A nested decomposition approach to a 3-stage 2-dimensional cutting stock problem

François Vanderbeck

MAB, Université de Bordeaux 1, 351, Cours de la Libération, F-33405 Talence Cedex, France

Fax: +33 (05) 57 96 21 23, Email: fv@math.u-bordeaux.fr, Url: http://www.math.u-bordeaux.fr/~fv

July 1999 (revised in January 2000)

Abstract

We consider the cutting of rectangular order pieces into stock pieces of specified width and length. The cutting process involves 3-stages of orthogonal guillotine cutting: stock pieces are cut into sections that are cut into slits that are cut into order pieces. Restrictions imposed on the cutting process make the combinatorial structure of the problem more complex but limit the scope of solution space. The objective of the problem is mainly to minimize waste, but our model also accounts for other issues such as aging stock pieces, urgent or optional orders, and fixed setup costs. Our solution approach involves a nested decomposition of the problem and the recursive use of the column generation technique: we use a column generation formulation of the problem (Gilmore and Gomory, 1965) and the cutting pattern generation subproblem is itself solved using a column generation algorithm. LP-based lower bound on the minimum cost are computed and, by rounding the LP solution, a feasible solution and associated upper bound is obtained. This approach could in principle be used in a branch-and-bound search to solve the problem to optimality. We report computational results for industrial instances. The algorithm is being used in industry as a production planning tool.

Keywords

Cutting Stock, Trim Loss, and Knapsack Problems; Integer Programming and Nested Decomposition.

Introduction

In cutting stock problems, one has a supply of pieces of stock material on one hand, and a set of demands for "small" pieces of this material on the other hand. One must satisfy these demands by cutting the required pieces out of the stock pieces. The objective is primarily to minimize the waste that is counted as the unused part of used pieces of stock material. A solution is given by a set of feasible cutting patterns, i.e. assortments of order pieces that can be cut out of a given piece of stock material, such that their accumulated production of ordered pieces covers the demands.

The specific problem considered here is 2-dimensional cutting problem where the stock and order pieces are rectangle area of specified width and length. The cutting process involves 3-stages and uses orthogonal guillotine cut only, i.e. cuts parallel to a side of the piece being cut, going from one edge of the piece to the opposite edge. In the first stage, stock pieces are cut into sections. The sections are then cut into slits. And, in the final stage, the slits are cut into order pieces. However, one has the possibility of not using the full length of a stock piece and to reserve the remaining part for later use, i.e. the remaining part returns to the stock. Thus, for our problem, one must decide what is the optimal length to be used out of the stock pieces while, in standard cutting stock problems, the used length of the stock pieces is fixed.

In the practical application that has motivated this study there are rather stringent restrictions that limit the scope of solution space:

i. The number of sections that can be cut in a stock piece is bounded: let U be that upper bound.

ii. The length of a section is bounded up due to a technical limitation of the cutting machine: let $L^{\max}$ denote the maximal section length.

iii. If part of a stock piece is returned to stock, it must be of length greater than a minimum, which we denote by $L^{\min}$.

iv. Due to order handling considerations, a slit can only include pieces of the same order and a section involve no more than 2 different types of slits (and hence no more that 2 different order types).

v. Due to the limitation on the number of knifes that can be accommodated on the cutting machine, the number of slits in any section is bounded by N.

The cutting process involves four types of waste (as illustrated in Figure 1):

1. If the sections that make a cutting pattern have a total length smaller than the stock piece length but the *unused length* is too small to be returned to stock (i.e. the unused length is less than $L^{\min}$) , then the unused part of the stock piece is counted as waste.

2. Each section involves some *setup waste* which corresponds to the part of the section length that is required to hold the piece of material at both end and another part required to test the correct positions of the knifes used to cut the slits. The total section length wasted in a setup is denoted L^{su}.

3. The length of a section is the length of its longest slit plus the setup waste length. When the slits making up a section have different lengths, there is therefore a third kind of waste referred to as *vertical waste*, which corresponds to the wasted area at the end of each short slits.

4. Finally, when the slits making up a section have a total width smaller than the stock piece width, there is a wasted slit of material referred to as *trim-loss*.

Another restriction is imposed on the cutting process:

vi. The width of the trim loss in any section must be smaller than the width of the orders involved in this section, i.e. a section is packed with as many slits of the types involved as feasible. The length of the vertical waste above any slit of a section must be smaller than the length of the order piece involved in this slit, i.e. a slit is packed with as many order pieces as feasible in the available section length.

Restriction (*vi*) amounts to saying that we only consider *maximal cutting patterns*. It was historically imposed as a "business rule" to avoid the criticisms that resulted from proposing sections where extra order pieces could have obviously been added. It is welcome in our approach as it limits further the number of combinations of slit width to be considered and it implies that the number of order pieces produced in a slit is completely determined from the section length (and vice versa).

We make full use of the simplifications implied by the above restrictions in our solution approach. In particular, restrictions (*iv*) and (*vi*) imply that the only section lengths to consider are those corresponding to integer numbers of lengthwise duplicates of the at most two orders involved in the section. Our algorithm exploits the fact that all combinations of two order lengths can be obtained from a generating set of small cardinality

(formed by the elements of a Hilbert basis of the cone of the so-called exchange vectors – see Weismantel, 1996). Our algorithm also makes some use of brute force enumeration, considering explicitly each feasible combination of order widths on each stock piece. This enumeration remains reasonable thanks to restrictions (*iv*) to (*vi*) and because, in the real-life instances that we aim to solve, the number of different order types is reasonably small, typically between 5 and 25 different order types (with different dimensional specifications), while the number of stock pieces typically remains below 20.

The objective of the problem is mainly to minimize waste. However, our model can account for other issues such as aging stock pieces, urgent or optional orders, and fixed setup costs associated with each section:

As it is common in real-life cutting problems, there is a tolerance on the production of order pieces because deviations from the target demand levels can translate into significant waste savings. Thus, there are lower and upper bounds on order productions instead of specific demands. The order list include all current orders which is often more than what can be cut out of the available stock pieces. The production lower bounds are typically set to zero for less urgent orders. They are named *optional* orders while the other are called *must-make* orders. But one cannot know a priori what can feasibly be cut out of the available stock material and the operator typically defines to many orders as must-make. Therefore, our model includes penalties for not meeting a lower bound on order production. If the operator considered an order as an absolute must-make, the associated penalty is raised automatically by the algorithm until either the order demand is satisfied or the problem is proved infeasible by bound.

A value is attached to each stock and order pieces. As waste minimization is the primary objective this value is normally set equal to the surface area of the piece. However, the operator may want to multiplied this value by a scaling factor to reflect the lesser value of a stock piece that has been in stock for a long time or the higher value of an order type that is more urgent for instance. It is also considered that the unused part of a stock piece that is returned to stock has lost some of its value. All value, cost, and penalty parameters are given in the units of a surface area for consistency. For instance, placing a figure on the fixed cost per setup amounts to specifying how much waste one is prepared to incur in order to avoid an extra setup.

The literature on 2-dimensional cutting stock problems using orthogonal guillotine cuts includes a few references that can inspire a solution approach to our problem:

Valerio De Carvalho and Guimaraes Rodriges (1994 and 1995) proposed a LP based heuristic approach to a *1-dimensional 2-stage* cutting problem where the stock material (steel rolls) is cut into "intermediate rolls" which are then cut into order rolls: they generate a priori all feasible intermediate rolls that involve one or several copies of a single order roll plus two edge trims and they formulate their problem as 1-d cutting stock problem where the cutting patterns are made of intermediate rolls (a column generation procedure is used to solve the LP formulation where the pricing subproblem is a knapsack problem whose items correspond to intermediate rolls).

