Column Generation and Sequential Heuristic Procedure for solving an Irregular Shape Cutting Stock Problem

Xiang Song
Logistics and management mathematics group, Department of mathematics, University of Portsmouth, Lion Gate building, Lion Terrace, Portsmouth, P01 3HF, UK {xiang.song@port.ac.uk}

Julia A. Bennell
School of Management/CORMSIS, University of Southampton, Southampton, SO17 1BJ, UK, J.A.Bennell@soton.ac.uk

Research addressing two-dimensional irregular shape packing has largely focused on the strip packing variant of the problem. However, it can be argued that this is a simplification. Materials from which pieces are required to be cut will ultimately have a fixed length either due to the physical dimensions of the material or through constraints on the cutting machinery. Hence, in order to cut all the pieces, multiple sheets may be required. From this scenario arises the two-dimensional irregular shape cutting stock problem. In this paper, we will present implementations of cutting stock approaches adapted to handle irregular shapes, including two approaches based on column generation and a sequential heuristic procedure. In many applications, set up costs can be reduced if the same pattern layout is cut from multiple sheets; hence there is a trade-off between material waste and number of patterns. Therefore, we describe the formulation and implementation of an adaptation of the column generation method to control the number of different patterns. Column generation is a common method for the cutting stock problem, however when the pieces are irregular the sub problem cannot be solved optimally. Hence we implement column generation and solve the sub problem using the beam search heuristic. Further, we introduce a version of column generation for instances where the number of rows is less than the number of columns.

Keywords: Column Generation; cutting and packing; heuristic; irregular.

---

1. Introduction

In this paper, we consider a variant of the two-dimensional (2D) cutting stock problem where the pieces to be cut are of irregular shape. Customer orders are composed of a variety of different shapes in various quantities. In our implementation, we only consider one stock sheet size, however, some applications may have multiple stock sheet sizes or use off-cuts of partially cut sheets, provided the off-cut is sufficiently large. The manufacturer needs to design a number of cutting patterns to satisfy customer demand while keeping the operating cost at an acceptable level. Operating costs include minimising the number of standard rectangular stock sheets in the solution and number of setups, where a setup is a change of pattern. Hence, the problem is a pattern restricted 2D irregular cutting stock problem. According to the typology proposed by Wäscher et al. (2007), this problem is a Single Stock Size Cutting Stock Problem (SSSCSP) with the refinement of irregular shapes. Very few papers solve this problem and those that do reduce the problem to a one-dimensional cutting stock problem (1DCSP) (Degraeve and Vandebroek (1998); Martens (2004); Rose and Shier (2007)). They first construct patterns of subsets of irregular pieces (stencils) that span the width of the stock sheet. The stencils form input pieces for the 1DCSP. Hence, this paper is distinctive in directly tackling the layout of irregular shapes across multiple stock sheets.

Theoretically, 1D and 2D cutting stock problems are NP-hard as shown in Garey and Johnson (1979). Thus, it becomes impractical to solve reasonable size problem to optimality. Researchers frequently adopt approximate procedures such as linear programming based heuristic methods, bespoke heuristic procedure, and metaheuristics. For a detailed survey see Dyckhoff (1990), Haessler and Sweeney (1991), Dowsland and Dowsland (1992) and Lodi et al. (2002). Given the extensive literature on cutting stock problems and the specialist nature of the problem we are proposing to tackle, our discussion of the literature will focus on problems with multiple-objectives and/or multiple-constraints.

Although linear programming (LP) may not be sufficiently flexible to consistently solve problems with multiple-objectives or with various additional restrictions that arise in practice, there are still many papers solving this kind of problem with LP-based approaches for one-dimensional problems (Belov and Scheithauer (2007); Diegel et al. (1996); Dyckhoff (1981); Gilmore and Gomory (1961, 1963); Haessler (1980); Valério and Rodrigues (1995); Vanderbeck (2000); Wäscher (1990); Zak (2002)). Gilmore and Gomory (1961, 1963) worked on the 1DCSP and provided the foundation for LP approaches for cutting stock problems. Haessler (1980) improved on their work by placing restrictions on the number of times a given ordered item can appear in a pattern. This resulted in LP solutions with the same trim loss values and fewer patterns. Dyckhoff (1981) developed another LP model that improved on Gilmore and Gomory (1961, 1963) approach when there were many items to be cut. Diegel et al. (1996) examined setup minimising conditions in the trim loss problem in the paper industry. They formulate an LP with constraints to guarantee the least setups. Valério and Rodrigues (1995) considered minimising setups and trim loss in the metal cutting industry. The cutting process is multi-stage and produces intermediate rolls, which must also be minimised. The multi-stage cutting stock problem also appears in the paper industry. Zak (2002) followed a row and column generation approach (CG) where the rows are the intermediate roll sizes and the columns the cutting pattern. Vanderbeck (2000) considered a variant of 1DCSP called the pattern minimisation problem (PMP) that minimizes the number of different cutting patterns while also placing an upper bound on the number of stock sheets available. Their IP formulation involves a large number of binary variables and associated columns. They propose an exact branch-and-price algorithm using a column generation approach, where the sub-problem is a non-linear IP that can be decomposed into a number of bounded knapsack problems. Belov and Scheithauer (2007) considered a different formulation for PMP. They use a smaller number of variables than that of Vanderbeck's formulation, and developed another exact branch-and-price algorithm.

Papers that address multi-objective, multi-constrained one-dimensional cutting stock problem (1DCSP) using problem specific mathematical models and solve them using a heuristic include (Antoni et al. (1999); Burkard and Zelle (2003); Chu and Antonio (1999); Coverdale and Wharton (1976); Haessler (1971, 1975); Haessler and Sweeney (1991); Kasimbeyli et al. (2011); McDiarmid (1999); Nonås and Thorstenson (2000, 2008); Sinuany-Stern and Weiner (1994); Song et al. (2006)). The Sequential Heuristic Procedure (SHP) is the most commonly applied. Haessler (1971) described the first documented SHP capable of finding better solutions than those found manually by schedulers. In Haessler (1975), the cutting stock problem is formulated to include a fixed cost for changing cutting pattern in addition to trim loss. The SHP sets goals for certain characteristics of the pattern in order for it to enter the solution. When no patterns satisfy these goals, some of the goals are relaxed. Coverdale and Wharton (1976) also extended the work of Haessler (1971) to consider factors such as set up time for the machine operators. Pure SHP has been shown to increase the trim loss because of so-called ending conditions (Haessler and Sweeney (1991)). Kasimbeyli et al. (2011) set up a two-objective mathematical model without cutting patterns for one-dimensional assortment problems and proposed a special heuristic algorithm to tackle the presented model.

A number of papers combine LP-based algorithms and heuristic techniques to obtain high quality solutions to these types of cutting and packing problems (Foerster and Wäscher (2000); Furini et al. (2012); Goulimis (1990); Gradišar et al. (1999); Gramani et al. (2009); Hendry et al. (1996); Richard (1996); Sweeney and Haessler (1990); Umetani et al. (2006)). Sweeney and Haessler (1990) combined LP within a two stage sequential heuristic to develop a procedure for solving a 1DCSP with quality variations across the width of the stock rolls. Goulimis (1990) used an SHP algorithm to generate a starting solution for the LP procedure to solve a 1DCSP in a board mill. Hendry et al. (1996) presented a two-stage solution procedure for a combined cutting-stock and lot-sizing problem from the copper industry. Richard (1996) reported on a randomized approach to the 1DCSP. Gradišar et al. (1999) examined a hybrid approach for optimising one-dimensional cutting stock. Foerster and Wäscher (2000) proposed a pattern combination heuristic algorithm called KOMBI, which starts from a solution obtained by LP, and combines patterns while keeping the number of stock rolls the same. Umetani et al. (2006) considered a variant of 1DCSP, called the pattern restricted problem (PRP), where the objective is to minimize the number of stock rolls while constraining the number of different cutting patterns within a user defined bound. They developed local search algorithms. As the neighbourhood size plays a crucial role in determining the efficiency of local search, they employ linear programming techniques to restrict the number of solutions in the neighbourhood. Gramani et al. (2009) set up a mathematical model for a coupled lot-sizing and cutting stock problem and proposed a heuristic method based on Lagrangian relaxation. Furini et al. (2012) tackled a two-dimensional rectangular guillotine cutting problem with stock of different sizes with a column generation based heuristic algorithm.

The literature discussed above focuses on multi-objective or multi-constrained 1DSCP. Our research has found very few papers solving the two-dimensional case. One exception is the paper of Imahori et al. (2005) who used local search to solve the pattern restricted 2DCSP, in which the pieces to be cut are all rectangles. In this research, we are interested in the irregular shape cutting stock problem and, to our knowledge, there are only three papers found dealing with this variant, all arising from the apparel industry (Degraeve and Vandebroek (1998); Martens (2004); Rose and Shier (2007)). All three treated the problem as a 1DCSP by constructing the patterns, called a stencil, for each size of garment separately. Degraeve and Vandebroek (1998) solved the transformed problem with mixed integer programming and Martens (2004) used genetic algorithms. The definition of the problem by Rose and Shier (2007) included a number of assumptions and practical considerations. As a result, an exact enumerative approach can identify all optimal solutions to the defined problem. In each case, the authors of these methods admit that the layout obtained by constructing the stencils for each size of garment separately is not very efficient.

