A $p$-Median Model for Assortment and Trim Loss Minimization with an Application to the Glass Industry

Claudio Arbib, Fabrizio Marinelli *

Dipartimento di Informatica, Università degli Studi di L’Aquila, via Vetoio, Coppito, I-67010 L’Aquila, Italy

Abstract

One of the main issues in the glass industry is the minimization of the trim loss generated when cutting large parts (stocks) into small items. In our application stocks are produced in the plant. Many distinct stock sizes are feasible, and technical constraints limit the variety of cutting patterns to those producing a single type of item per stock. Consequently, the focus is not on seeking an optimal subset of cutting patterns, but rather on choosing an optimal subset of a limited number of stock sizes. In this paper we discuss a 0-1 linear programming formulation for this problem based on a $p$-median model. Tested on data from the field, the formulation shows an impressive reduction of the trim loss produced in the present plant operation and definitely outperforms traditional exact approaches in terms of computation time.

Key words: Trim Loss Minimization, Assortment Problem, $p$-Median Problem, Integer Programming.

* Corresponding author.

Email addresses: arbib@di.univaq.it (Claudio Arbib), marinelli@di.univaq.it (Fabrizio Marinelli).

21 June 2004
1 Introduction

One main issue of many manufacturers is the minimization of trim loss derived from cutting large parts (stocks) into small items. This problem is generally known as cutting stock [5], and has been widely studied in a large number of variants, according to technical aspects of the production process, constraints and objectives. A very important, and difficult, variant of the problem is when setups are involved. Aim of this paper is to present a new method for solving a class of cutting stock problems with setups. The method, based on a reformulation of the problem in terms of p-median, approximates the optimal solution within a fixed (asymptotically null) ratio, and is much more efficient than the classical exact methods used for similar problems.

In this paper we test the method on a problem of this kind arising in a plant hold by one of the world's most prominent manufacturers of glass. A key phase of the glass production process, where a relevant fraction of the total trim loss is generated, consists of cutting large rectangular stocks into rectangular items of various sizes. In many industries, the minimization of the trim loss derived from such a phase is a 2-dimensional cutting stock problem, where one wants to find the best packing of the required items into stocks of a fixed size. A pack of items into a single stock represents a cutting pattern that can be replicated a number of times and, generally, involves items of different sizes. In our application, however:

(i) stocks are produced in the plant, and a large number of distinct stock sizes are feasible;
(ii) technology/organization constraints limit the variety of cutting patterns to those producing a single type of item per stock.

By (i) and (ii), the focus is therefore on choosing stock sizes rather than cutting patterns. Since stock sizes are not problem data but decision variables, one could in principle choose the ideal stock sizes with zero trim loss that are obtained as integral multiples of each item size. However, due to setup costs and cutting tolerances, one cannot produce all the ideal stock sizes needed to cover the requirement of items in the planning period. Therefore a zero trim loss solution is, in practice, always unfeasible. Furthermore, the following simple example shows that a minimum trim loss solution may contain no ideal stock:

Example 1 Suppose we have to produce $d_1 = 4800$ items of size $57 \times 145$ and $d_2 = 4800$ items of size $60 \times 135$ (centimetres), and setup costs force us to use just one stock size. Suppose also that, due to slitters range and tolerance, only two ideal stock sizes are feasible: $285 \times 580$ for item 1 and $300 \times 540$ for item 2 (each stock yields 20 items). Adopting the former (the latter) stock size
would produce a trim loss of 1.071 (1.216) square metres. However, a stock of size $300 \times 580$, which is not ideal for any of the two items, would only produce a loss of 497 square metres.

The discussion above justifies the interest for a particular assortment problem, where we want to select a limited subset of stock sizes which allow us to produce the required amount of items with a minimum trim loss.

In the remainder of this section we describe plant and process details relevant to the problem addressed (§1.1), and discuss similar problems encountered in the literature (§1.2). The rest of the paper is organized as follows.

In Section 2 we give a formal definition of the problem and survey possible solution approaches: a straightforward formulation as integer linear programming is described in §2.1; a simplifying assumption is proposed in §2.2, and its properties are analysed; based on this assumption, in §2.3 we then introduce and justify a $p$-median model for assortment and trim loss minimization, and embed it into a 0-1 linear programming formulation with side constraints taking into account real process features; a discussion on the complexity of the approaches described and the relevant optimization problems is finally reported in §2.4.