Riehme, Scheithauer, Terno (1996) construct solutions to *2-dimensional 1-stage* guillotine cutting problems in two steps: first they aggregate the stock pieces sharing the same width W into one long piece and, along the length of each aggregated stock piece, they cut horizontal strips of width W that are made of order pieces so as to cover the order demands (this is solved approximately as a variant of the 1-d CSP where the cutting patterns correspond to feasible strips of varying length and the total length of the selected strips cannot exceed the available aggregate stock piece length). Then, in the second step, the selected strips become "order pieces" that must be cut into the initial stock pieces (this is solved approximately as another 1-d CSP).

Beasley (1985) proposes an exact optimisation based algorithm to find heuristic solution for the *2-dimensional n-stages* cutting stock problem: he generates a priori cutting patterns with best value (difference between order piece value and stock piece value) using the dynamic programming procedure of Gilmore and Gomory (1966) for an unlimited production of order pieces; then, he formulates the cutting problem in terms of this subset of cutting patterns (plus some heuristically generated patterns) and he obtains a feasible solution to the problem by rounding the LP solution of this restricted formulation.

The trend that arises from these references is the use of a staged approach and a priori generation to reduce the problem to solving 1-dimensional problems. We shall push these ideas further.

1 Outline of the approach

We have developed an approximate solution method for our *2-dimensional 3-stages* cutting problem that is based on a nested decomposition of the problem. The first layer of decomposition leads to considering the subproblem associated with the generation of a feasible *cutting pattern* for a given stock piece. The problem of cutting pattern generation is it-self decomposed into subproblems associated with the generation of *sections*. The third layer of decomposition consists in seeing the generation of a section as the selection of a *horizontal combination* of orders that can fit along the width of the stock piece and the selection of a *section length*. Thus, order pieces are combined into horizontal combinations, which are used to generate sections, and sections are used to generate the cutting patterns that compose a solution to the cutting problem. The generations of horizontal combinations is done a priori by brute force enumeration. As argued in the introduction, their number is reasonably small in the instances we need to consider. However, the number of feasible sections and cutting patterns are huge. Hence, they must be handled implicitly using a column generation technique.

Thus, the cutting problem is formulated implicitly as an integer program whose columns correspond to feasible cutting patterns. A nested column generation algorithm is used to compute the linear programming lower bound on the minimum cost. By that we mean that the column generation subproblem is itself solved using a column generation procedure. I.e., the subproblem that consists in generating a cutting pattern with minimum reduced cost is formulated as an bounded integer knapsack problem whose columns correspond to feasible sections. The feasible sections are generated dynamically as the solutions of the sub-sub-problem where a section with maximum ratio of value per unit of stock length used is generated (for each feasible horizontal combinations of orders, we optimize the section length). The section generation subproblem is solved exactly, but the cutting pattern generation subproblem is only solved approximately: we compute lower and upper bounds on the minimum reduced cost of a cutting pattern. The lower bounds on the pattern reduced costs give rise to a Lagrangian bound on the cutting problem that is a lower estimate of the LP bound. Then, a rounding heuristic is used to generate incumbent feasible solutions to the cutting problem. In principle, this approach can be used at each node of a branch-and-bound tree to solve the problem to optimality.

The rest of this paper is organized so as to present each level of the embedded decomposition in turn: Section 2 presents the formulation of our cutting problem and the column generation approach used to solve it. Section 3 presents the column generation

subproblem and our approach to it. In Section 4, we define horizontal combinations and show how a section with optimal value per unit of length can be generated from a given horizontal combination. The section length can be optimized using a greedy algorithm once one observes that all vertical combinations of two order lengths can be obtained from a generating set. From the generation of sections with best value per unit of length for a given horizontal combination, we go on to the generation of multiple sections involving one horizontal combination, which we call *partial cutting pattern*, and to the generation of cutting patterns mixing different horizontal combinations (Section 5). Section 6 presents the rounding heuristic. We then summarize the overall algorithm in Section 7 and present some computational results on real-life problems. Finally, we say how our model could be extended to other problem variants and how the algorithm could be used in a branch-and-bound search. And we conclude by outlying the strong points of our approach that gave rise to an optimiser used in industry.

2 Solving the Cutting Problem

In the mathematical programming formulations of our problem and its subproblems, we use the following notations. The stock pieces are indexed by $k = 1, \ldots, K$. For each stock piece k, W_k and L_k denote respectively its width and length, V_k denotes its value (normally, we set $V_k = W_k\, L_k$), and r_k denotes the value per unit of length of a part returned to stock ($r_k < V_k/L_k$). The order pieces are indexed by $i = 1, \ldots, n$. For each order piece i, w_i and l_i denote respectively its width and length, $\underline{d}_i$ and $\overline{d}_i$ denote respectively the lower and the upper bound on its production, v_i denotes its value (normally, we set $v_i = w_i\, l_i$), and p_i denotes the penalty per unit short of the required production (we can set $p_i = 10\, v_i$, for instance). Let f^{su} denote the fixed cost per section. Notations for the parameters of the cutting process were introduced above.

In the standard formulation of 1-dimensional cutting stock problems, one defines variables associated with the selection of feasible cutting patterns. This extensive formulation due to Gilmore and Gomory (1961-1963) involves a number of variables that is exponential in n but its LP relaxation provides typically much better bound than an alternative compact formulation using a polynomial number of variables that describe the composition of cutting patterns. The large number of variables can be dealt with implicitly using a column generation procedure: i.e., the columns and associated variables are generated in the course of the optimisation if and when they are likely to enter in the solution.

Gilmore and Gomory (1965) showed that a similar column generation formulation could be used for 2-dimensional problem as well. However, the generation of the feasible cutting patterns is more complex. Assuming that Q_k represent the set of feasible cutting patterns for stock piece k, we associate a binary variable λ_q^k to each element q of Q_k that takes value 1 if cutting pattern $q \in Q_k$ is chosen to cut stock piece k and zero otherwise. For convenience, the notation q is used to denote both the index of the element $q \in Q_k$ and the vector in $\mathbb{N}^n$ defining the production of cutting pattern $q \in Q_k$ whose components q_i represent the number of cut order piece i in this cutting pattern. Moreover, L_q^k denotes the length of the part of the stock piece k that is returned to stock in cutting pattern $q \in Q^k$; n_q^k denotes the number of sections in cutting pattern $q \in Q^k$, and c_q^k denotes its cost:

$$c_q^k = V_k - r_k \, L_q^k + f^{su} \, n_q^k - \sum_i v_i \, q_i \ .$$

Then, our problem takes the form

$$Z \ = \ \min \qquad \sum_{k=1}^{K} \sum_{q \in Q_k} c_q^k \, \lambda_q^k + \sum_{i=1}^{n} p_i \, u_i \tag{1}$$

$$[M] \qquad \text{s.t.} \tag{2}$$

$$\sum_{k=1}^{K} \sum_{q \in Q_k} q_i \, \lambda_q^k \ \geq \ \underline{d}_i - u_i \qquad \forall \, i = 1, \ldots, n \tag{3}$$

$$\sum_{k=1}^{K} \sum_{q \in Q_k} q_i \, \lambda_q^k \ \leq \ \overline{d}_i \qquad \forall \, i = 1, \ldots, n \tag{4}$$

$$\sum_{q \in Q_k} \lambda_q^k \ \leq \ 1 \qquad \forall \, k = 1, \ldots, K \tag{5}$$

$$\lambda_q^k \ \in \ \{0, 1\} \qquad \forall \, k = 1, \ldots, K, \ q \in Q_k.$$

$$u_i \ \geq \ 0 \qquad \forall \, i = 1, \ldots, n$$

where the variable u_i denotes the number of unit short from the lower bound requirement on the production of order i. In a column generation approach, this formulation of the global problem is traditionally referred to as the *master problem*, M.