In this paper, we consider a generic irregular cutting stock problem, where most of the ordered items have irregular shape. In addition to minimising trim loss, we want to use as few patterns as possible in order to reduce setup costs. As a result, we set an upper limit, n, on the number of patterns in the solution, where n is smaller than the number of item types. To solve this problem, we design and evaluate three solution approaches; Column Generation (CG), an Adapted Column Generation (ACG) algorithm and a Sequential Heuristic Procedure (SHP). Both our CG and ACG algorithms follow the traditional CG approach in decomposing the master problem into a series of sub-problems and improving the master problem by solving the corresponding sub-problems. The CG approach applies the simplex method to basic feasible solution matrices with the same number of rows and columns, where the number of rows equals the number of setups and the number of columns equals the number of piece types. When the problem definition restricts the number of setups to be less than or equal to a given number n, which is far smaller than the number of piece types, the basic feasible solution matrices will have more rows than columns. Hence, we design our ACG approach, which is an adaption of the simplex method in CG scheme, to deal with such matrices. The SHP is an adaptation of Haessler (1975) work. The procedure generates patterns that satisfy certain characteristics and uses them sequentially until all the requirements are met.

The contribution of this paper includes directly solving the irregular cutting stock problem, without pre-constructing stencils, applying Beam Search (BS) algorithm (Bennell and Song (2010)) within CG, and developing the CG methodology to deal with matrices that have more rows than columns. Each approach implemented includes methodology innovations to either make the approach capable of handing the irregular shape packing problem directly, or deal with the additional set up constraint. Hence the paper includes contributions to methodology and to the application of combinatorial optimisation in cutting and packing, as well as introducing benchmark instances.

The paper is organized as follows; in the next section, we give a mathematical description of the problem. In section 3, we describe our implementation of column generation for the irregular cutting stock problem, including the BS methodology for generating patterns. We describe the ACG approach in section 4 and the SHP approach in section 5 and give the corresponding algorithm for both. In section 6 we discuss computational experiments for some problem instances, which are modified from the benchmark problems from ESICUP (www.fe.up.pt/esicup) to fit our problem type.

2. Problem description

The problem considered can be formulated as follows: let b_i, $i = 1, 2, \dots, m$, be a finite number of ordered items of type i, where m is the number of different items, and let there be a theoretically infinite set of stock sheets of a given length L and width W. Let Π be a set containing all feasible pattern layouts. Since each feasible pattern layout can be mapped into a feasible cutting pattern column, we work with feasible cutting pattern columns only. Note, different layouts may be mapped into the same column. Thus, we define $\mathbf{A}$ as a vector set containing all feasible cutting pattern columns. The problem becomes: Given a set of feasible cutting pattern columns, $\mathbf{a}_j = (a_{1j}, a_{2j}, \dots, a_{mj})^T \in \mathbf{A} \neq \phi$, $j = 1, 2, \dots,$ where non-negative integer a_{ij} is the number of times ordered item i is cut in pattern $\mathbf{a}_j$, find x_j that minimises the number of stock sheets required to fulfil all ordered items, where x_j is the number of times the solution uses each feasible pattern. The number of patterns in the solution must be less than or equal to n, hence, only n variables may be non-zero and positive. The mathematical description of this problem is as follows,

$$
\begin{aligned}
\text{P: Minimize} \quad & \sum_{j\in J} x_j \\
\text{subject to} \quad & \sum_{j\in J} a_{ij}x_j \geq b_i \qquad i \in I && (1)\\
& \sum_{j\in J} y_j \leq n && (2)\\
& x_j \leq M \cdot y_j \qquad j \in J && (3)\\
& x_j \geq 0 \qquad \textit{integer},\ j \in J && (4)\\
& y_j \in \{0,1\} \qquad j \in J && (5)
\end{aligned}
$$

Where $I = \{1, 2, \ldots, m\}$ is the index set of ordered irregular shape items, $J = \{1, 2, \ldots\}$ is the index set of all feasible cutting pattern columns, non-negative M is an unlimited big number, and y_j is a binary variable that takes value 1 if pattern $\mathbf{a}_j$ is used and zero otherwise. Constraint (1) requires that the customers' demands be met. Constraint (2) ensures that only n patterns are used. Constraint (3) provides the proper definition of variable y_j.

Set $\mathbf{A}$ can be huge and this problem is NP-hard, since its simplification is a generalization of the Bin Packing Problem (BPP), which is know to be strongly NP-hard Garey and Johnson (1979).

3. Column Generation (CG) variant for the irregular shape 2DCSP

In this section, we describe the column generation scheme for the irregular shape cutting stock problem. Since the sub-problem cannot be solved using the traditional optimization techniques like dynamic programming or branch and bound, we propose a beam search heuristic to generate feasible layouts based on Bennell and Song (2010). This section includes details of the algorithmic procedure for this step.

3.1 Column generation scheme

In order to apply column generation, we need to convert problem P (see Section 2) into a linear programming formulation by relaxing the integer requirement for x_j, $j \in J$ and simplify the problem P by setting $n \to \infty$. The LP is as follows.

$$
\begin{aligned}
\text{LP: Minimize} \quad & \sum_{j\in J} x_j & & \\
\text{subject to} \quad & \sum_{j\in J} a_{ij}x_j \geq b_i & i \in I & \qquad (6)\\
& x_j \geq 0 & j \in J & \qquad (7)
\end{aligned}
$$

This LP problem is the *master problem* (MP) (Lübbecke and Desrosiers (2005)). To solve this problem using the simplex method, we look for a non-basic variable to price out and enter the basis. That is, given the dual multiplier vector $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, we wish to find $\arg\min\{\overline{c}_j := c_j - \pi \cdot \mathbf{a_j} | j \in J\}$. In our LP problem, $c_j = 1$, for all $j \in J$. $|J|$ may be too large to make a comprehensive search in reasonable computational time, hence it is common to work with a small subset of columns, called the *restricted master problem* (RMP) (Lübbecke and Desrosiers (2005)). The size of the small subset of columns in RMP should equal the number of elements in a feasible cutting pattern column $\mathbf{a_j}, j \in J$, which is m. Thus, let $\overline{J} = \{1, 2, \ldots m\}$ be the index set of the small subset of columns in RMP, $\overline{\mathbf{x}}$ and $\overline{\mathbf{a}}$ be primal and dual optimal solutions of the RMP, respectively. Since the cost coefficient $c_j = 1$ for all $\mathbf{a}_j$, the reduced cost coefficient $\overline{c}$ of at least one feasible cutting pattern $\mathbf{a} = (a_1, a_2, \ldots, a_m)^T \in \mathbf{A}$ must satisfy inequality: $\overline{c} = 1 - \pi \cdot \mathbf{a} < 0$, otherwise $\overline{\mathbf{x}}$ optimally solves the LP as well, where $a_i (i \in I)$ is the decision variable: the number of times item i is selected in the feasible cutting pattern. Thus, to tell if $\overline{\mathbf{x}}$ optimally solves the LP, the following equivalent knapsack problem (KP) with geometric constraints needs to be solved to optimality:

$$
\begin{aligned}
\text{KP: Maximize} \quad & \pi \cdot \mathbf{a} & & \\
\text{subject to} \quad & \mathbf{a} = (a_1, a_2, \ldots, a_m)^T \; is\ a\ feasible\ cutting\ pattern & & \qquad (8)\\
& a_i \geq 0 \qquad integer, & i \in I & \qquad (9)
\end{aligned}
$$

Here the bound on the maximum number of times a piece may appear in the solution is unconstrained. If the solution of the KP is larger than 1, then the LP is improved by replacing the associated column to the RMP and updating the feasible basis. We continue to solve sub-problems iteratively until the solution of the knapsack problem is less than or equal to 1. That is $\overline{c}^* = \min\{1 - \pi \cdot \mathbf{a} | \mathbf{a} \in \mathbf{A}\} \geq 0$, then we have the optimal solution $\overline{\mathbf{x}}$ for the LP.

In our CG approach, the sub-problem KP is to generate feasible 2D irregular pattern layouts, which can only be done reasonably using a heuristic. As a result, the KP cannot be guaranteed to be solved to optimality and the solution found from the above procedure is unlikely to be the optimal solution for the LP. We use $\widetilde{\mathbf{x}}$ to represent the best solution found for the LP.