In Section 3 the solution approaches are tested on data from the field. The computational results show the applicability and efficiency of the $p$-median model, which largely outperforms both the policy presently adopted in the plant (with an impressive downsizing of the trim loss) and existing exact methods (with similar solutions obtained in outstandingly shorter time).

1.1 Basic process features and specifications

The production process is articulated into three macro-phases (see Figure 1):

- **Float**: glass is melted in a furnace; a ribbon of flat glass leaves the furnace and flows on a conveyor. Rectangular sheets (stocks) of sizes ranging in $[450, 610]$ (width) and $[280, 321]$ (height, data are expressed in centimeters) are obtained by means of both ribbon width changeovers and vertical cutters. A (constant) cost due to the glass wasted during the setup occurs at every width changeover, whereas height changeovers cause no setup cost. At the end of the conveyor, stocks are stacked and sent to warehouse.
- **Offline Cutting**: stocks taken from the warehouse are cut into smaller rectangular parts (items) according to requirements. There are five cutting machines, each provided with an outbuffer. As production goes ahead, the buffer is filled with items of the same size, until the stack is completed and,
subsequently, the buffer emptied. Since outgoing items cannot be rotated, the items of a cutting pattern are all oriented in the same way.

- *Shaping*: one or more finite parts are obtained from each rectangular item after moulding and bending.

![Diagram](image)

**Fig. 1. Phases of the process.**

Four types of colored glass are produced. Since the color changeover is the most expensive (it can take up to 3 days), the master production schedule is cyclically organized in campaigns ranging from fifteen days to two months, and within each campaign the same color is produced. The float production planning horizon corresponds therefore to the whole production cycle, i.e., to the period between two campaigns of the same color. As previously observed, final items are not directly produced at the float stage, which instead produces stocks, that is intermediate components to be cut again later on. This approach is adopted in order to smooth unplanned orders: in fact, customer orders are completely known only one month in advance, whereas the length of the planning horizon is computed on the basis of an estimate of the requirements.

Although the possibility of combining different stock sizes would help improve the material utilization [7], the assortment of stock sizes in the warehouse should be kept to a certain amount in order to meet stock handling requirements and reduce setups at the downstream cutting machines.
Waste sources are of four types:

- glass imperfections (amounting to 8-9% of the total production);
- trim loss due to width changeovers and offline cutting (4-5%);
- breakage due to handling (around 3%);
- breakage due to offline cutting (less than 1%).

In this paper we focus on the trim loss due to width changeovers and offline cutting. This loss represents about 30% of the total waste and is the only one that can be substantially reduced by mere planning. In our case, trim loss is computed as the difference between the total area of the material used and the total area of the material required, i.e., overproduction is regarded as a loss. In fact, if an item type is overproduced, the overproduction involves the cut of just one stock, and the cost of handling such a few items is greater than the value of the items themselves.

1.2 Related problems

Al-Khayyal et al. [1] describe a similar industrial environment, but address a different problem. In that case the required items are in fact directly cut from the glass ribbon and unloaded by means of spurlines. Unlike our case, different item sizes can be produced by the same cutting pattern. Moreover, since setups occur at the spurlines depending on item sizes, the items have to be conveniently scheduled. The problem is therefore a two-dimension cutting stock coupled with a scheduling problem.

In the literature, a cutting-stock problem in which cutting patterns are applied to get both intermediate components and finished parts is often called multi-stage, see [6], [12]. Although the process addressed in this paper consists of more than one stage, our problem cannot be properly included in such a category: in fact no cutting pattern producing different stock sizes is applied at the float, but different stock widths are simply obtained by narrowing or widening the glass ribbon, and waste at this stage is only generated during width changeovers.

Under some respects, our problem is related to a cutting-stock with multiple stock sizes. Several contributions to this problem can be found in the literature, starting from the seminal work by Gilmore and Gomory [7] and, recently, by Schilling and Georgiadis [9], and Belov and Scheithauer [3]. The problem is almost always modelled through the classical Gilmore and Gomory’s formulation, where the stock size multiplicity is handled by solving a distinct pricing problem for each stock size. Belov and Scheithauer report that it is sufficient to take into account just the distinct stock sizes coming from integer non-negative combinations of the item lengths, and provide on this basis a
dynamic programming algorithm to generate the significant cutting patterns. Indeed, such an algorithm could easily be modified for our purposes. Moreover, in our case assumption (\textit{ii}) makes the pricing problem trivial. However, when Gilmore and Gomory’s formulation is adapted to limit the maximum number of distinct sizes used, its relaxation provides weak bounds and the approach becomes unpractical. (This drawback also occurs when one wants to reduce or minimize the number of distinct cutting patterns used, see [4], [10], [11]).