We obtain a lower bound on our problem by computing a lower estimate of the LP relaxation value of the above formulation, using a column generation procedure. At an intermediate stage of the procedure, we have only a subset of the feasible cutting patterns for each stock piece (those generated so far). We solve the LP relaxation of the formulation restricted to those known patterns only (called the restricted master problem). We record the dual solution of this restricted formulation: let us denote by $\pi \in \mathbb{R}_+^n$, $\mu \in \mathbb{R}_+^n$,

and $\sigma \in I\!R_+^K$ the dual variables respectively associated with constraints (3), (4), and (5). With this set of dual prices, we compute a lower and an upper bound on the minimum reduced cost of any cutting pattern for each stock piece k, i.e. we solve the pricing subproblems approximately.

The reduced cost of a cutting pattern $q \in Q_k$ is

$$rc_q^k = c_q^k - \sum_{i=1}^{n} (\pi_i - \mu_i)\, q_i + \sigma_k \ .$$

Thus, we compute a lower bound $\underline{rc}^k$ and an upper bound $\overline{rc}^k$ on the optimal solution of each subproblems

$$\min\{rc_q^k \ : \ q \in Q_k\} \tag{6}$$

for $k = 1, \ldots K$. The lower bound $\underline{rc}^k$ is based on the LP solution of a relaxation of the subproblem (6), while the upper bound $\overline{rc}^k$ is obtained using a greedy heuristic that constructs a feasible solution to (6). If the reduced cost $\overline{rc}^k$ is negative, the associated cutting pattern is added to the current master formulation. On the other hand, the reduced cost lower bounds $\underline{rc}^k$ are used to compute a Lagrangian bound on the master problem (see Vanderbeck and Wolsey, 1996):

$$L(\pi, \mu, \sigma) = Z_{LP}^R - \sum_{k=1}^{K} \underline{rc}^k \ , \tag{7}$$

where Z_{LP}^R denotes the objective value of the current LP solution to the restricted master problem.

We initialize the master formulation with one cutting pattern for each stock sheet that we obtained as the greedy solution to the pricing subproblem (6) for $(\pi, \mu, \sigma) = 0$. The restricted master formulation always admits a feasible solutions thanks to the presence of variables u_i's. We then iterate on the column generation procedure that we just outlined and we record the best intermediate lower bound on the master LP value:

$$LB = \max_{(\pi,\mu,\sigma) \in D} \{L(\pi, \mu, \sigma)\} \ , \tag{8}$$

where D is the set of dual solutions of the restricted master LP that were obtained at intermediate iterations. The column generation procedure is ended either when we obtain $LB = Z_{LP}^R$ proving that the current LP solution is optimal for the unrestricted master problem (Vanderbeck and Wolsey 1996), or when the current column generation iteration has produced no feasible cutting pattern with negative reduced cost — then, the current

lower bound LB may only be a lower approximation of the master LP solution. In any case, a rounding heuristic is then used to construct feasible solutions to the master integer problem as explained in Section 6. Below we explain how we obtain lower and upper bounds on subproblems (6).

3 The column generation subproblem

If we exclude the constants, the pricing subproblem (6) for stock sheet k takes the form:

$$v^k_{\max} = \max\{\sum_i (v_i + \pi_i - \mu_i)\, q_i + r_k\, L^k_q - f^{su}\, n^k_q :\ q \in Q_k\}\ , \tag{9}$$

where orders for which $(v_i + \pi_i - \mu_i) \leq 0$ can be ignored. Our approach to solving problem (9) is similar to the approach used to tackle the master problem (2). We give a formulation of Q_k in terms of variables associated with the selection of feasible sections. We solve approximately the LP relaxation of that exponential size formulation using a column generation procedure. This gives us an upper bound on $v^k_{\max}$. We then apply a greedy heuristic that makes use of the LP solution to obtain a lower bound and associated incumbent solution.

Let S_k be the set of feasible sections that can be cut in stock piece k. To each section $s \in S_k$, we associate a variable x_s that denotes the number of copies of section s chosen in the cutting pattern that is being constructed. Moreover, let L_s denote the length of section s, and let $q^s \in I\!N^n$ be an integer vector whose components q^s_i indicate the number of order pieces cut in section s for each order $i = 1, \ldots, n$. Then, problem (9) can be formulated as

$$\begin{aligned}
v^k_{\max} = \max \quad & r_k\, l \ + \ \sum_{s \in S_k} (\sum_{i=1}^{n} (v_i + \pi_i - \mu_i)\, q^s_i - f^{su})\, x_s \\
& \text{s. t.} && (10)\\
l + \sum_{s \in S_k} L_s\, x_s \ &\leq \ L_k \\
\sum_{s \in S_k} q^s_i\, x_s \ &\leq \ \overline{d}_i \qquad \forall i && (11)\\
L_{\min}\, y \leq l \ &\leq \ L_k\, y \\
l \ &\geq \ 0 \\
y \ &\in \ \{0,1\} \\
x_s \ &\in \ I\!N \qquad \forall s \in S_k\ ,
\end{aligned}$$

where y is a binary variable that takes value 1 if part of the stock piece is returned to stock in the generated cutting pattern and zero otherwise, and l is the length of the part that is returned to stock. As $r_k \geq 0$, we only consider sections for which $\sum_{i=1}^{n}(v_i + \pi_i - \mu_i)\, q_i^s > f^{su}$. A solution of (l^*, y^*, x^*) of (10) defines a cutting pattern $(L_q^k = l^*,\ n_q^k = \sum_{s\in S_k} x_s^*,\ q = \sum_{s\in S_k} q_s\, x_s^*)$ for stock piece k.

We observe that problem (10) is mostly an integer knapsack problem in variables x_s' s with additional constraints (11) and the extra issue of an eventual return to stock of part of the stock piece. The latter issue is resolved by considering separately the optimisation over solutions with $y = 0$ and solutions with $y = 1$. The production upper bound constraints are relaxed in a Lagrangian fashion to give rise to two bounded knapsack subproblems (respectively for $y = 0$ and $y = 1$):

$$
\begin{aligned}
v^0(\nu) = \max \quad & \sum_{s\in S_k} (\sum_{i=1}^{n}(v_i + \pi_i - \mu_i - \nu_i)\, q_i^s - f^{su})\, x_s \\
\text{s. t.} \quad & \\
\sum_{s\in S_k} L_s\, x_s \quad & \leq \quad L_k \\
I\!N \ni x_s \quad & \leq \quad u_s = \min_i \left\lfloor \frac{\overline{d}_i}{q_i^s} \right\rfloor \qquad \forall s \in S_k\ ,
\end{aligned}
\tag{12}
$$

and

$$
\begin{aligned}
v^1(\nu) = \max \quad & r_k\,(L_{\min} + l) \quad + \quad \sum_{s\in S_k} (\sum_{i=1}^{n}(v_i + \pi_i - \mu_i - \nu_i)\, q_i^s - f^{su})\, x_s \\
\text{s. t.} \quad & \\
L_{\min} + l + \sum_{s\in S_k} L_s\, x_s \quad & \leq \quad L_k \\
l \quad & \geq \quad 0 \\
I\!N \ni x_s \quad & \leq \quad u_s = \min_i \left\lfloor \frac{\overline{d}_i}{q_i^s} \right\rfloor \qquad \forall s \in S_k\ ,
\end{aligned}
\tag{13}
$$

where ν_i denotes the Lagrangian price associated with the relaxation of constraint (11) for the corresponding i and $(L_{\min} + l)$ now denotes the length of the part returned to stock. This relaxation of problem (10) yields an upper bound:

$$
v_{\max}^k \quad \leq \quad \max_{y=0,1} \min_{0\leq\nu\leq(v_i+\pi_i-\mu_i)} \{v^y(\nu) + \sum_i \nu_i\, \overline{d}_i\}\ .
$$

We use a sub-gradient algorithm to perform the optimisation in ν: If a production upper bound is exceeded, we increase the associated ν_i by $\alpha\,(v_i+\pi_i-\mu_i-\nu_i)$, while if production is strictly less than the upper bound, we decrease ν_i by $\alpha\nu_i$, where $0 < \alpha < 1$ is a step size.