Clearly, an important component of our CG based heuristic approach is an efficient algorithm for solving the KP sub-problem. We describe a beam search algorithm based on Bennell and Song (2010) to find near optimal solution for this problem in the next section. Note that, the elements $x_1, x_2, \cdots, x_m$ in solution vector $\widetilde{\mathbf{x}}$ found by our CG are not generally integral. The quality of the integer solution will depend on the algorithm for solving the sub-problem, the rounding procedure, and the structure of the demands. The method we adopt is as follows; round down all non-integer $x_j, j \in \overline{J}$ leaving a residual problem of unsatisfied demand defined by $r_i = b_i - \sum_{j \in \overline{J}} a_{ij} \cdot \lfloor x_j \rfloor, i \in I$. Solve the residual problem greedily using SHP, where the solution is described by a set of cutting patterns $\mathbf{a}_r = \{a_{r_1}, a_{r_2}, \dots\}$ and the frequency of the patterns $\mathbf{x}_r = \{x_{r_1}, x_{r_2}, \dots\}$. We describe the SHP in section 5.

3.2 Solving the sub-problem using the Beam Search (BS) heuristic

Bennell and Song (2010) demonstrated that beam search is an effective approach for solving the irregular strip packing problem and has the advantage of generating many alternative solutions in parallel. First, we introduce the main steps of the beam search algorithm for irregular packing and then describe our implementation for the KP sub-problem.

The heuristic takes the form of constructing a pattern layout piece by piece where the selection of the next piece is dynamic. BS allows us to explore a number of alternative dynamically generated packing orders. It is analogous to branch and bound where the tree is searched breadth first and aggressively pruned at each level. There is no backtracking and the user defines the maximum number of branches. The approach uses local or global evaluation functions to select the most promising nodes to branch from in the next level. A global evaluation will be accurate but may be computationally expensive, whereas a local (approximate) evaluation can be fast but carries the risk of discarding branches that lead to good solutions.

In our BS implementation, the search tree represents the packing order of the pieces on the stock sheet. Each node in the search tree corresponds to a partial solution and a branch from a node represents the decision to add a piece to generate the next partial solution. The basic idea of BS implementation is given as follows:

Step 1: Select a piece as the as the initial partial solution forming the root node of the search tree.

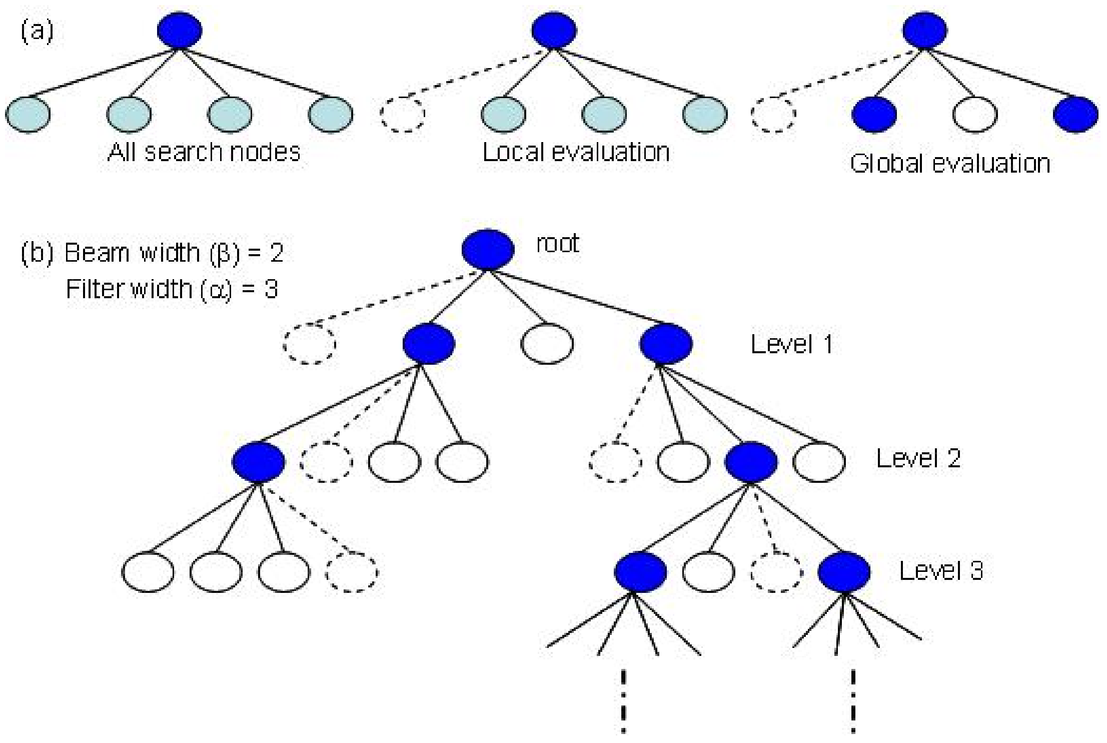

Figure 1: (a) each of the beam search evaluation steps at each level, (b) illustration of an expanded beam search tree.

Step 2: At each level,

Step 2.1: A local evaluation function evaluates all child nodes; which provides a crude approximation of solution quality by measuring the incremental cost of adding a piece to the partial solution. Using this evaluation, prune the tree by retaining the best α nodes, where α is the *filter width*.

Step 2.2: Apply the global evaluation function to these nodes from Step 2.1, which evaluates a complete packing order, and retain the best β nodes for branching, where β is the *beam width*.

Step 3: Repeat step 2 until the packing order is complete.

The nodes at the lowest level of the tree represent a complete pattern. Figure 1a illustrates the pruning process, where the local evaluation reduces the four child nodes to three ($\alpha = 3$), and the global evaluation reduces the three retained nodes to two ($\beta = 2$). Figure 1b illustrates an expanded tree.

The local evaluation function must perform two sets of evaluations. It must determine the best position for each piece type in the partial solution (*placement evaluation*), which become the child

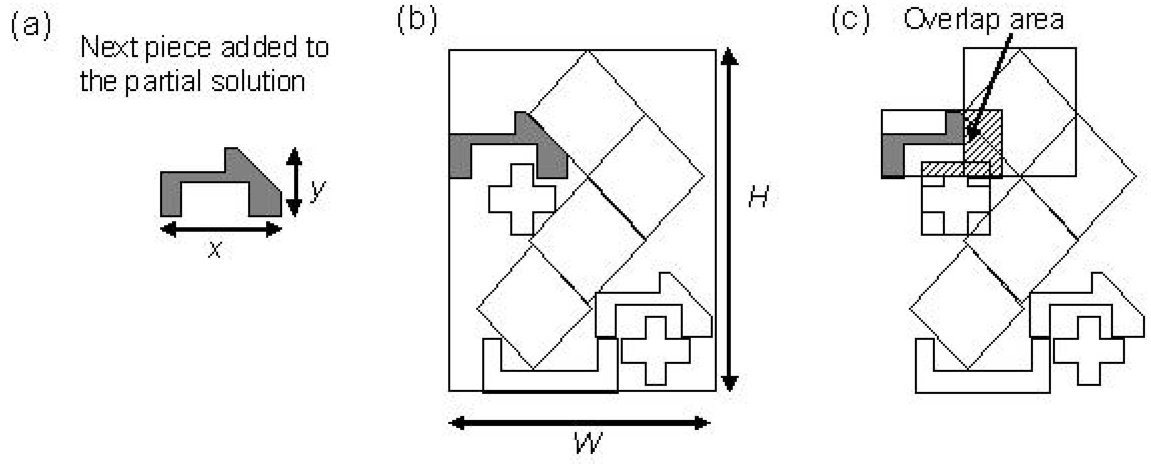

Figure 2: Illustration of variables used to formulate criteria for construction heuristic

nodes, and determine the best partial solutions (*next piece evaluation*), which are the child nodes that go forward for global evaluation. We use the improved TOPOS approach of Bennell and Song (2010), which builds on the work of Oliveira et al. (2000). In both cases, the first piece has an orientation but no fixed position; this is the first partial solution. The pattern layout grows one piece at a time. In order to add a piece, first identify the best position for each piece type in all orientations. There exists a finite number of positions that will optimise the *placement evaluation.* Given this set of alternative partial solutions, the *next piece evaluation* identifies the nodes that go on to global evaluation. Note that the placement evaluation and the next piece evaluation can be identical or different. In this research, they are different. See Bennell and Song (2010) for a full analysis of evaluation approaches.

Bennell and Song (2010) identify three *attributes* of a partial solution that are useful in measuring its quality. We adopt all three for the placement evaluation. Figure 2 illustrates these attributes. These are: rectangular enclosure, $R = \frac{W \times H}{x \times y}$; packing length, $L = \frac{W}{x}$; and overlap, $O = \frac{overlap}{x \times y}$, where x and y are the maximum width and height respectively of the piece to be added , W and H are the width and height of the enclosing rectangle of the partial solution after the pieces is added, and *overlap* is the area of overlap between the enclosing rectangles of the piece added and the pieces in the partial solution. Since we are using BS to solve the KP in this case, the dual price of the piece is more important than the physical properties of the partial solution when it comes to selecting which piece to place next. Hence, we define a new attribute $Pi = \frac{\pi}{x \times y}$, where π is the dual price of the next piece. We use Pi, R and O for the next piece evaluation, replacing L since this is less important for multi-stock sheet problems.