The problem of choosing the best inventory of stock sizes so as to both limit their assortment and minimize the trim loss is known as the \textit{assortment or the stock-size selection problem}. An integer programming formulation and a heuristic for this problem are described in [8]. Due to assumption (\textit{ii}), this problem is a generalization of ours, which however remains NP-hard, see Section 2.4. A further particularization of the problem, where one can cut just one item per stock, is considered in [2] where a polynomial-time dynamic programming algorithm is proposed.

\section{Problem formulation and solution approaches}

In this paper we discuss the following problem:

\textbf{Problem 2} \textit{What distinct stock sizes should be produced in a campaign to fulfill the demand of rectangular items so as to}

\begin{enumerate}
\item[(i)] limit the number of width changeover to a certain amount \( q \),
\item[(ii)] limit the assortment of distinct stock sizes to a certain amount \( p \),
\item[(iii)] reduce the area of the stocks used as far as possible,
\end{enumerate}

\textit{when a single item size can be cut from each stock size?}

Besides parameters \( p \) and \( q \), an instance of Problem 2 is defined by:

\begin{itemize}
\item \( J \) = the set of distinct feasible stock sizes, \( |J| = n \);
\item \( I \) = the set of distinct item sizes to be produced in the campaign, \( |I| = m \);
\item \( d_i \) = the requirement of the \( i^{th} \) item size, \( i \in I \).
\end{itemize}

where each item or stock \( j \in I \cup J \) is associated with its size, given as a pair of integers \((w_j, h_j)\).
2.1 Formulation as integer linear programming

For $i \in I$ and $j \in J$, let $a_{ij}$ denote the maximum number of items of the $i^{th}$ size one can cut from one stock of the $j^{th}$ size. Parameter $a_{ij}$ corresponds to a saturating cutting pattern. With no loss of generality, exactly one such pattern is associated with each pair $(i, j)$, let $x_{ij}$ denote its activation level. Let $c_j = w_j h_j$ denote the area of stock $j \in J$ (of course, among all the stock sizes producing the same amount of items of type $i$, we will include in the model just the one with minimum area.) The Gilmore-Gomory formulation of the problem of fulfilling the demand of each item minimizing the total trim loss (computed according to Section 1.1) reads as:

$$
\min c(x) = \sum_{j \in J} c_j \sum_{i \in I} x_{ij} \quad (1)
$$

$$
\sum_{j \in J} a_{ij} x_{ij} \geq d_i \quad \text{for } i \in I \quad (2)
$$

$$
x_{ij} \geq 0, \text{ integer } \quad \text{for } i \in I, j \in J \quad (3)
$$

A solution to the above integer program does not necessarily meet the maximum allowance of distinct stock sizes and width changeovers (clauses (i) and (ii) of Problem 2). Additional variables and constraints must therefore be added. Let

- $K = \text{the set of all the distinct feasible stock widths};$
- $J_k \subseteq J = \text{the set of all feasible stock sizes sharing the } k^{th} \text{ width } (k \in K)$.

Let then $y_j$, $z_k$ be 0-1 decision variables defined for $j \in J, k \in K$, with the following meaning:

$$
y_i = \begin{cases} 
1 & \text{if stock size } j \text{ is used at least once} \\
0 & \text{otherwise}
\end{cases}
$$

$$
z_k = \begin{cases} 
1 & \text{if at least one stock size with the } k^{th} \text{ width is used} \\
0 & \text{otherwise}
\end{cases}
$$

A feasible solution to Problem 2 must fulfill the following constraints:

$$
x_{ij} \leq M_{ij} y_j \quad \text{for } i \in I, j \in J \quad (4)
$$

$$
\sum_{j \in J} y_j \leq p \quad (5)
$$
\[ y_j \leq z_k \quad \text{for } j \in J_k, k \in K \] (6)

\[ \sum_{k \in K} z_k \leq q \] (7)

\[ 0 \leq y_j, z_k \leq 1, \quad \text{integer} \] (8)

where \( M_{ij} = \lceil \frac{d_i}{a_{ij}} \rceil \) is an upper bound to the activation level of pattern \((i, j)\).

2.2 A simplifying assumption

Although problem (1)-(3) is NP-hard even with one item size (see Section 2.4), commercial solvers can normally solve practical instances in few seconds. The addition of setups, however, frequently makes such instances intractable: in fact, due to activation constraints (4), the linear relaxation of problem (1)-(8) is generally too weak for an efficient run of a branch-and-bound method. To cope with this inconvenience, let us make the following simplifying assumption

**Assumption 2.1** Each item size \( i \in I \) must be produced by exactly one stock size \( j \in J \).

Indeed, Assumption 2.1 can lead to sub-optimal solutions, as shown in the following example.