In practise, we stop the sub-gradient algorithm prematurely as soon as the current solution satisfies constraints (11): Such feasible solution would only be optimal for (10) with y fixed if it satisfied the complementary slackness conditions: $(\sum_{s \in S_k} q_i^s \, x_s - \overline{d}_i) \, \nu_i = 0 \; \forall i$; otherwise, the ν_i's could be over-estimates of the optimal Lagrangian prices (which we try to avoid by choosing a small step size α). However, this heuristic stopping rule yield relatively few iterations (typically 2 or 3), except at the outset of the algorithm when the dual prices π_i can be relatively high. In the latter case, we stop the optimisation in ν after 10 iterations (anyway, at the outset, the bound (7) would be weak even with we had the exact values of $v_{\max}^k$ and the associated reduced cost rc_k).

At each iteration of the sub-gradient algorithm (initially for $\nu = 0$), we would need to solve the bounded knapsack problems (12) and (13) approximately. The LP relaxations can be solved using a greedy heuristic (see Martello and Toth, 1980): consider the sections s in order of non-increasing value per unit of length, v_s/L_s, where $v_s = (\sum_{i=1}^n (v_i + \pi_i - \mu_i - \nu_i) \, q_i^s - f^{su}) > 0$ and fill the knapsack capacity with the most attractive sections while satisfying their upper bounds. Thus, assuming the sections have been sorted, the LP solution for (12), f.i., would be $x_s^{LP} = \min\{(L_k - \sum_{\rho=1}^{s-1} L_\rho x_\rho^{LP})/L_s, \; u_s\}$ for $s = 1 \ldots, |S^k|$. An incumbent integer solution can be obtained by taking integer parts in the above greedy procedure: f.i., $x_s^{inc} = \min\{\lfloor (L_k - \sum_{\rho=1}^{s-1} L_\rho \, x_\rho^{inc})/L_s \rfloor, \; u_s\}$ for problem (12). For problem (13), if the current best section ratio is less than the returned to stock value $r_k \geq 0$, the remaining capacity of the knapsack will be returned to stock. Because the cardinality of S^k is typically huge and as the feasible sections are not known a priori, we need to implement such greedy procedures in a column generation framework.

In a column generation approach to solving the LP relaxation of the knapsack problems (12) and (13), the subproblem consists in generating a feasible section with maximum ratio of value per length, i.e., it is

$$\max\{\frac{(\sum_{i=1}^n (v_i + \pi_i - \mu_i - \nu_i) \, q_i^s - f^{su})}{L_s} \; : \; s \in S_k\} \tag{14}$$

In the next Section, we present an exact optimization procedure for this subproblem. Then, in the following section, we say how the dynamic generation of sections can be integrated into the above greedy procedures to generate upper and lower bounds on the cutting pattern generation pricing problem.

4 The generation of sections

Given restrictions (iv) and (v) on the cutting process, building a feasible section for a stock piece k can be decomposed into choosing a horizontal combination of order, involving at most N slits and at most 2 order types, that can fit along the width of the stock piece and then optimize the section length. Moreover, restriction (vi) further limits the number of feasible horizontal combinations of orders to those yielding a trim loss that is too small to accommodate an extra slit of the orders involved. Therefore, the set of feasible horizontal combinations of orders for stock piece k can be defined as

$$\begin{aligned} H_k = \{(h_1, \ldots, h_n) \in I\!N^n : \quad & W_k - \min\{w_i : \ \delta(h_i) > 0\} < \sum_{i=1}^{n} w_i h_i \leq W_k \ , \\ & \sum_{i=1}^{n} h_i \leq N \ , \\ & \sum_{i=1}^{n} \delta(h_i) \leq 2 \ \} \end{aligned} \tag{15}$$

where $\delta(x) = 1$ if $x > 0$ and zero otherwise. We generate all the elements of H_k by enumeration on all singletons and pairs of orders and all feasible number of duplicates for the first order. Thus, $|H_k| \in O(n^2 N)$. Hence, for a fixed number of knifes, the number of horizontal combinations of orders is polynomial.

From a given horizontal combination of orders $h \in H_k$, we shall generate a section whose length is determined by the number of order pieces cut along the length of the stock piece. For a horizontal combination h involving *one order*, say $h_i > 0$, while $h_j = 0$ for all $j \neq i$, the section length can only be

$$L_s = L^{su} + l_i \, n_i \quad \text{for some } n_i \in \{1, \ldots, \min\{ \left\lfloor \frac{\overline{d}_i}{h_i} \right\rfloor , \ \left\lfloor \frac{L^{\max} - L^{su}}{l_i} \right\rfloor \} \ \} \ .$$

For a horizontal combination involving *two orders*, say $h_i > 0$ and $h_j > 0$, while $h_k = 0$ for all $k \neq i$ or j, the section length is determined by the longest slit. We refer to the order present in the longest slit as the *dominant order*, say it is i. Once the dominant order i is determined, the section length L_s is thus given by the number n_i of duplicates of i as above. The number of duplicates for the second order is then uniquely defined as $n_j = \lfloor (L_s - L^{su})/l_i \rfloor$ because of restriction (vi). This implies another bound on n_i: $n_i \leq \left\lceil (\left\lfloor \frac{\overline{d}_j}{h_j} \right\rfloor \ l_j)/l_i \right\rceil$. The production of order piece of such a section is $q_i^s = n_i \, h_i$ (and $q_j^s = n_j \, h_j$ for two-order combinations) while the other components of q^s are null. Thus, we have implicitly defined the set S_k.

To solve problem (14), we enumerate on the horizontal combinations of orders and, for each $h \in H_k$, we optimized on L_s to obtain the associated section with best ratio of value per unit of length. For a horizontal combination h involving one order, the optimization in L_s is trivial: one simply needs to set n_i to its maximum feasible value as this allows to amortize the setup waste and cost on the longest possible length. For a horizontal combination involving two orders (say i and j where i is the dominant order), we observe that only a small number of length setting needs to be considered: There are the section length associated with the pairs (n_i, n_j) marking the different thresholds beyond which there is an decrease in *vertical waste* per unit of length.