The attributes (R, L, O) for placement evaluation and (Pi, R, O) for next piece evaluation are aggregated using a priority strategy, which is a lexicographic preference. For example, if we used Pi, R and O, in that priority order, then Pi is the primary objective, R is the secondary objective considered when there are multiple solutions with the same maximum Pi, and O is considered when there are ties in both Pi and R.

We find the global evaluation by continuing to pack greedily all remaining pieces after the child node under consideration. At each level, the greedy decision is to select the node with the best local evaluation function. The value of the global evaluation is the sum of the dual price of the pieces in the order they are packed.

Our beam search implementation can be summarised by the below procedure, in which we construct β parallel patterns, $a_s (s = 1, 2, \ldots, \beta)$, piece by piece. Since the stock sheet has fixed dimensions, the numbers of pieces in each complete solution vary. Hence, let $T_s (s = 1, 2, \ldots, \beta)$ be the total number of pieces packed in a feasible pattern a_s. Let m be the total number of piece types including alternative orientations. Let PS_t be a partial solution containing t nested pieces, and let S_t and m_t be the set of remaining pieces and piece types respectively to be nested at stage t, corresponding to a given PS_t. Let $GLOBAL(PS_{t+1}, t+1, T_s)$ be the global evaluation function that returns the sum of the dual price of the feasible pattern obtained using the improved TOPOS strategy, where the piece order of the first $t+1$ pieces in the partial solution PS_{t+1} have been determined by the search tree and only the following $(T_s - t - 1)$ unpacked pieces need to be ordered using TOPOS. Let $LOCAL(PS_{t+1}, t+1)$ be the local evaluation function that returns the value determined by TOPOS strategy for the next piece $(t+1)$.

Step 0: Generation of PS_1. Let $t = 1$. If $m < \beta$, then use all m piece types (including rotations) as root nodes, go to step 1. Otherwise, sort the piece types in descending order of dual price and use the first β piece types in the ordered list as root nodes and initial beam nodes. For each beam initialise S_t and m_t, go to step 3.

Step 1: If the total number of nodes $< \beta$, then for all PS_t move to next level by adding each piece type i (for all $i \in m_t$) to the corresponding PS_t, $t = t + 1$, repeat step 1. Otherwise, go to Step 2.

Step 2: Select initial beam nodes. Compute the global evaluation function values for all the nodes with $GLOBAL(PS_{t+1}, t+1, T_s)$ and select the best β nodes as initial beam nodes.

Step 3: Child node generation. For all PS_t move to next level by adding piece type i (for all $i \in m_t$) to the corresponding PS_t, Step 4: Filtering. Compute $LOCAL(PS_{t+1}, t+1)$ for each child node. For each parent node select the best α child nodes and prune all remaining nodes.

Step 5: Select beam nodes. Compute $GLOBAL(PS_{t+1}, t+1, T_s)$ for selected child nodes. Select the best β nodes out of all child nodes as beam nodes.

Step 6: Update sets. For each selected beam node, set PS_{t+1} by adding the beam node (piece j) to PS_t. Remove piece j from S_t, update m_t, $t = t+1$ and T_s.

Step 7: Termination test. If $t < \min\{T_s | s = 1, 2, \ldots, \beta\}$, go to step 3.

Step 8: Finish all the β solutions using TOPOS strategy. Among the β solutions, select the one with maximum sum of the dual price of the pieces packed.

The values of T_s are dependant on the previous packing decisions so can only be estimated by each global evaluation; hence, we update T_s in step 6. Note that we terminate the BS Tree at the minimum T_s. The other $\beta - 1$ layouts are completed greedily using the TOPOS strategy. Clearly, continuing the search to the lowest level for all β solutions could yield better solutions. However, our initial experiments indicate that the improvement does not warrant the significant increase in computational effort.

4. Adapted Column Generation to control the number of patterns

In general the application of CG to the CSP does not restrict the maximum number of setups or patterns, and can apply the simplex method to basic feasible solution matrices that have the same number of rows and columns. In our application, the number of patterns is restricted to $n < m$, requiring a basic feasible solution that has fewer non-zero decision variables than rows. In order to force the restricted number of non-zero decision variables, we reduce the number of columns in the basic feasible solution matrix and present an adaption of the simplex method to deal with such matrices, called Adapted Column Generation (ACG). The ACG algorithm works with basic feasible solution matrices $A_{m\times t} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_t\}$, which have more rows than columns ($m > n \geq t \geq n_{LB}$), where n_{LB} is a lower bound on the number of cutting patterns required (see section 4.2 for its derivation), and $a_1, a_2, \ldots, a_t$ are a set of feasible patterns from J. This problem is called *the restricted cutting stock problem.*

Clearly, the solution to this restricted cutting stock problem is also a feasible solution for the original problem P. The solution is iteratively improved by substituting or adding, provided $t < n$, the new cutting pattern into the initial solution matrix until no further improvement can be found. In this section, we discuss the following implementation issues: how to generate an initial solution, how to construct the new cutting pattern to improve the original solution, and how to adapt the column generation method to solve the problem directly.

4.1 Outline of the ACG algorithm

Adapted column generation adopts the principles of classic column generation (CG) while adapting certain aspects of the approach where the problem does not meet the necessary conditions. Specifically, our problem formulation does not allow the construction of basic feasible solution that has the same number of non-zero decision variables as rows. Since the number of setups and therefore patterns are restricted to be less than or equal to n, which is much smaller than the number of piece types, m. As a result, we are dealing with basic feasible solution matrices that have more rows than columns. Like classic CG, we decompose the master problem into a series of sub-problems and improve the master problem by solving the corresponding sub-problems.

Classic CG defines the dual price π as $\pi = c_B \cdot B^{-1}$, where B is the basic solution matrix, which is a square matrix with the rank m. Given that a non-square matrix does not have an inverse, we adopt the idea of the pseudoinverse or generalized inverse matrix, A^+, calculated by $A^+ = (A' \cdot A)^{-1} \cdot A'$ (Penrose (1955)). Hence an approximation of the dual price is $\hat{\pi} = c_A \cdot A^+$. The generalised inverse has the property that the solution vector $\mathbf{x} = A^+ \cdot \mathbf{b}$ minimizes the Euclidean length of the vector $\mathbf{e}$, where $\mathbf{e} = A \cdot \mathbf{x} - \mathbf{b}$, in which $A \cdot \mathbf{x}$ gives the calculated numbers of times that the items are cutted out and vector $\mathbf{b}$ gives the actual customers' demands. Since a true inverse B^{-1} would give $|\mathbf{e}| = 0$, but is unobtainable, then the minimum $|\mathbf{e}|$ is desirable.

From the above analysis and analogous to CG algorithm, $\max \sum_{i \in I} (\hat{\pi}_i) \cdot a_i$ is one objective of the sub-problem when using ACG algorithm, where a_i $(i \in I)$ is the number of times piece i is packed in the new feasible pattern layout.

Since $|\mathbf{e}| \neq 0$ in ACG algorithm, we introduce the following measure of deviation; $\mathbf{d} = \frac{\mathbf{e}}{|\mathbf{e}|} = (d_1, d_2, \ldots, d_m)$, where each element d_i $(i \in I)$ gives us the weighted difference between the calculated number of item i to cut out and the actual customer's demand. If more items are cut out than necessary, more wastage will be incurred. Thus we also wish to minimize the sum of the deviation of the pieces in the new feasible pattern layout. As a result, we derive a weighted objective function for the sub-problem: $\max \sum_{i \in I} (\hat{\pi}_i - d_i) \cdot a_i$, where $a_i (i \in I)$ is the number of times piece i is packed in the new feasible pattern layout.

Since we allow fewer patterns than piece types, it is possible to generate cutting patterns

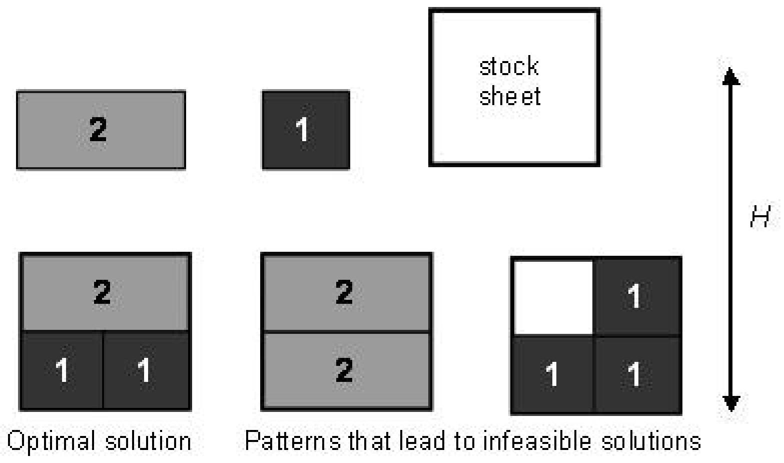

Figure 3: Simple example to illustrate that some patterns lead to infeasible solutions

that result in infeasible solutions. The simple example in figure 3 illustrates this. There are two rectangular piece types; the first has both width and height one, the second has width two and height one. The demand is five and ten respectively, the stock sheet has width and height two, and the maximum number of patterns is one. It is clear that the optimal solution uses five copies of a pattern that contains one of piece type one and two of piece type two. Patterns that contain two of piece type one or more than two of piece type two would result in unsatisfied demand or the need for another pattern, both of which are infeasible.