**Example 3** Suppose we want produce 13 items of size \( 10 \times 10 \) (centimeters), and stocks A and B of sizes \( c_A = 20 \times 25 \) and \( c_B = 20 \times 35 \) are available. A saturating pattern using stock A (stock B) yields 4 items (6 items). In an optimal solution disregarding Assumption 2.1, one can choose 1 stock of A and 2 stocks of B, with a total usage of 1,900 square centimeters of raw material. However, an optimal solution respecting Assumption 2.1 gives 4 stocks of A, with a total usage of 2,000 square centimeters. \( \square \)

However, let

\[ d = \min_{i \in I} \{d_i\}, \quad \gamma = \min_{i \in I, j \in J} \{c_j/a_{ij}\}, \quad C = \max_{j \in J} \{c_j\}. \]

Then one can prove the following

**Theorem 4** Let \( x^* \) be an optimal solution of Problem 2. Then there exists a feasible solution \( x \) respecting Assumption 2.1 and such that

\[ \frac{c(x) - c(x^*)}{c(x^*)} \leq \frac{C}{\gamma d} \] (9)
Proof. Let $J(i) = \{ j \in J : x^*_{ij} > 0 \}$ be the set of stocks activated by $x^*$ to produce item $i$. For any $i \in I$ denote as $j_i$ a stock size such that

$$
\frac{c_{j_i}}{a_{ij_i}} \leq \frac{c_j}{a_{ij}}
$$

for all $j \in J(i)$. A solution $x$ which covers the requirement of $i \in I$ just producing $d_i$ parts from stock $j_i$ is clearly feasible, as it activates no more distinct stock sizes and width changeovers than $x^*$.

Suppose that $x^*$ covers $i \in I$ by producing $d_{i1}$ parts from stock 1, $d_{i2}$ from stock 2, ..., $d_{in}$ from stock $n$, with $d_{i1} + \ldots + d_{in} = d_i$. The costs of $x, x^*$ clearly verify the following:

$$
c(x) < \sum_{i \in I} c_{j_i} \left( \frac{d_i}{a_{ij_i}} + 1 \right)
$$

$$
c(x^*) \geq \sum_{i \in I} \sum_{j \in J(i)} c_j \frac{d_{ij}}{a_{ij}} \geq \sum_{i \in I} \min_{j \in J(i)} \left\{ c_j \right\} \sum_{j \in J(i)} d_{ij} = \sum_{i \in I} c_{j_i} \frac{d_i}{a_{ij_i}}
$$

where we do not round the ratios and, in the first bound, take the worst case in which the last stock is used to cut just one item. Hence

$$
\frac{c(x) - c(x^*)}{c(x^*)} < \frac{\sum_{i \in I} c_{j_i}}{\sum_{i \in I} (c_{j_i}/a_{ij_i})d_i} \leq \frac{1}{d} \cdot \frac{|I| \max_{j \in J} \{ c_j \}}{|I| \min_{i \in I, j \in J} \{ c_j/a_{ij} \}} = \frac{C}{\gamma d}
$$

\Box

Since $c_j/a_{ij} \geq w_i h_i$ for all $i \in I, J \in J$, one has

$$
\frac{C}{\gamma d} \leq \frac{C/c}{d}
$$

where $c = \min_{i \in I} \{ w_i h_i \}$. Therefore, in such cases of mass production as the plant considered in this paper, Assumption 2.1 has no practical effect on trim loss minimization (see Section 3 for details). From a computational viewpoint, instead, the advantage is quite relevant: Assumption 2.1 transforms in fact integer program (1)-(3) into a partition matroid and, more in general, permits a pure combinatorial (and more efficient) formulation of Problem 2, as we will see in the next section.
2.3 A $p$-median model

Let

- $c_{ij} =$ the stock area used when producing the whole requirement of item $i$ by stock $j$ ($i \in I, j \in J$).