For a horizontal combination h involving two orders (i and j), the problem of finding an optimal section length can be expressed in terms of variables n_i and n_j that are not independent since

$$n_j = \left\lfloor \frac{l_i \, n_i}{l_j} \right\rfloor$$

if i is the dominant order. Each pair (n_i, n_j) gives rise to a vertical waste

$$r(n_i, n_j) = (l_i \, n_i - l_j \, n_j) \, h_j \, w_j \; .$$

Weismantel (1996) has shown that the set of feasible pairs (n_i, n_j) (which he calls exchange vectors) can be generated from a small subset of them, which we denote (n_i^t, n_j^t), for $t = 1, \ldots, T_{ij}$. This generating subset defines a Hilbert basis of the cone generated by the pairs (n_i, n_j) (for n_i unbounded). Thus all feasible exchange vectors can obtained from the generating set: $(n_i, n_j) = \sum_t (n_i^t, n_j^t) \, y_t$ with $y_t \in I\!N \; \forall t$. The elements (n_i^t, n_j^t), for $t = 1, \ldots, T_{ij}$, of this generating set are the combinations that give rise to successive local minima in vertical waste: i.e. $r(n_i^t, n_j^t) \leq r(n_i, n_j)$ for all (n_i, n_j) with $n_i \leq n_i^t$. Weismantel (1996) also gave a recursive procedure to obtain this generating set and a procedure to decompose any exchange vector (n_i, n_j) in the Hilbert basis. Below we give these procedures which we have adapted to our problem.

Given a horizontal combination involving order i and j where i is the dominant order, the generating set $\{(n_i^t, n_j^t)\}_{t=1,\ldots,T_{ij}}$ can be obtained using the procedure given in Table 1 where (n_i, n_j) (respectively (m_i, m_j)) denotes the combination with minimal residuum $r_n = n_i \, l_i - n_j \, l_j$ (respectively the combination with maximal residuum $r_m = m_i \, l_i - m_j \, l_j$) encountered so far, f is the minimum number of residuum r_n which one must add to

residuum r_m to fit an extra unit of j, and

$$n_i^{\max} = \min\{ \left\lfloor \frac{\overline{d}_i}{h_i} \right\rfloor , \left\lfloor \frac{L^{\max} - L^{su}}{l_i} \right\rfloor , \left\lceil \frac{\left\lfloor \frac{\overline{d}_j}{h_j} \right\rfloor l_j}{l_i} \right\rceil \} . \tag{16}$$

Having computed the generating set, a vertical combination (n_i, n_j) can be expressed as a unique integer combination of its elements, i.e., $(n_i, n_j) = \sum_t (n_i^t, n_j^t)\, y_t$ with $y_t \in I\!N\ \forall t$, using the procedure given in Table 2.

Table 1: Procedure to construct the generating set of all vertical combinations (n_i, n_j) for a given horizontal combination involving two orders

Step 0: Let $m_i = n_i = 1$.
Let $m_j = n_j = \left\lfloor \frac{n_i\, l_i}{l_j} \right\rfloor$.
Let $r_m = r_n = n_i\, l_i - n_j\, l_j$.
Let $t = 0$.
Step 1: **If** $(n_i <= n_i^{\max})$, let $t = t + 1$ and
record $(n_i^t, n_j^t) = (n_i, n_j)$
Else, goto Step 4
Step 2: **If** $(r_n = 0)$, **goto Step 4**
Step 3: **Compute** $f = \left\lceil \frac{l_j - r_m}{r_n} \right\rceil$, then
the next smallest residuum combination is:
$n_i' = f\, n_i + m_i$ and $n_j' = f\, n_j + m_j + 1$.
The combination just before it has largest residuum:
$m_i' = (f-1)\, n_i + m_i$ and $m_j' = (f-1)\, n_j + m_j$.
The resulting residuum are $r_n' = f\, r_n + r_m - l_j$ and $r_m' = m_i\, l_i - m_j\, l_j$.
Then, reset $(n_i, n_j) = (n_i', n_j')$, $r_n = r_n'$,
$(m_i, m_j) = (m_i', m_j')$, $r_m = r_m'$, and **Goto Step 1**.
Step 4: $T_{ij} = t$.

We can now explain how to generate a section with best ratio of value per unit of length from a given horizontal combination, $h \in H_k$, involving two orders i and j, i being the dominant order. We construct the generating set of all vertical combination that are feasible for h, $\{(n_i^t, n_j^t)\}_{t=1,\ldots,T_{ij}}$, using the above procedure. Each element t, defines what we call a *block* whose production yield is

$$q_i^t = h_i\, n_i^t \text{ pieces of } i \text{ and } \quad q_j^t = h_j\, n_j^t \text{ pieces of } j. \tag{17}$$

Table 2: Procedure to decompose a vertical combination (n_i, n_j) into the elements of the generating set.

Step 0: $y_t = 0 \; \forall t. \; t = T_{ij}$.
Step 1: while $(n_i^t, n_j^t) > (n_i, n_j)$, let $y_t = 0$ and $t = t - 1$
Step 2: while $(n_i^t, n_j^t) <= (n_i, n_j)$, $(n_i, n_j) = (n_i, n_j) - (n_i^t, n_j^t)$ and $y_t = y_t + 1$.
Step 3: If $(n_i, n_j) > (0, 0)$, goto Step 1.

Its value and length are respectively

$$v_t = (v_i + \pi_i - \mu_i - \nu_i)\, q_i^t + (v_j + \pi_j - \mu_j - \nu_j)\, q_j^t \; , \tag{18}$$

$$L_t = l_i \, n_i^t \;\; \text{with } n_j^t = \left\lfloor \frac{L_t}{l_j} \right\rfloor \; . \tag{19}$$

By construction the elements of the generating set have the property that

$$r(n_i^t, n_j^t) < r(n_i^{t-1}, n_j^{t-1}) \quad \text{for} \;\; t = 2, \ldots, T_{ij} \; .$$

Since element T_{ij} yields the smallest vertical waste, it gives rise to the block with highest value per unit of length, v_t/L_t. Hence, it is optimal to take in the section as many copies of this block as feasible. This yields a section with ratio $(v_{T_{ij}} \, y_{T_{ij}} - f^{su})/(L_{T_{ij}} \, y_{T_{ij}} + L^{su})$ where $y_{T_{ij}}$ denotes the maximum number of copies of the block based on element T_{ij} that can fit in a section. Then, one needs to consider the other elements $t = T_{ij} - 1, \ldots, 1$ in decreasing order of their index. If adding the block associated with current element in the section improves the current ratio, i.e. if v_t/L_t is larger than the current ratio, then it is optimal to take the largest number of copies of it that can feasibly fit in the section. Else, the procedure stops, as no more improvement in the value of the section ratio can be expected.

Observe that the procedure for generating sections from horizontal combinations with one order is just a special case of the above where there is only one element in the generating set, i.e. $T_i = 1$, and the associated block entity is defined by

$$n_i^1 = 1 \quad \text{while} \quad n_j^1 = 0 \; . \tag{20}$$

In the sequel, we therefore make no notational distinctions between *one-order* and *two-order* sections.

5 The generation of cutting patterns

We presented the cutting pattern generation subproblem in Section 3 and concluded that we would need to be able to generate sections with maximum ratio of value per unit of length to solve it. We have just seen that a greedy procedure allows to generate a section that solves problem (14). We can now resume the presentation of the resolution of the cutting pattern generation subproblem (10).

In fact, to solve the LP relaxation of knapsack problems (12) and (13) using a column generation procedure, it is *not enough* to be able to solve subproblem (14). Indeed, once we have generated a section with best ratio and taken it into the knapsack solution, we must generate the section with the second best ratio, and at the t^{th} iteration of the greedy procedure, the candidate section to enter the knapsack solution is that with the t^{th} best ratio of value per unit of length. It is of course difficult to adapt the section generation procedure so as to guarantee to generate a section with t^{th} best ratio. Instead, we solve a different formulation of problems (12) and (13) that is amenable to our section generation procedure.

