when the input has few different widths, then the output consists of a limited number of patterns. Our approach makes use of this phenomenon.

The only theoretical result that we are aware of concerning this issue is described in ref. 11: in our context, it says that when $n = 2, S = N$ (that is the case with 2 open orders, no exact orders), a solution with minimum total cutting loss can be described using at most 3 different patterns, for any values of the widths, demands and K.
- About half the orders are open orders.

The algorithm that currently is implemented at the plant has the option to stop computations when no improvement in cutting loss is found for 3 consecutive values of m. This feature was added to be able to prevent occasional large running times when in fact satisfactory solutions had already been obtained.

Conclusion

We described a real-life cutting stock problem where two criteria play a role: minimising total cutting loss and minimising the number of patterns used. We presented a branch-and-bound algorithm that outputs all Pareto-optimal solutions. In this way the decision maker can decide which solution fits his/her preferences best without making explicit choices concerning weights of the two criteria. The algorithm has been implemented and is currently in use.

Acknowledgements—We would like to thank T Hintzen and J Oprey of GELVA BV for their willingness to provide us with data. Also, we would like to thank the referees and the editor for their comments which improved the presentation of this paper.

References

1 Kantorovich LV (1960). Mathematical methods of organizing and planning production. *Man Sci* **6**: 366–422.

2 Gilmore P and Gomory RE (1961). A linear programming approach to the cutting-stock problem part I. *Opns Res* **9**: 849–859.

3 Gilmore P and Gomory RE (1963). A linear programming approach to the cutting-stock problem part II. *Opns Res* **11**: 863–888.

4 Degraeve Z and Peeters M (1998). Benchmark results for the cutting stock and bin packing problem. Report 9820, Catholic University Leuven, Belgium.

5 Dychoff H and Finke U (1992). *Cutting and Packing in Production and Distribution*. Physica-Verlag: Heidelberg.

6 Chvátal V (1980). *Linear Programming*. W.H. Freeman and Company: New York.

7 Haessler RW and Sweeney PE (1991). Cutting stock problems and solution procedures. *Eur J Opl Res* **54**: 141–150.

8 Chengbin C and Antonio J (1999). Approximation algorithms to solve real-life multicriteria cutting stock problems. *Opns Res* **47**: 495–508.

9 Diegel A *et al* (1996). Setup minimising conditions in the trim loss problem. *Eur J Opl Res* **95**: 631–640.

10 Valério de Carvalho JM and Guimarães Rodrigues AJ (1995). An LP-based approach to a two-stage cutting stock problem. *Eur J Opl Res* **84**: 580–589.

11 McCormick ST, Smallwood SR and Spieksma FCR (1997). A polynomial algorithm for multiprocessor scheduling with two job lengths. In: *Proceedings of the 8th Symposium On Discrete Algorithms*, ACM-SIAM, USA, pp 509–517.

Received September 1999;
accepted June 2000 after two revisions