Let then $y_{ij}$ be 0-1 decision variables defined for $i \in I, j \in J$, with the following meaning:

\[
y_{ij} = \begin{cases} 
1 & \text{if stock size } j \text{ is used to produce item size } i \\
0 & \text{otherwise}
\end{cases}
\]

and let $y_j, z_k$ be defined as in Section 2. An approximate solution (in the sense of Theorem 4) to Problem 2 can then be obtained by solving the following 0-1 linear program:

\[
\begin{align*}
\min & \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
\text{s.t.} & \quad \sum_{j \in J} y_{ij} \geq 1 \quad i \in I \quad (11) \\
& \quad y_{ij} \leq y_j \quad i \in I, j \in J \quad (12) \\
& \quad \sum_{j \in J} y_{ij} \leq p \quad i \in I \quad (13) \\
& \quad y_j \leq z_k \quad j \in J, k \in K \quad (14) \\
& \quad \sum_{k \in K} z_k \leq q \quad (15) \\
& \quad 0 \leq y_{ij}, y_j, z_k \leq 1, \text{ integer} \quad (16)
\end{align*}
\]

The objective of formulation (10)-(16) is to minimize the total trim loss.

Constraints (11)-(13) together with clauses (16) describe the feasible region of a special $p$-median problem: in particular, constraints (11) ensure demand satisfaction; constraints (12) activate a stock $j$ whenever it is used by some item $i$; constraint (13) limits the assortment of the stock sizes produced.

The addition of constraints (14), (15) allows to control the number of width changeovers. In particular, constraints (14) activate variables $z_k$, and constraint (15) limits the changeovers to the maximum allowance $q$.

Theorem 4 implies
Corollary 5  Program (10)-(16) is asymptotically exact, that is, its optimal solution tends to the optimum of Problem 2 as the minimum requirement of items increases.

Formulation (10)-(16) can be improved by adding suitable valid inequalities.

Proposition 6  The linear relaxation of program (10)-(16) is enforced by the following inequalities:

\[
\sum_{j \in J_k} y_{ij} \leq z_k \quad i \in I, k \in K
\]  

(17)

Proof. Clearly, every integer solution of (10)-(16) is also a solution of (17). Thus we simply have to show that the linear relaxation of formulation (10)-(16) admits a fractional solution which is cut off by some of the (17). In fact, suppose that stocks 1 and 2 have the same width. Then the point

\[
y_{11} = y_{21} = x_{12} = x_{22} = 0.5, y_1 = y_2 = 0.5, z_1 = 0.5
\]

is a solution to \( y_{11} \leq y_1 \leq z_1 \) and \( y_{12} \leq y_2 \leq z_1 \) (\( i = 1, 2 \)) but does not verify

\[
y_{11} + y_{12} \leq z_1, y_{21} + y_{22} \leq z_2.
\]

\[\square\]

Observe that, although inequalities (14) can be replaced in formulation (10)-(16) by inequalities (17), the latter do not in general dominate the former in the linear relaxation of the problem, as shown by the following example.

Example 7  Again, suppose that stocks 1 and 2 have the same width. Then the point

\[
y_{11} = x_{21} = 0.3 \quad y_{12} = y_{22} = 0.6
\]
\[
y_1 = y_2 = 1.0, z_1 = 0.9
\]

is a solution to

\[
y_{11} + y_{12} \leq z_1, y_{21} + y_{22} \leq z_2
\]

but does not verify \( y_1 \leq z_1 \) and \( y_2 \leq z_1 \). \[\square\]
2.4 A note on complexity

The computational complexity of Problem 2 is immediately stated by observing that program (1)-(3), written for a single item of unit size, allows us to find positive integers \( x_1, \ldots, x_n \) verifying

\[
d \leq c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \leq d'
\]

for any fixed \( 0 \leq d \leq d' \). Thus the problem is NP-hard for \( m = 1 \) and independently on any bound on the number of stock sizes or width changeover, i.e., even if \( p = q = n \). As said in Section 2.3, Assumption 2.1 makes it possible to approximate the solution of Problem 2 via a particular \( p \)-median model. When \( p = q = n \), the model becomes a partition matroid, and can therefore be solved via the greedy algorithm. For arbitrary \( p \) and \( q \) we cannot infer the complexity of the model from that of the general \( p \)-median problem, since the \( c_{ij} \)'s are indeed peculiar (they are in fact obtained as the minimum area from which one can cut all the required items of size \( i \) from stocks of size \( j \)). Nevertheless we can prove the following

**Theorem 8** Problem 2 is NP-hard even under Assumption 2.1.

*Proof.* By reduction from Set Covering: given a collection \( \mathcal{F} \) of distinct subsets of a finite set \( S \) and an integer \( k \), find a subset \( \mathcal{C} \) of \( \mathcal{F} \) such that every element in \( S \) belongs to at least one member of \( \mathcal{C} \), and \( |\mathcal{C}| \leq k \).