We said that sections are generated from a fixed horizontal combination. Then, a section might be replicated in the cutting pattern that we are building, and we might also include in that cutting pattern other sections generated from the same horizontal combination but having a different length. Hence, the idea leading to the proposed reformulation is to generate all sections based on a given horizontal combination in one go and to build a partial cutting pattern out of them. Thus, we define *a partial cutting pattern* as a sequence of sections based on the same horizontal combination. The knapsack problems (12) and (13) can be reformulated in terms of partial cutting patterns.

Let $P_k(h)$ denote the set of feasible partial cutting patterns for stock piece k based on horizontal combination $h \in H_k$ and $P_k = \cup_{h \in H_k} P_k(h)$. Let us define a *core subproblem* for the generation of cutting patterns whose formulation is:

$$\max \quad r\, l + \sum_{p \in P_k} \Big(\sum_{i=1}^{n} (p_i + \pi_i - \mu_i - \nu_i)\, q_{i\,p} \quad - \quad f^{su}\, n_p\Big)\, z_p$$

[core subproblem] s.t.

$$l + \sum_{p \in P_k} L_p\, z_p \quad \leq \quad L \tag{22}$$

$$\sum_{p \in P_k(h)} z_p \quad \leq \quad 1 \quad \forall h \in H_k \tag{23}$$

$$
\begin{aligned}
l &\geq 0 \\
z_p &\in \{0,1\} \quad \forall p \in P \ ,
\end{aligned}
$$

where $q_{i\,p}$ is the number of pieces of order i cut in partial cutting pattern p, while n_p is the number of sections; $r = r_k$ and $L = L_k - L^{\min}$ if a return to stock is considered (case $y = 1$), otherwise (case $y = 0$), $r = 0$, and $L = L_k$. Problem (21) is exactly equivalent to problem (12) or (13) depending on the values of r and L (the objective differs by the constant $r_k\, L^{\min}$ in the case $y = 1$). A solution (z, l) of (21) give rise to a solution (x, l) of (12) or (13) such that $x_s = \sum_p n_s(p)\, z_p$ where $n_s(p)$ is the number of duplicates of section s in partial cutting p. Inversely, $z_p = \Pi_{s \in p}\, \delta(x_s = n_s(p))$, where $\delta(a = b) = 1$ is $a = b$ and zero otherwise.

Problem (21) is a binary knapsack problem with special ordered set constraints (23) that are disjoint. Its LP relaxation is also solvable by a greedy procedure (Johnson and Padberg, 1981). The presence of constraints (23) facilitates the greedy solution of problem (21) using a column generation procedure. Indeed, the LP relaxation can be solved in two rounds of generation of partial cuttings. The algorithm is given in Table 3, where $v_p = (\sum_{i=1}^{n}(p_i + \pi_i - \mu_i - \nu_i)\, q_{i\,p} - f^{su}\, n_p)$, $KNLP(P) \equiv \max\{r\, l + \sum_{p \in P} v_p\, x_p :$ $l + \sum_{p \in P} L_p\, x_p \leq L;\ l \geq 0,\ x_p \geq 0\ \forall\, p \in P\}$ is the LP relaxation of a standard binary knapsack problem that can be solved easily by a greedy algorithm, and κ represents the value per unit of length of the break item in the solution of $KNLP(P)$, i.e. κ is the value of the dual variable associated with the knapsack constraint (22). The second round of partial cutting generation in Step 2 of the procedure we give in Table 3 accounts for the fact that the LP solution of a binary knapsack problem with disjoint special ordered sets will involve a convex combination of two items from the same ordered set if this allows to assign a greater share of the knapsack capacity to a set with items of large value per unit of length (see Johnson and Padberg, 1981).

To generate a partial cutting pattern with maximum ratio of value per unit of length from a given horizontal combination, $h \in H_k$, as it is required in Step 1 and 2 of the procedure we presented in Table 3, we proceed as we did for the generation of sections. For each possible value of $n_p = 1, \ldots, U$ (where U is the upper bound on the number of sections in a cutting pattern), we consider filling a part of the stock piece of length no more than $L^{\max}\, n_p$ with the blocks associated to the elements of the Hilbert basis (blocks are defined in (17-20)). Starting with the blocks that yield the highest value per unit of length, we add more blocks as long as it improves the ratio of value per unit of length of the partial cutting pattern that is being constructed.

Table 3: Procedure for solving the LP relaxation of the core subproblem (21).

Step 0: $P = \emptyset$, $\kappa = r$.

Step 1: For each $h \in H_k$, do:

Solve $\max\{r_p = \frac{v_p}{L_p} \ : \ p \in P_k(h) \text{ and } r_p > \kappa\}$.

If a solution p^* is found, set $P = P \cup \{p^*\}$,

solve $KNLP(P)$, and update κ.

Step 2: For each $h \in H_k$ such that $x_p = 1$ in the current

solution of $KNLP(P)$ for some $p \in P_k(h) \cap P$, do:

Solve $\max\{r_h = \frac{v_{p'} - v_p}{L_{p'} - L_p} \ : \ p' \in P_k(h) \text{ and } r_h > \kappa\}$.

If a solution p^* is found, set $P = P \cup \{p^*\}$ and

solve $KNLP(P)$ with additional constraint $\sum_{p \in P_k(h)} x_p \leq 1$

and update κ.

This generation procedure will only produce what we call *pseudo partial cuttings*. A pseudo partial cutting pattern is a *real* partial cutting pattern only if the blocks forming it can be partitioned into n_p sections of length less than $L^{\max}$. The procedure used to generate pseudo partial cuttings does not guarantee that such partition is feasible, even though the blocks have typically a small length and in any case smaller than $L^{\max}$ (see (16)). In fact, checking that a pseudo partial cutting can give rise to a real partial cutting pattern amounts to solving a bin packing problem (with bins of size $L^{\max}$); thus, it is NP-complete. We do not perform such a check. Instead, we solve problem (21) over the set of pseudo partial cuttings, $P'_k \supseteq P_k$. Since we consider a relaxation of the problem, the LP bound that we obtain is still a valid upper bound on problem (21).

To obtain a lower bound on problem (21) and, in the process, to generate a feasible cutting pattern, we use a greedy heuristic based on the dynamic generation of sections. Thus, we consider the case $y = 0$ and $y = 1$ separately and we construct a feasible solution by iteratively generating and incorporating the section with the best ratio from amongst all sections that can fit in the remaining length of the stock piece and that have an order production that does not exceed the remaining gap to the production upper bounds (so as to satisfy constraints (11)). This attempt at constructing a good cutting pattern is performed after the sub-gradient optimisation, since we found that having good values for the ν penalties helps in generating good sections that covers all type of orders. A second

heuristic is used that differs from the first one by the first sections that is incorporated: we initially put into the cutting pattern the section that was the break item in the LP solution of the knapsack problem (21). Applying these two heuristics to knapsack problem (12 or 13) amounts to enumerating over a core of size 1 (Pisinger, 1999).

6 The rounding heuristic

Good feasible solution to our cutting problem are obtained from LP solutions of the restricted master problem (2) using a rounding procedure: we round-up the master variables that has the largest fractional part in the LP solution. After fixing a binary variable of (2) to one, we consider the residual problem with the remaining demands and stock pieces and we return to the column generation solution of the LP relaxation of the master formulation associated with the residual problem. It is indeed important to combine the rounding procedure with a dynamic generation of further cutting patterns because the set of cutting patterns generated for the LP solution of the original problem might not contain an integer solution to the residual problem that avoids penalties for under-production even when one exists. Hence, proceeding to do column generation for the residual problem allows to generate cutting patterns that complement the partial solution while their production levels is within the admissible interval on the residual demands. The idea of concentrating the optimisation on a residual problem, having fixed part of the solution, is often encountered in the literature on cutting problems. The combination of the rounding procedure with column generation was used for the 1-dimensional cutting stock problem in Vanderbeck (1996).