As a result, we need to set a good upper bound on the number of times a piece can appear in a cutting pattern and add the constraint, $a_i \leq UB_i (i \in I)$. Any bound must ensure that every piece type can appear at least once across all the permitted patterns. A good upper bound should take account of the relative demand of the pieces to allow the higher demand pieces to appear more frequently. The following procedure for generating the upper bound UB_i crudely evaluates whether the pieces will fit within the permitted patterns, and if not, considers patterns will be cut multiple times:

Initialization: Let $c = 1$ and s_i denote the area of piece type $i \in I$.

Step 1: Calculate $UB_i = \lceil \frac{b_i}{c} \rceil (i \in I)$ and $S = \sum_{i=0}^{m} s_i \cdot UB_i$.

Step 2: If $\lceil \frac{S}{L \times W} \rceil \leq n_{LB}$, return $UB_i(i \in I)$, else, set $c = c + 1$, Return to step 1.

4.2 Construction of an initial solution

It is straightforward to construct the initial basic solution matrix for the LP problem described in section 3.1. This problem has no restriction on the number of patterns nor on the number of times a piece appears on the cutting pattern. Hence, there are in total m cutting patterns, where each cutting pattern contains only one type of piece.

To generate an initial solution to the pattern restricted CSP, we start with the minimum number of cutting patterns, n_{LB} required to meet demand. A simple lower bound would be $n_{LB} = \lceil \sum_{i=1}^{m} s_i / L \cdot W \rceil$. This is not a tight lower bound, since it is rare for the shapes to fit together with close to zero waste. Instead, we pack one of each type of piece using the following procedure to minimise the number of the stock sheets.

Initialization: Set $n_{min} = 1$ as the minimum number of patterns used. Let $\overline{b}_i = 1$ be the demand of the piece $i \in I$ and $\overline{A} = \phi$ be the set of cutting patterns.

Step 1: Construct a feasible cutting pattern $\overline{a} = \{\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_m\}$, where $\overline{a}_i$ equals 1 if piece i has been packed, 0 otherwise, using the beam search algorithm described in Section 3.2, where the global evaluation function returns the total area of the pieces packed and the local evaluation uses Priority(R, L, O) for both placement evaluation and next piece evaluation, which was called *one-step* strategy in Bennell and Song (2010).

Step 2: Reduce the demand $\overline{b}_i = \overline{b}_i - \overline{a}_i$ and add the feasible cutting pattern $\overline{a}$ to $\overline{A}$, set $\overline{A} = \overline{A} + \overline{a}$. If $\sum_{i=1}^{m} \overline{b}_i = 0$, go to step 3, else, set $n_{min} = n_{min} + 1$, go to step 1.

Step 3: Set $n_{LB} = n_{min}$. Stop.

The above procedure gives a tighter lower bound n_{LB} on the number of patterns and provides the initial solution matrix $A_{m \times n_{LB}}$ for the ACG procedure.

4.3 Solve the knapsack problem

The following knapsack problem arises from the discussion in Section 4.1:

$$KP^{+}: \text{Maximize} \quad \sum_{i \in I} (\hat{\pi}_i - d_i) \cdot a_i$$

$$\text{subject to} \qquad a = (a_1, a_2, \ldots, a_m)^T \ is\ a\ feasible\ cutting\ pattern \tag{10}$$

$$UB_i \geq a_i \geq 0 \qquad integer, \qquad i \in I \tag{11}$$

We solve this KP^{+} with the beam search algorithm (see section 3.2). However, since we only have the approximate dual, the local evaluation uses an alternative criterion. Let $\hat{\pi}$ be the approximate dual price and d be the deviation of the next piece to pack, as defined in section 4.1, then the local evaluation function is priority($\frac{\hat{\pi}_i - d_i}{x \cdot y}$, R, O). The global evaluation function is the same as the KP^+ objective function; sum of the approximate dual price minus sum of the deviation of the pieces packed. The upper bound on the number of pieces packed in constraint (11) has been given in Section 4.1.

4.4 The complete ACG procedure

Initialization: Generate initial solution matrix that has more rows than columns $A_{m\times n_t}$ in a greedy way (as described in section 4.2), where n_t denotes the temporary number of patterns and $n_t < n$. Initially, we set $n_t = n_{LB}$.

Step 1: Calculate $\hat{\pi} = c_{n_t} \cdot A^+_{m\times n_t}$ and $d = \frac{e}{|e|}$, where $e = b - A \cdot x$. Solve the KP^+ with beam search algorithm (as described in section 3.2 and 4.3), where the beam width is set to β, to obtain β feasible cutting pattern $\mathbf{a}_k, (k = 1, 2, \ldots, \beta)$.

Step 2: Substitute each $a_k, (k = 1, 2, \ldots, \beta)$ with each column in $a_j, (j = 1, 2, \ldots, n_t)$ and solve the following integer programming problem P^+

$$\begin{array}{llll} \text{P}^+\text{: Minimize} & \sum_{j=1}^{n_t} x_j & & \\ \text{subject to} & \sum_{j=1}^{n_t} a_{ij}x_j \geq b_i & i \in I & (12) \\ & x_j \geq 0 & \textit{integer},\ j = 1, 2, \ldots, n_t & (13) \end{array}$$

Determine the best substitution a_{k^*} and a_{r^*}, which gives minimum $\sum_{j=1}^{n_t} x_j$. If P^+ is improved, go to Step 3.

Else, if $n_t < n$, add $\mathbf{a}_{best}$ to $A_{m\times n_t}$, where $\mathbf{a}_{best}$ is chosen from the β newly generated feasible solutions using global evaluation function, set $n_t = n_t + 1$ and go to Step 1. Else STOP.

Step 3: Substitute a_{r^*} with a_{k^*}, giving a new matrix that have more rows than columns $A_{m\times n_t}$, go to Step 1.

In step 2, we use XPRESS to solve the integer programming problem P^+ and evaluate the improvement the substitution makes.

5. Adaptation of Sequential Heuristic Procedure (SHP) to solve irregular shape 2DCSP

The sequential heuristic procedure (SHP) generates cutting patterns and uses them sequentially until the solution meets all the problem requirements. Haessler (1971) first described a SHP algorithm. He pointed out that the key to success with this type of procedure is to select intelligently the patterns that are used early in the SHP. Ideally, the partial solution arising from the early pattern selection should have low trim loss and leave a set of requirements for future patterns that will combine well without excessive trim loss. Our SHP is based on the procedure described by Haessler and Sweeney (1991), as follows:

Initialization: Let $n_t = 1$ be the number of patterns used and $x_j = 0$ $(j = 1, 2, \ldots, n)$ be the frequency of use of pattern j. Thus the total stock sheets used is $N = \sum_{j=1}^{n} x_j$. We have $\overline{b}_i = b_i$ be the remaining demand of piece $i \in I$ and $\overline{A} = \phi$ be the cutting pattern set constructed.

Step 1: Construct feasible cutting pattern $\mathbf{a} = \{a_1, a_2, \ldots, a_m\}$ with beam search algorithm described in Section 3.2, where the global evaluation function returns the total area of the pieces packed and the local evaluation uses Priority(R, L, O) for both placement evaluation and next piece evaluation, which was called *one-step* strategy in Bennell and Song (2010).

Step 2: Let x be the number of times pattern $\mathbf{a} = \{a_1, a_2, \ldots, a_m\}$ is used, set $x = 0$.

S1: Set $x = x + 1$.

S2: If there exists piece i that satisfies $\overline{b}_i - x \cdot a_i < 0$, set $x = x - 1$, go to Step 3. Else, go to S1.

Step 3: Reduce the demand $\overline{b}_i = \overline{b}_i - x \cdot a_i$ for each piece i and add the feasible cutting pattern $\mathbf{a}$ to A, set $A = A + \mathbf{a}$. If $\overline{\mathbf{b}} = \{\overline{b}_1, \overline{b}_2, \ldots, \overline{b}_m\} = 0$, stop, else set $x_{n_t} = x$ and $n_t = n_t + 1$, go to step 1.

In Step 1, we generate a new pattern and in Step 2 we determine how many times this pattern will be used. (Note that although the beam search procedure in step 1 is deterministic, the demand values are different for each call leading to alternative patterns).

In the same way as the ACG, we set an upper bound UB_i on the number of times a piece can appear in a pattern. As a result, the SHP can be used to control the number of patterns used, however with no guarantee that a given number of patterns is attained. Also, a tradeoff between trim loss and number of patterns is hard to find because only one solution can be obtained using SHP.