In the decision version, an instance of Problem 2 includes also a positive integer \( c \) indicating an upper bound to the area of the stocks used in a solution. Construct an instance where \( p = k \), and items (stocks) are in a one-to-one correspondence with elements of \( S \) (of \( \mathcal{F} \)). Items and stocks have all the same width \( w \), and \( q = 1 \). The height \( h_i \) of item \( i \in I \) equals the \( i \)-th prime number \( \pi_i \). Let \( S_j \subseteq S \) be a member of \( \mathcal{F} \). The height of the corresponding stock \( j \in J \) is given by

\[
h_j = \prod_{i \in S_j} \pi_i
\]

(Notice that item and stock sizes are all different from each other.) To define the requirement \( d_i \) of item \( i \in I \), consider the collection \( \mathcal{F}_i \) of the members of \( \mathcal{F} \) containing the corresponding element of \( S \): \( \mathcal{F}_i = \{ S_j \in \mathcal{F} | i \in S_j \} \). Then

\[
d_i = \text{lcm}\{ h_j | S_j \in \mathcal{F}_i \}
\]

Finally, \( c = w \sum_{i \in I} d_i h_i \), that is, we search for zero trim loss solutions. (Notice that the construction can be carried out in polynomial time and space: in fact \( \pi_i < 2^{2i} \), so the encoding of \( h_k \), \( k \in I \cup J \), and \( d_i \), \( i \in I \), takes less than \( 2m^2(m + n + 1) \) bits.)
A solution of the above instance gives a set $J^* \subseteq J$ of $\leq p$ distinct stock sizes from which one can cut exactly $d_i$ items for each size $i \in I$ with no trim loss. Consider then the collection

$$C = \{S_j \in F| j \in J^*\}$$

Observe that for each $j \in J$ and $i \in I$, $h_i$ divides $h_j$ if and only if $i \in S_j$. Therefore, in order to have a zero trim loss solution, stock $j$ can be used to produce only items in $S_j$. Hence for any $i \in I (i \in S)$ there exists a $j \in J^* \ (an \ S_j \in C)$ such that $i$ is cut from $j$ ($i$ belongs to $S_j$). Thus $C$ is a cover of $S$ with $\leq p = k$ elements.

Conversely, let $C$ be a cover of $S$, and $C_i$ be the collection of all the sets of $C$ containing $i \in S$, $C_i = \{S_j \in C|i \in S_j\} \subseteq F_i$. Let $J^*_i$ (let $J^*_i$) contain all the stock sizes corresponding to the members of $C$ (of $C_i$). Since $h_j/h_i$ is the (integer) number of items of size $h_i$ produced by a single stock of size $h_j$, the activation levels $x_{ij}$ of patterns $j \in J_i$ must verify the following linear diophantine equation:

$$\sum_{j \in J_i^*} h_j x_{ij} = d_i h_i$$

which, being $d_i/h_k$ integer for any $k \in C_i$, admits the solution $x_{ij} = d_i h_i/h_j$ for $j = k$, $x_{ij} = 0$ for $j \neq k$. Such a solution clearly verifies Assumption 2.1. □

Table 1 and Figure 2 resume the dependence of the problem complexity on different assumptions/constraints.

<table>
<thead>
<tr>
<th>≤ p</th>
<th>one item size per stock</th>
<th>one stock size per item size</th>
<th>Problem</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>⌈</td>
<td>–</td>
<td>–</td>
<td>Cutting Stock</td>
<td>Hard</td>
</tr>
<tr>
<td>–</td>
<td>–</td>
<td>×</td>
<td>Problem (1)-(3)</td>
<td>Hard</td>
</tr>
<tr>
<td>–</td>
<td>×</td>
<td>–</td>
<td>Problem (1)-(3) + Assumption 2.1</td>
<td>Easy</td>
</tr>
<tr>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Cutting Stock with Setups</td>
<td>Hard</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>–</td>
<td>Problem 2</td>
<td>Hard</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>Problem 2 + Assumption 2.1 = Problem (10)-(16)</td>
<td>Hard</td>
</tr>
</tbody>
</table>

Table 1
Complexity of the problems considered.
3 Computational experience

A preliminary computational test was done on real data coming from the plant. The test aims to evaluate:

- the size of the optimization models in terms of variables and constraints,
- the waste savings,
- the theoretical bound (9) and the actual approximation,
- the computation time and the amount of branch-and-bound nodes explored.

All the integer programs were solved by Cplex 8.0 with default setting on a Pentium III 400MHz with 120Mb RAM.