We go through several passes of the rounding heuristic (typically 3 to 5), each pass differs by what cutting pattern is the first to be rounded up. The rounding procedure first fix to their current value any master variable that is integer in the LP solution. Then, on the t^{th} pass, the first variable to be rounded up is that with the largest fractional part from amongst those that were not rounded up first in any previous passes $1, \ldots, t-1$ (this insures some diversity in the heuristic search). Subsequently, we round up that variable with the largest fractional part, after having fixed the variables that happen to be integer. Each time a variable is fixed, the upper and lower bounds on productions are updated and all cutting patterns with an order production that violates a production upper bound or that concerned stock pieces that are already used are momently deleted. If no more cutting pattern can be generated but some order production lower bounds remain unsatisfied, we incorporate a penalty in the heuristic solution.

7 The Overall Algorithm and Computational Results

In Table 4, we summarize the overall approach to our cutting problem. At the outset, we generate horizontal combinations and the blocks associated with the generating elements of the vertical combinations. Then, we use a column generation procedure to solve the LP relaxation of the master formulation, but we stop the procedure before its completion if our heuristic generation of cutting patterns fails to produce any negative reduced cost column. In any case, we obtain a lower bound on the master: the best of the intermediate Lagrangian bounds (7). Finally, we construct feasible solutions to the cutting problem using a rounding heuristic that uses the column generation procedure on the residual problems.

Table 4: The algorithm for solving problem (2) approximately.

Step 0: $LB = -\infty$

Step 1: For each $k = 1, \ldots, K$, generate set H_k (15), and
for each $h \in H_k$, construct the generating set $\{(n_i^t, n_j^t)\}_{t=1,\ldots,T_{ij}}$ (see Table 1)
and the associated blocks (see 17-20).

Step 2: For each $k = 1, \ldots, K$, generate an initial cutting pattern
using the greedy heuristic presented in Section 5
for dual prices $(\pi, \mu) = 0$.
Initialize the master formulation (2) with the associated columns.

Step 3: Solve the restricted master LP relaxation, record its value Z_{LP}^R, and
collect the dual solution (π, μ, σ).

Step 4: For each $k = 1, \ldots, K$, compute $\underline{rc}^k$ and $\overline{rc}^k$, as explained in Sections 3-5

Step 5: Compute the intermediate lower (7) and update the lower bound LB (8).

Step 6: If no cutting patterns could be generated in step 4, goto Step 8,
else add the generated cutting patterns to the master.

Step 7: If $LB < Z_{LP}^R$, goto Step 3.

Step 8: Use the rounding heuristic of Section 6 to produce a good feasible solution.

We have tested this procedure on a set of 12 industrial problems. They are representative of the type of instances for which this algorithm was developed. There are typically many customer orders but they concerns only a few different products. Thus, once the customer orders have been aggregated by types (order pieces sharing the same dimensions), one is left with only a few order types. In Table 5, we give the size of the

12 instances considered along with our computational results. The column headers read as follows: *pr* is the problem reference, n is the number of different order types, K is the number of stock pieces, *blocks* is the total number of blocks generated at the outset for all horizontal combinations, *CP* is the number of cutting patterns that were generated in solving the instance, *mast* is the number of resolution of the restricted master LP, *sp* is the total number of resolution of column generation subproblems for all stock pieces, LB is the lower bound we obtained, UB is the value of our best feasible solution, $Ogap$ is the optimality gap, $perW$ is the percentage of wasted material in our best feasible solution, *time* is the cpu time in seconds on our PC (Pentium MMX, 233Mhz), Tma is the percentage of this time spent in solving master LPs, and Tsp is the percentage spent in solving the pattern generation subproblems. The algorithm was implemented in C. The linear programs are solved using Xpress-mp callable library (Xpress-mp V10, 1997).

Table 5: Computational results for industrial problems on a PC (Pentium MMX, 233Mhz).

pr	n	K	blocks	CP	mast	sp	LB	UB	Ogap	perW	time	Tma	Tsp
p1	2	2	8	10	9	11	-1.63	5.82	3.93	3.07	1.07	37.7	8.4
p2	4	2	84	19	18	15	50.25	2877	88.70	23.5	2.55	32.1	42.0
p3	4	5	101	43	23	78	13.91	52.58	2.11	2.86	3.86	28.5	51.8
p4	6	3	151	27	15	34	-36.15	12.14	14.21	3.57	5.51	12.8	74.1
p5	7	3	22	45	120	271	-2340	1312	144	33.9	7.18	78.4	9.2
p6	5	4	199	128	79	173	90.73	281.1	2.02	2.98	21.0	18.6	76.7
p7	8	4	1743	33	12	40	-5.76	6.19	128.8	6.62	94.8	0.6	93.0
p8	16	5	2349	121	59	173	1.06	92.08	2.93	2.96	136	2.2	94.0
p9	11	4	1000	355	136	382	-2.89	43.13	5.21	2.52	340	2.4	96.8
p10	27	3	666	1683	875	2616	7.52	75.0	7.31	7.87	785	23.0	76.3
p11	16	12	2349	442	135	1278	-6.12	19.48	2.51	1.91	948	1.1	97.7
p12	17	12	2593	915	217	1782	11.58	144.5	3.47	2.33	2010	1.7	97.6

The objective function of our problem is quite peculiar as it is the difference between the value of the resources that are used and the value of the production. When values are equal to surface area and there are no fixed cost nor incurred penalties, this measure represent waste and should be close to zero. If the order value per unit of area are larger than those of the resources, this difference can be negative. In the presence of fixed costs and penalties, the objective value is greater than pure waste. In this context, it is difficult

to identify benchmarks in order to evaluate the quality of our solutions. However, we propose to use two criteria for this assessment: the optimality gap and the percentage waste. The former, denoted *Ogap*, is the difference between our upper bound and our lower bound divided by the value of the minimum required production and multiplied by 100. Indeed, it made no sense to divide this difference by an estimation of the optimal objective value (as it is usually done), since the latter is close to zero. Similarly, the percentage waste, denoted *perW*, is a measure of the waste as a percentage of the area of the required order pieces (minimum production).

Thus, *Ogap* is a measure of the optimality gap relative to the value of a minimal production: a gap of $x\%$ says that our solution might be away from the optimum by a value equal to $x\%$ of the minimum total order value. Comparing the optimality gap (resp. the waste) to the actual production value (resp. area) or the used stock piece value (resp. area) would have lead to smaller gaps (resp. percentage waste) but these benchmarks are not constant for the problem instance. For the instance $p9$, the percentage waste of 2.52% reduced to 1.88% if waste is compared to the area of the used stock pieces. Although the objective was not to minimise pure waste, the percentage waste measure is reported because it complements our assessment. Indeed, setting fixed costs or values may be a difficult exercise and somewhat arbitrary; so the percentage waste gives an idea of the quality of the solution, ignoring these settings.

In the results of Table 5, the optimality gap is less than 4% for half of the instances, giving us good confidence in the quality of those solutions. On the other hand, for a quarter of the instances, the optimality gap is so large that we have no means of assessing our solution. The explanation could be that we did not get a good Lagrangian lower bound because the column generation procedure was truncated too early or that our incumbent solution is poor. The results also indicates that the number of blocks does not blow up with the problem size, but the computational times do increase significantly with the number of stock pieces. On difficult problems, more than 90% of the time is spent in generating cutting patterns.