6. Experimental Results

We implement Column Generation (CG), Sequential Heuristic Procedure (SHP) and Adapted Column Generation (ACG) as described in section 4 and Section 5 for the irregular CSP, where their sub problems are solved using a beam search algorithm. We use the Priority(R, L, O) to decide the position of a piece in the pattern layout for all approaches. Given the candidate position for each piece, the criterion to decide which piece to place next is different for different master problems. In CG, we used Priority(Pi, R, O). In ACG, we used Priority($\frac{\hat{\pi}_i - d_i}{x \cdot y}$, R, O). In SHP, we used Priority(R, L, O). The procedures were coded in Visual Studio C++ and the instances were run on a PC with 512 MB, 1.6 GHz.

For each method, Beam widths of 1, 10 and 100 are tested across 14 benchmark data sets for irregular shape strip packing from ESICUP (www.fe.up.pt/esicup), which we modify for our irregular shape CSP. There are two changes: one is that the length of the stock sheet is set equal to the width of the stock sheet, the other is to multiply the demand of each piece by 100 so that the demand is highly enlarged. For each data set the filter width cannot be greater than the number of piece types (including all orientations). Given that the filter width influences the number of global evaluations for each beam at each level, a large filter width can be costly. However, it must be large enough to provide scope to find good solutions arising from less greedy partial solutions. The filter width is set as follows, filter width = $min\{10$, no. of piece types$\}$ (Bennell and Song (2010)). For ACG, We test NP = [n_{LB}, 1.3*no. of piece type]. Table 1, 2 and 3 compare the CG, SHP and ACG results in terms of solution quality and computation time where Beam width = 1, 10 and 100 respectively. Table 4 and 5 report the usage ratio of the stock sheets and computer running time against the number of cutting patterns used for the ACG algorithm, where Beam width = 10 and 100 respectively. We do not report these results for beam width = 1, because they do not inform our analysis. Finally, in Table 6, we give the average percentage times spent in the solution of subproblems (single sheet heuristic) over the complete computer running time for CG and ACG algorithms, where beam width = 1, 10 and 100.

In Table 1, 2 and 3, column "Data Set" gives the name of the 14 problem instances tested. Column "no. of piece types" gives the number of piece types. Column "no. of pieces" gives the total number of pieces to be cut out. Column "α" gives the filter width of the beam search. Column "β" gives the beam width of the beam search. Column "CG","ACG" and "SHP" give the result from algorithm CG, ACG and SHP respectively. For each method, column "RATIO" gives the usage ratio of the stock sheets, which can be calculated as RATIO$=\frac{\sum_{i=1}^{m} s_i \cdot b_i}{\sum_{j=1}^{nP} x_j \cdot L \cdot W}$; column "NSHEET" gives the number of the stock sheets used, which is $\sum_{j=1}^{nP} x_j$; column "NP" gives the number of patterns used and column "TIME (S)" gives the computer running time in seconds. In ACG, "*ratio**" and "*nSheet**" gives the best usage ratio and the least number of stock sheets used respectively, regardless of the number of patterns used.

In Table 4 and 5, for each data set, the usage ratio of the stock sheets and computer running time in seconds are reported against the number of cutting patterns used, where the number of patterns are increased sequentially. Due to the size of the table, we don't give the the numbers of stock sheets used for each given pattern in Table 4 and 5, because these can be easily found out with the usage ratio of the stock sheets. The second row contains the names of the 14 problem instances, and column "NP" gives us the number of patterns used.

Table 1: Comparison of CG, ACG and SHP algorithms (Beam Width = 1)

DATA	NO. OF	NO. OF		$\beta = 1$											
SET	PIECE	PIECES	α	CG				ACG				SHP			
	TYPES			RATIO	NSHEET	NP	TIME (S)	*ratio**	*nSheet**	NP	TIME (S)	RATIO	NSHEET	NP	TIME (S)
Albano	8	2,400	6	**0.83**	**213**	11	15	0.83	215	7	26	0.79	225	7	0
Shapes2	7	2,800	6	**0.81**	**178**	9	5	0.79	183	7	8	0.74	194	5	0
Dagli	10	3,000	7	**0.84**	**101**	14	387	0.82	103	9	261	0.84	101	11	23
Dighe2	10	1,000	7	0.81	124	13	15	0.77	130	11	29	**1.00**	**100**	1	0
Dighe1	16	1,600	7	**0.78**	**128**	19	79	0.68	146	14	58	0.76	131	5	0
Fu	12	1,200	7	**0.89**	**112**	14	8	0.74	133	12	6	0.79	126	6	0
Jakobs1	25	2,500	7	0.83	329	28	78	0.80	339	13	74	**0.86**	**315**	9	0
Mao	9	2,000	6	0.67	164	12	137	**0.76**	**144**	7	129	0.71	154	7	1
Marques	8	2,400	6	**0.84**	**137**	9	49	0.58	200	1	0	0.83	139	6	4
Poly(5B)	25	2,500	7	**0.77**	**64**	33	1,656	0.65	76	6	69	0.73	68	10	37
Shapes1	4	4,300	4	**0.69**	**144**	5	6	0.66	150	4	12	0.67	148	7	3
Shirts	8	9,900	6	**0.85**	**158**	12	136	0.81	167	8	473	0.85	159	10	86
Swim	10	4,800	7	0.67	114	13	533	**0.71**	**108**	11	1,481	**0.71**	**108**	13	158
Trousers	17	6,400	7	**0.75**	**367**	20	1,471	0.63	434	3	9	0.71	385	13	6

From Table 1, 2 and 3, we can see that with larger beam width, the sub-problems of all the algorithms provide better solutions and in turn the solutions of their master problems are better. For the same problem instance, usage ratios of stock sheets are improved when the beam width is changed from 1 to 10 to 100. For all beam widths, the usage ratios of CG algorithm are better than those of SHP algorithm in general, however, CG algorithm is worse than SHP algorithm in terms of number of patterns used. ACG algorithm gives solutions slightly worse than the other two algorithms in terms of usage ratio, however, the maximum number of patterns can be controlled. When nP is set to be the no. of piece types, ACG may produce inferior results to CG, because the resolution of the ACG auxiliary problem is heuristic and the BS uses an alternative primary selection criterion. On average, ACG is 4.7% worse than CG and 4.1% worse than SHP when

Table 2: Comparison of CG, ACG and SHP algorithms (Beam Width = 10)

Data	no. of	no. of		$\beta = 10$											
Set	piece	pieces	α	CG				ACG				SHP			
	types			ratio	nSheet	nP	Time (s)	*ratio**	*nSheet**	nP	Time (s)	ratio	nSheet	nP	Time (s)
Albano	8	2,400	6	**0.84**	**211**	10	74	0.83	213	8	398	0.79	224	6	2
Shapes2	7	2,800	6	0.80	180	9	56	**0.80**	**179**	8	107	0.76	188	10	5
Dagli	10	3,000	7	**0.85**	**99**	12	5,979	0.82	103	9	1,869	0.84	100	9	49
Dighe2	10	1,000	7	0.90	111	11	252	0.82	122	9	111	**1.00**	**100**	1	0
Dighe1	16	1,600	7	**0.83**	**121**	16	984	0.77	130	13	1,025	0.82	122	8	21
Fu	12	1,200	7	**0.90**	**110**	15	77	0.81	123	12	93	0.84	119	5	0
Jakobs1	25	2,500	7	**0.90**	**301**	26	1,705	0.85	321	16	1,477	0.87	314	9	3
Mao	9	2,000	6	0.68	161	12	3,225	**0.79**	**138**	7	596	0.79	139	7	7
Marques	8	2,400	6	**0.86**	**134**	11	905	0.57	200	1	0	0.83	139	10	28
Poly(5B)	25	2,500	7	**0.81**	**61**	29	19,743	0.67	74	6	648	0.75	66	9	271
Shapes1	4	4,300	4	**0.70**	**143**	6	69	**0.70**	**143**	4	103	0.68	146	6	13
Shirts	8	9,900	6	**0.87**	**155**	11	6,983	0.86	157	9	7,309	0.87	156	12	1,029
Swim	10	4,800	7	0.72	106	13	6,373	0.71	109	13	28,949	**0.73**	**105**	9	1,145
Trousers	17	6,400	7	**0.79**	**348**	18	17,430	0.78	353	9	1,499	0.79	350	7	37

Table 3: Comparison of CG, ACG and SHP algorithms (Beam Width = 100)

Data	no. of	no. of		$\beta = 100$											
Set	piece	pieces	α	CG				ACG				SHP			
	types			ratio	nSheet	nP	Time (s)	*ratio**	*nSheetn**	nP	Time (s)	ratio	nSheet	nP	Time (s)
Albano	8	2,400	6	0.86	207	8	2,181	0.86	207	7	1507	**0.88**	**205**	2	30
Shapes2	7	2,800	6	**0.83**	**174**	9	476	0.77	186	7	430	0.77	186	7	23
Dagli	10	3,000	7	**0.87**	**97**	12	74,449	0.86	98	7	34334	0.85	100	8	312
Dighe2	10	1,000	7	0.93	108	8	2,002	0.87	115	2	2343	**1.00**	**100**	1	2
Dighe1	16	1,600	7	**0.89**	**114**	18	11,627	0.81	124	14	9965	0.85	118	4	42
Fu	12	1,200	7	**0.92**	**108**	15	580	0.84	119	10	1164	0.84	118	4	4
Jakobs1	25	2,500	7	**0.92**	**296**	30	20,524	0.83	327	21	7923	0.87	314	12	31
Mao	9	2,000	6	**0.81**	**136**	10	51,627	0.77	143	6	1020	0.79	139	7	50
Marques	8	2,400	6	**0.87**	**132**	11	10,713	0.58	200	1	11229	0.84	138	8	721
Poly(5B)	25	2,500	7	**0.82**	**60**	27	152,825	0.70	71	17	79158	0.78	63	8	2,169
Shapes1	4	4,300	4	**0.71**	**140**	6	335	0.71	141	5	2112	0.69	144	8	149

the beam width=10. Increasing the beam width to 100 gives better results in all cases where CG exhibits the greatest improvement at 3.9% on average. However, run times grow significantly. Note that the last three data sets are omitted in table 3 as the run time extended beyond a day.