Ten real problem instances have been solved. For each instance, Table 2 reports the number \( m \) of item types to be produced, the maximum allowances of assortment \( (p) \) and width changeovers \( (q) \), the total production required \( \sum_i d_i \) (number of items), and the corresponding net requirement of material \( \sum_i w_i h_i d_i \) (square metres). The sizes of the corresponding integer programs (number of variables and constraints resulting after Cplex pre-processing) are also listed for the \( p \)-median model (10)-(16) and for its enforcement (10)-(17).
<table>
<thead>
<tr>
<th>Problem instance</th>
<th>m</th>
<th>p</th>
<th>q</th>
<th>( \sum_i d_i )</th>
<th>Total area required (( m^2 ))</th>
<th>(10)-(16)</th>
<th>(17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VR5</td>
<td>14</td>
<td>11</td>
<td>3</td>
<td>35625</td>
<td>38869.52</td>
<td>675</td>
<td>645</td>
</tr>
<tr>
<td>VR38</td>
<td>26</td>
<td>18</td>
<td>4</td>
<td>809202</td>
<td>656349.90</td>
<td>3321</td>
<td>3225</td>
</tr>
<tr>
<td>VR35</td>
<td>12</td>
<td>9</td>
<td>4</td>
<td>180834</td>
<td>152343.70</td>
<td>793</td>
<td>745</td>
</tr>
<tr>
<td>VR31</td>
<td>19</td>
<td>11</td>
<td>4</td>
<td>169717</td>
<td>117663.50</td>
<td>2980</td>
<td>2851</td>
</tr>
<tr>
<td>VR+38</td>
<td>35</td>
<td>19</td>
<td>5</td>
<td>791686</td>
<td>717527.60</td>
<td>11556</td>
<td>11271</td>
</tr>
<tr>
<td>VR+35</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>201258</td>
<td>151436.00</td>
<td>65</td>
<td>57</td>
</tr>
<tr>
<td>VR+31</td>
<td>50</td>
<td>23</td>
<td>5</td>
<td>1486614</td>
<td>1176084.00</td>
<td>23511</td>
<td>23101</td>
</tr>
<tr>
<td>OPT32</td>
<td>29</td>
<td>20</td>
<td>5</td>
<td>1280535</td>
<td>1002743.00</td>
<td>9510</td>
<td>9223</td>
</tr>
<tr>
<td>CH38</td>
<td>21</td>
<td>17</td>
<td>4</td>
<td>120750</td>
<td>89859.39</td>
<td>3762</td>
<td>3613</td>
</tr>
<tr>
<td>CH31</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>143760</td>
<td>87031.62</td>
<td>490</td>
<td>451</td>
</tr>
</tbody>
</table>

Table 2
Ten problem instances from the plant: number of distinct items required, allowance of distinct stock sizes and width changeovers, items required on the whole, net requirement of material, and sizes of the integer programs solved.

<table>
<thead>
<tr>
<th>Problem instance</th>
<th>(operator)</th>
<th>% Waste</th>
<th>% Waste</th>
<th>% Saving</th>
<th>% Waste</th>
<th>% Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>VR5</td>
<td></td>
<td>2,94</td>
<td>2,10</td>
<td>28,58</td>
<td>2,02</td>
<td>31,41</td>
</tr>
<tr>
<td>VR38</td>
<td></td>
<td>2,08</td>
<td>1,31</td>
<td>36,79</td>
<td>1,30</td>
<td>37,20</td>
</tr>
<tr>
<td>VR35</td>
<td></td>
<td>3,14</td>
<td>1,38</td>
<td>56,00</td>
<td>1,37</td>
<td>56,37</td>
</tr>
<tr>
<td>VR31</td>
<td></td>
<td>7,00</td>
<td>1,52</td>
<td>78,29</td>
<td>1,50</td>
<td>78,61</td>
</tr>
<tr>
<td>VR+38</td>
<td></td>
<td>3,22</td>
<td>0,96</td>
<td>70,24</td>
<td>0,95</td>
<td>70,52</td>
</tr>
<tr>
<td>VR+35</td>
<td></td>
<td>1,92</td>
<td>0,33</td>
<td>82,96</td>
<td>0,32</td>
<td>83,26</td>
</tr>
<tr>
<td>VR+31</td>
<td></td>
<td>2,92</td>
<td>1,58</td>
<td>45,80</td>
<td>1,58</td>
<td>46,08</td>
</tr>
<tr>
<td>OPT32</td>
<td></td>
<td>2,48</td>
<td>1,81</td>
<td>27,29</td>
<td>1,80</td>
<td>27,46</td>
</tr>
<tr>
<td>CH38</td>
<td></td>
<td>2,54</td>
<td>1,54</td>
<td>39,51</td>
<td>1,50</td>
<td>41,19</td>
</tr>
<tr>
<td>CH31</td>
<td></td>
<td>2,79</td>
<td>2,57</td>
<td>7,85</td>
<td>2,55</td>
<td>8,60</td>
</tr>
</tbody>
</table>