8 Extensions

There are several additional practical issues that have later been incorporated in our model and that required only minor adaptations to our solution approach. If there are identical stock pieces with the same dimensions, the RHS of constraints (5) represent the

number of copies of stock piece of type k and the master variables are integer variables not restricted to zero or one (then, in the rounding heuristic, we take the LP solution rounded down as an initial partial solution). If there are multiple cutting machines that differ in their characteristics (setup waste, maximum number of knife, etc), we generate patterns for each of them individually. (Moreover, machine capacities could be taken into account, by incorporating such constraints in the master). The stock piece might have defaults; each default results in an area of the stock piece that cannot be used to produce order pieces. Our approach can take into account large defaults that result either in a lost slit along the length of the stock piece or in a lost piece of width equal to that of the stock piece. In the former case, we generate horizontal combinations that can fit around the lost slits. In the case of a lost piece along the width, we generate sections up to the default area and then continue to generate sections beyond the default area in the heuristic that generates cutting patterns, while for the LP solution of the column generation subproblem, we simply account for the lost length.

Restriction (iv) that says in particular that a section should not involved more than 2 order types, is central to our solution approach. However, it can be relaxed slightly. Our treatment of the vertical combinations of order pieces remains valid for sections involving more than 2 order types provided that no more than 2 order lengths are involved (the restriction of not more than 1 order type per slit remaining in place). Thus we could work with horizontal combinations involving more than 2 order types but only two order lengths. In practical applications, one often finds products that share the same length but differ by their width; thus this extension is not insignificant.

The algorithm that we presented could be used at each node of a branch-and-bound tree for an exact resolution of our cutting problem. For the application that motivated this study, it did not seem necessary to strike for optimality given that the setting of values and fixed cost is somewhat arbitrary. Moreover, the response time of the optimiser was an important consideration. In any case, the rounding heuristic can be assimilated to a heuristic tree search (where only a few branches are explored). They are however several branching rules that would be well suited for this problem and are compatible with our solution approach. We would suggest enforcing the following rules in order. First, check that the order production is integer, else branch by redefining the production interval for the order concerned. Then, enforce that the total number of cutting pattern should be integer, branch by deciding whether a stock piece is used, branch on the number of cutting patterns involving x setups, or a return to stock, branch on the number of patterns producing a given product, or a given pair of product, or using a specific horizontal

combination. The dual variables associated with those branching constraints can all be taken into account in our approach to generating patterns.

9 Conclusions

This study is an attempt at bridging the gap between the exact optimisation of "pure" models that over-simplify cutting problems, on one hand, and the myopic heuristics commonly used for realistic models on the other hand. The algorithm presented here has served as a base for a industrial optimiser. Before it was introduced, a greedy procedure was used to solve the cutting problem manually: a good cutting pattern was constructed and used as many times as possible, then the same approach was reiterated on the residual problem. Compared to those solutions, the algorithm proposed here yielded small improvements on small (easy) instances but larger improvements for difficult instances. The order of magnitude of the some significant improvements can be given by two examples: a reduction from 5% waste to 2% was observed on an instance that was solved in 7 seconds, while, a reduction from 18% waste to 1.8% was observed on an instance that was solved in 4 minutes. Of course, the industrial objective is not just to minimise waste, but it is also to minimise handling (return to stock, knife changes, and the like), and maximize customer satisfaction. The flexibility build into our algorithm through the setting of the values and fixed costs was welcome to balance these different objectives of the business. However, finding good settings for this parameters required some expertise. Thus, such an algorithm remains only a tool to help in the construction of satisfactory solutions for the cutting problem on hand.

From a math programmer point of view, the approach developed for this problem is original in its use of nested column generation procedures. The underlying principle is simple: it is to take a difficult problem and to break it down to pieces that are tractable to solve, but the breaking-up (decomposition) must be done in such a way that the solutions to the pieces can be recombined into solutions for the original problem (through the column generation mechanism). Our approach can be seen as a heuristic based on the tools of exact optimisation. Compared to greedy heuristics that are traditionally used for cutting problems (construct a good pattern and use it as many times as feasible, then reiterate on the residual problem), it is not as myopic. To solve a master problem is to take a global view at the problem. Greedy heuristics have often attempted to be less myopic by adapting the values of orders/stock pieces, while this is done automatically

in a column generation approach through the dual prices. Compared to an approach based on the exact solution of a formulation that includes a selection of a priori generated patterns (that are believed to be "good" patterns), the dynamic generation used in our approach as obvious advantages. First, a set of a priori generated patterns might not even allow to generate a feasible solution, while a dynamic generation shall eventually generate the "missing" patterns. More importantly, patterns that have been generated a priori because they seemed to be "good" for the global problem are not necessarily "good" for the residual problem that results from fixing part of the solution (through branching or in the rounding heuristic), while the dynamic generation will provide patterns for the residual problem.

Acknowledgment
We thank Constantine Goulimis who described this problem to us, and Robert Weismantel who made his work on Hilbert bases available to us. We are also very grateful to the anonymous referees for their pertinent remarks that contributed to improve this presentation. The author was supported in part by a Management Research Fellowship granted by the British Economic and Social Research Council (ESRC).

References

Beasley, J.E. (1985). An algorithm for the two-dimensional assortment problem. *European Journal of Operations Research*, **19**, pp 253-261.

Gilmore, P.C. and R.E. Gomory (1961). A Linear Programming Approach to the Cutting Stock Problem, *Operations Research*, **9**, 849-859.

Gilmore, P.C. and R.E. Gomory (1963). A Linear Programming Approach to the Cutting Stock Problem: Part II, *Operations Research*, **11**, 863-888.

Gilmore, P.C. and R.E. Gomory (1965). Multistage cutting stock problems of two and more dimensions, *Operations Research*, **13**, 94-120.

Gilmore, P.C. and R.E. Gomory (1966). The theory and computation of knapsack functions, *Operations Research*, **14**, 1045-1074.

Johnson, E.L. and M. W. Padberg (1981). A Note on the Knapsack Problem With Special Ordered Sets. *Operations Research letters*, Vol **1**, no 1, pp 18-22.

Pisinger, D. (1999). Core Problems in Knapsack Algorithms, *Operations Research*, **47**, No 4, 570-575.

Riehme, J., G. Scheithauer, and J. Terno (1996). The solution of two-satge guillotine stock problems having extremely varying demands, *European Journal of Operational Research*, **91**, pp 543-552.

Valerio De Carvalho, J.M. and A.J. Guimaraes Rodriges (1994), A computer based interactive approach to a two-stage cutting stock problem. *INFOR*, Vol **32**, No 4, pp 243-252.

Valerio De Carvalho, J.M. and A.J. Guimaraes Rodriges (1995), A LP-based approach to a two-stage cutting stock problem. *European Journal of Operational Research*, **84**, pp 580-589.

Vanderbeck F. and L. A. Wolsey (1996). An Exact Algorithm for IP Column Generation, *Operations Research Letters* Vol. 19, No. 4, pp 151-159.

Vanderbeck, F. (1996). Computational Study of a Column Generation algorithm for Bin Packing and Cutting Stock problems, *Research Papers in Management Studies*, University of Cambridge, no 1996-14. Forthcoming in Mathematical Programming.

Weismantel, R. (1996). Hilbert Bases and the Facets of Special Knapsack Polytopes, *Mathematics of Operations Research*, Vol **21**, No 4, pp886-904.

Xpress-MP (1997): User guide and Reference Manual, Release 10, Dash Associates (http://www.dash.co.uk).