From Table 4 and 5, we can see that although ACG is less efficient in getting high usage patterns, we can obtain a list of pattern usage ratio against the number of patterns, which could provide the opportunity to make a decision to balance the objectives of maximising usage ratio and minimising the number of patterns. This feature may be favoured in industry when the set up cost is high. The computer running times for ACG and CG are high in comparison to those of SHP.

Note, the cells marked "-" indicates that no result is reported in that cell. This is possible, since for some given "nP", the optimal solutions by XPRESS for solving integer programming problem P^+ in Step 2 in Section 4.4 returns zero for the number of times a specific pattern is used. In that case, no usage ratio or running time is reported. Empty cells occur when "nP" is greater than the number of piece types in the data set.

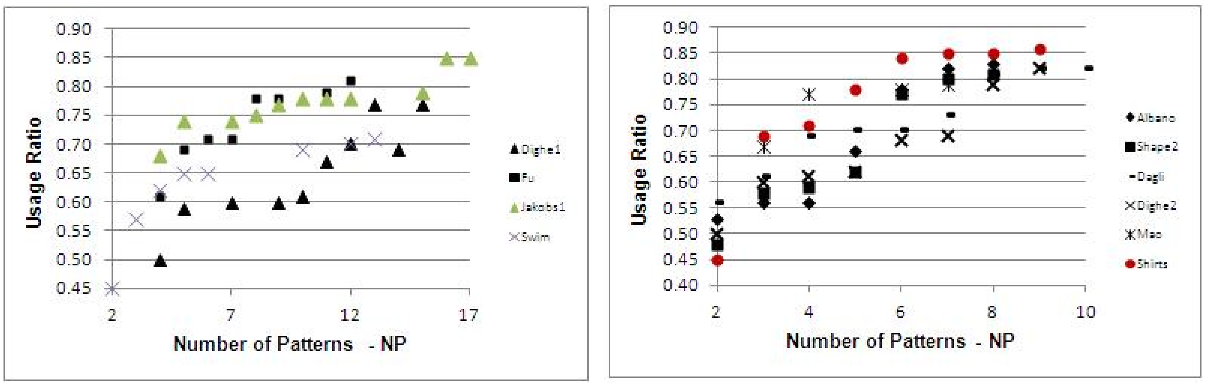

Figure 4: Trade off of usage ratio and number of patterns used in problem instances

Figure 4 shows the trade off between the usage ratio of the solution and "NP" using ACG, where beam width = 10, for two sets of instances. The four instances "Dighe1","Fu","Jakobs 1" and "Swim" (see left) have a large no. of piece types giving a long list of patterns, whereas the other 6 instances have a short list of patterns (see right). From Figure 4, we can see clearly that the usage ratio of the stock sheets improves in general along with the increase of the number of patterns used. However, this is not always the case. At some points, the usage ratio drops with higher "NP". There is no guarantee that the best usage ratio uses the least "NP", because ACG is heuristic.

Table 6 shows the average percentage time spent in the solution of the subproblem over the complete computer running time for CG and ACG algorithms. With larger beam width, this percentage increases in CG algorithm and decreases in ACG algorithm. Clearly, a larger beam width will lead to an increase in computation time in generating patterns. Since CG improves the restricted master problem using the revised simplex method, which is fast, the relative time spent on the subproblem increases. ACG algorithm solves the integer programming problem P^+ (Step 2 in Section 4.4) for each pattern arising from the beam search, which increases the computation time for this step. The relative increase in run time for step 2 is greater than the the beam search, thus the percentage of subproblem running time over the complete computer running decreases.

Overall, the results show that CG provides the best results in terms of trim loss, closely followed by SHP. Further SHP has significantly lower computation times and can produce good results with fewer patterns. However, neither approach can guarantee or control the maximum number of patterns in the solution. ACG can control the maximum number of patterns and explore the trade-off between the usage ratio of stock sheets and number of patterns used, but at a small cost in packing efficiency.

7. Conclusion

In this paper we present implementations of CG and SHP adapted to the irregular shape cutting stock problem. The sub problems are solved using the beam search algorithm. We also designed an ACG algorithm to control the number of different patterns, where the generalized inverse matrix is used to generate the approximate dual price to generate the next pattern.

The experiments show that CG provides the most efficient solutions in terms of trim loss. SHP can reduce the number of patterns and give good packing effciency, but the maximum number of patterns cannot be controlled, as in the CG approach. The ACG algorithm provides a tool to evaluate the trade off between trim loss and setup cost, but at a reduced packing efficiency. To a certain extent, we can control the number of patterns without generating too much trim loss. The running time of ACG and CG is much higher than SHP, which is due to the complexity of the sub-problem they need to solve.

This paper has contributed to the field of cutting and packing in solving directly the irregular cutting stock problem, without pre-constructing stencils. It has also contributed to the field of combinatorial optimisation by applying BS within CG, and developing the CG methodology to deal with basic feasible solution matrices that have more rows than columns. Each approach (CG, ACG and SHP) has been innovated to be capable of handing the irregular shape packing problem directly, or deal with the additional set up constraint.

8. Acknowledgements

This work has been funded by the UK's Engineering and Physical Sciences Research Council (EP-SRC), under grant GR/S52421/01.

References

Antonio, J., F. Chauvet, C. Chu, J. M. Proth. (1999). The cutting stock problem with mixed objectives: Two heuristics based on dynamic programming. *European Journal of Operational Research* **114:** 395-402.

Belov, G., G. Scheithauer. (2007). Setup and open-stacks minimization in one-dimensional stock cutting. *INFORMS Journal on Computing* **19:** 27-35.

Bennell, J., X. Song. (2010). A beam search implementation for the irregular shape packing problem. *Journal of Heuristics* **16:** 167-188.

Burkard, R.E., C. Zelle. (2003). A local search heuristic for the real cutting problem in paper production. *Technical Report, Institute of Mathematics B* **Technical University Graz.** SFB-257.

Chu, C.B., J. Antonio. (1999). Approximation algorithms to solve real-life multicriteria cutting stock problems. *Operations Research* **47:** 495-508.

Coverdale, I., F. Wharton. (1976). An improved heuristic procedure for a nonlinear cutting stock problem. *Management Science* **23:** 78-86.

Degraeve, Z., M. Vandebroek. (1998). A mixed integer programming model for solving a layout problem in the fashion industry. *Management Science* **44:** 301-310.

Diegel, A., E. Montocchio, E. Walters, Schalkwyk S.V., S. Naidoo. (1996). Setup minimizing conditions in the trim loss problem. *European Journal of Operational Research* **95:** 631-640.

Dowsland, K.A., W.B. Dowsland. (1992). Packing problems. *European Journal of Operational Research* **56:** 2-14.

Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research* **29:** 1092-1104.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research* **44:** 145-159.

ESICUP. *EURO Special Interest Group on Cutting and Packing.* **www.fe.up.pt/esicup**

Foerster, H., G. Wäscher. (2000). Pattern reduction in one-dimensional cutting stock problems. *International Journal of Production Research* **38:** 1657-1676.

Furini, F., E. Malaguti, R.M. Durán, A. Persiani, P. Toth. (2012). A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research* **218:** 251-260.

Garey, M.R., D.S. Johnson. (1979). Computer and Intractability: A Guide to the Theory of NP-Completeness. *San Francisco: W.H. Freeman.*

Gilmore, P.C., R.E. Gomory. (1961). A linear programming approach to the cutting stock problem. *Operations Research* **9:** 849-859.

Gilmore, P.C., R.E. Gomory. (1963). A linear programming approach to the cutting stock problem, part II. *Operations Research* **11:** 863-888.

Goulimis, C. (1990). Optimal solutions for the cutting stock problem. *European Journal of Operational Research* **44:** 197-208.

Gradišar, M., M. Kljajić, G. Resinovič, J. Jesenko. (1999). A sequential heuristic procedure for one-dimensional cutting. *European Journal of Operational Research* **114:** 557-568.

Gramani, M.C.N., P.M. França, M.N. Arenales. (2009). A lagrangian relaxation approach to a coupled lot-sizing and cutting stock problem. *International Journal of Production Economics* **119:** 219-227.