Table 3
Different solutions to the ten problem instances from the plant: present plant operation, program (10)-(17), program (1)-(8).
Table 3 shows the solutions obtained by the operator (1st column), compared to those computed through integer programs (10)-(17) (2nd column) and (1)-(8) (3rd column). The 2nd and 3rd columns are divided in two parts to show the percentage savings obtained by the two models. The savings obtained through the two models are agreeable: for the $p$-median model (10)-(17), they range between 7,85% and 78,29%; for model (1)-(8) between 8,60% and 78,61%.

In fact, to a total net requirement of 4.189.908 square metres, the operator adds a 2,81% waste, corresponding to about 118.393 square metres. With the $p$-median model, such a waste is reduced by 48,10%, i.e. to 61.446 square metres. The optimal solution is approximately the same: a waste reduction of 48,42%, corresponding to 61.068 square metres.

This nice behaviour of model (10)-(17) is detailed in Table 4, where, for each instance, the approximation ratio obtained in practice is compared to the theoretical bound given by Theorem 4. The noticeable gap of instance VR+31 is due to an exceptionally small requirement $d$ for a single item type.

<table>
<thead>
<tr>
<th>Problem instance</th>
<th>Theoretical bound (%)</th>
<th>Practical approximation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VR5</td>
<td>11,28</td>
<td>0,085</td>
</tr>
<tr>
<td>VR38</td>
<td>1,92</td>
<td>&lt; 0,009</td>
</tr>
<tr>
<td>VR35</td>
<td>0,85</td>
<td>0,012</td>
</tr>
<tr>
<td>VR31</td>
<td>7,05</td>
<td>&lt; 1,033</td>
</tr>
<tr>
<td>VR+38</td>
<td>66,17</td>
<td>&lt; 2,050</td>
</tr>
<tr>
<td>VR+35</td>
<td>3,94</td>
<td>0,006</td>
</tr>
<tr>
<td>VR+31</td>
<td>100,48</td>
<td>&lt; 2,049</td>
</tr>
<tr>
<td>OPT32</td>
<td>2,19</td>
<td>&lt; 0,004</td>
</tr>
<tr>
<td>CH38</td>
<td>3,30</td>
<td>&lt; 2,085</td>
</tr>
<tr>
<td>CH31</td>
<td>1,48</td>
<td>0,021</td>
</tr>
</tbody>
</table>

Table 4
Comparison between the theoretical bound (Theorem 4) and the practical approximation obtained by program (10)-(17) in the ten problem instances from the plant.

Finally, Table 5 resumes the computational performance obtained on all instances by the enforced $p$-median (10)-(17), the $p$-median (10)-(16), and model (1)-(8). The performance is expressed by the CPU time (seconds), the number of branch-and-bound nodes explored by Cplex 8.0 and the integrality gap (current solution vs. best lower bound). Data for the latter parameter are gathered.
after Cplex 8.0 pre-processing in default setting. A (*) in the CPU column indicates that the optimality of the current solution could not be certified after 1000 seconds. This never happened with the \( p \)-median models, whereas it occurred in six instances with model (1)-(8). The table gives also evidence of the effect of inequalities (17) in several instances: VR38, VR+38 and, most of all, VR+31 and OPT32.

<table>
<thead>
<tr>
<th>Problem instance</th>
<th>(10)-(17)</th>
<th>(10)-(16)</th>
<th>(1)-(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>% nodes</td>
<td>% gap</td>
</tr>
<tr>
<td>VR5</td>
<td>0,31</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VR38</td>
<td>0,73</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VR35</td>
<td>0,06</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VR31</td>
<td>0,15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VR+38</td>
<td>0,85</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VR+35</td>
<td>0,03</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VR+31</td>
<td>2,45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OPT32</td>
<td>0,40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH38</td>
<td>0,15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH31</td>
<td>0,01</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5
CPU times (seconds), nodes explored (Cplex 8.0) and integrality gap in the solutions to the ten problem instances from the plant obtained via programs (10)-(17), (10)-(16) and (1)-(8).

References