Haessler, R.W. (1971). A heuristic solution to a nonlinear cutting stock problem. *Management Science* **17:** B793-B803.

Haessler, R.W. (1975). Controlling cutting pattern changes in one-dimensional trim problems. *Operations Research* **23:** 483-493.

Haessler, R.W. (1980). A note on some computational modifications to the Gilmore-Gomory cutting stock algorithm. *Operations Research* **28:** 1001-1005.

Haessler, R.W., P.E. Sweeney. (1991). Cutting stock problems and solution procedures. *European Journal of Operational Research* **54:** 141-150.

Hendry, L.C., K.K. Fok, K.W. Shek. (1996). Cutting stock and scheduling problem in the copper industry. *Journal of the Operational Research Society* **47:** 38-47.

Imahori, S., M. Yagiura, S. Umetani, S. Adachi, T. Ibaraki. (2005). Local search algorithms for the two-dimensional cutting stock problem with a given number of different patterns. *Metaheuristics: Progress as Real Problem Solvers* **32:** 181-202.

Kasimbeyli, N., T. Sarac, R. Kasimbeyli. (2011). A two-objective mathematical model without cutting patterns for one-dimensional assortment problems. *Journal of Computational and Applied Mathematics* **235:** 4663-4674.

Lodi, A., S. Martello, D. Vigo. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research* **141:** 241-252.

Lübbecke, M.E., J. Desrosiers. (2005). Selected topics in column generation. *Operations Research* **53:** 1007-1023.

Martens, J. (2004). Two genetic algorithms to solve a layout problem in the fashion industry. *European Journal of Operational Research* **154:** 304-332.

McDiarmid, C. (1999). Pattern minimisation in cutting stock problems. *Discrete Applied Mathematics* **98:** 121-130.

Nonås, S.L., A. Thorstenson. (2000). A combined cutting-stock and lot-sizing problem. *European Journal of Operational Research* **120:** 327-342.

Nonås, S.L., A. Thorstenson. (2008). Solving a combined cutting-stock and lot-sizing problem with a column generating procedure. *Computers and Operations Research* **35:** 3371-3392.

Oliveira, J.F., A.M. Gomes, J.S. Ferreira. (2000). TOPOS - a new constructive algorithm for nesting problems. *OR Spektrum* **22:** 263-284.

Penrose, R. (1955). A generalized inverse for matrices. *Proc. Cambridge Phil. Soc.* **51:** 406-413.

Richard, V. (1996). Random search in the one-dimensional cutting stock problem. *European Journal of Operational Research* **85:** 191-200.

Rose, D.M., D.R. Shier. (2007). Cutting scheduling in the apparel industry. *Computers and Operations Research* **34:** 3209-3228.

Sinuany-Stern, I., Z. Weiner. (1994). The one dimensional cutting stock problem using two objectives. *Journal of the Operational Research Society* **45:** 231-236.

Song, X., C.B. Chu, Y.Y. Nie, J. Bennell. (2006). An iterative sequential heuristic procedure to a real-life 1.5-dimensional cutting stock problem. *European Journal of Operational Research* **175:** 1870-1889.

Sweeney, P.E., R.W. Haessler. (1990). One-dimensional cutting stock decisions for rolls with multiple quality grades. *European Journal of Operational Research* **44:** 224-231.

Umetani, S.J., M. Yagiura , T. Ibaraki. (2006). One-dimensional cutting stock problem with a given number of setups: a hybrid approach of metaheuristics and linear programming. *Journal of Mathematical Modelling and Algorithms* **5:** 43-64.

Valério de Carvalho, J.M., A.J. Guimarães Rodrigues. (1995). An LP-based approach to a two-stage cutting stock problem. *European Journal of Operational Research* **84:** 580-589.

Vanderbeck, F. (2000). Exact algorithm for minimizing the number of setups in the onedimensional cutting stock problem. *Operations Research* **48:** 915-926.

Wäscher, G., H. Haußner, H. Schumann. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research* **183:** 1109-1130.

Wäscher, G. (1990). An LP-based approach to cutting stock problems with multiple objectives. *European Journal of Operational Research* **44:** 175-184.

Zak, E.J. (2002). Row and column generation technique for a multistage cutting stock problem. *Computers and Operations Research* **29:** 1143-1156.

Table 4: The usage ratio of the stock sheets against nP for ACG (BW=10)

NP	ACG ($\beta = 10$)																											
	Albano		Shapes2		Dagli		Dighe2		Dighe1		Fu		Jakobs1		Mao		Marques		Poly(5B)		Shapes1		Shirts		Swim		Trousers	
	ratio	time (s)	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.58	0	-	-	-	-	-	-	-	-	-	-
2	0.53	5	0.48	3	0.56	40	0.50	0	-	-	-	-	-	-	0.34	7	-	-	0.49	0	0.50	0	0.45	0	0.45	0	0.69	45
3	0.56	17	0.58	5	0.61	61	0.60	7	-	-	-	-	-	-	0.67	60	-	-	-	-	0.52	42	0.69	870	0.57	1941	-	-
4	0.56	26	0.59	8	0.69	109	0.61	14	0.50	0	0.61	7	0.68	0	0.77	462	-	-	0.62	407	0.70	103	0.71	1116	0.62	2420	0.76	112
5	0.66	57	0.62	11	0.70	188	0.62	17	0.59	51	0.69	16	0.74	67	-	-	-	-	0.63	551	0.70	267	0.78	2601	0.65	3356	0.76	129
6	0.78	144	0.77	47	0.70	220	0.68	24	-	-	0.71	19	-	-	0.78	518	-	-	0.67	648			0.84	4507	0.65	4018	-	-
7	0.82	196	0.80	77	0.73	324	0.69	32	0.60	144	0.71	28	0.74	87	0.79	596	-	-	-	-			0.85	5130	-	-	0.77	185
8	0.83	398	0.81	107	0.81	1,712	0.79	86	-	-	0.78	60	0.75	115	-	-	-	-	-	-			0.85	5825	-	-	0.78	351
9	-	-			0.82	1,869	0.82	111	0.60	113	0.78	64	0.77	132	-	-	-	-	-	-			0.86	7309	-	-	0.78	1499
10					0.82	5,078	-	-	0.61	192	-	-	0.78	151	-	-			-	-					0.69	7339	-	-
11									0.67	471	0.79	74	0.78	197					-	-					-	-	-	-
12									0.70	797	0.81	93	0.78	226					-	-					0.70	16296	-	-
13									0.77	1025	-	-	-	-					-	-					0.71	28949	-	-
14									0.69	699			-	-					0.67	5576							-	-
15									0.77	1107			0.79	292					-	-							-	-
16									-	-			0.85	1477					-	-							-	-
17									-	-			0.85	1803					-	-							-	-
18													-	-					-	-							-	-
19													-	-					-	-								
20													-	-					-	-								
21													-	-					-	-								
22													-	-					-	-								
23													-	-					-	-								
24													-	-					-	-								
25													-	-					-	-								
26													-	-					-	-								
27													-	-					-	-								

Table 5: The usage ratio of the stock sheets against nP for ACG (BW=100)

NP	ACG ($\beta = 100$)																					
	Albano		Shapes2		Dagli		Dighe2		Dighe1		Fu		Jakobs1		Mao		Marques		Poly(5B)		Shapes1	
	ratio	time (s)	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.55	85	0.57	1	-	-	-	-
2	0.59	43	0.58	47	0.63	372	0.87	1,108	-	-	-	-	-	-	0.37	51			0.49	0	0.50	0
3	0.68	130	0.70	145	-	-			-	-	-	-	-	-	0.70	259			-	-	0.69	405
4	0.68	163	0.72	175	-	-			0.50	0	0.61	61	0.68	0	0.76	592			0.62	3671	0.70	847
5	0.81	989	0.75	207	0.71	2,167			0.58	411	-	-	0.75	415	0.75	500			0.64	4593	0.71	2112
6	0.77	891	0.76	311	-	-			-	-	-	-	0.76	636	0.77	1020			0.66	5468		
7	0.86	1,507	0.77	430	0.86	34,334			-	-	0.62	128	-	-					-	-		
8									0.60	749	-	-	0.77	962					-	-		
9									-	-	0.73	365	0.78	1104					-	-		
10									0.61	1,140	0.84	1,164	0.78	1350					-	-		
11									-	-	0.84	1,358	-	-					-	-		
12									0.61	1,865			-	-					-	-		
13									0.78	9,225			0.78	1918					-	-		
14									0.81	9,965			0.79	2251					-	-		
15													-	-					-	-		
16													0.79	2674					-	-		
17													0.79	3050					0.70	79158		
18													0.80	4534								
19													-	-								
20													-	-								
21													0.83	7923								
22																						
23																						
24																						
25																						
26																						
27																						

Table 6: Average percentage times spent in the solution of subproblems

Percentage times spent on subproblems	CG	ACG
beam width =1	0.95	0.81
beam width = 10	0.97	0.78
beam width =100	1.00	0.66