Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 1107–1120
computers & operations research
www.elsevier.com/locate/cor

An exact algorithm for generating homogenous T-shape cutting patterns

Yaodong Cui*

Department of Computer Science, Guangxi Normal University, Guilin 541004, China

Available online 23 June 2005

Abstract

Both the material usage and the complexity of the cutting process should be considered in generating cutting patterns. This paper presents an exact algorithm for constrained two-dimensional guillotine-cutting problems of rectangles. It uses homogenous T-shape patterns to simplify the cutting process. Only homogenous strips are allowed, each of which contains rectangular blanks of the same size and direction. The sheet is divided into two segments. Each segment consists of strips of the same length and direction. The strip directions of the two segments are perpendicular to each other. The algorithm is based on branch and bound procedure combined with dynamic programming techniques. It is a bottom-up tree-search approach that searches the solution tree from the branches to the root. Tighter bounds are established to shorten the searching space. The computational results indicate that the algorithm is efficient both in computation time and in material usage.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Cutting; Packing; Cutting stock; Constrained two-dimensional cutting

1. Introduction

The constrained two-dimensional cutting (CTDC) problem discussed is as follows: m types of rectangular blanks are to be cut from a rectangular sheet of size $L \times W$, where any cuts that are made are restricted to be guillotine cuts. The ith type has size $l_i \times w_i$, value c_i and demand d_i, $i = 1, \ldots, m$. Assume

* Corresponding author. Tel.: +86 773 5851286.
E-mail address: ydcui@263.net.

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.05.025

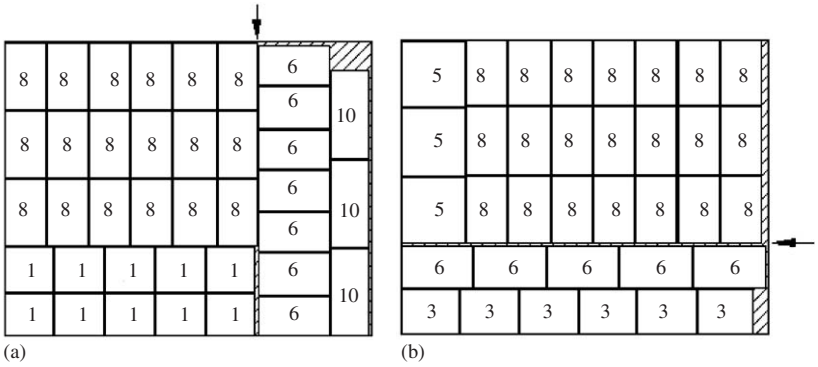

Fig. 1. Homogenous T-shape patterns. (a) A TX-pattern. (b) A TY-pattern.

that pattern A includes z_i pieces of type i. The mathematical model for the CTDC problem is

$$\max\left(\sum_{i=1}^{m} c_i z_i\right); \quad A \text{ is a feasible pattern}; \quad z_i \text{ integer}, \quad 0 \leqslant z_i \leqslant d_i, \quad i = 1, \ldots, m.$$

This paper applies the cutting patterns shown in Fig. 1, where the numbers indicate the blank types. A cut denoted by the arrow divides the sheet into two segments. Each segment consists of strips in the same direction. The strip directions of the two segments are perpendicular to each other. Only homogenous strips consisting of blanks of the same type are allowed. These patterns are referred to as homogenous T-shape patterns. The pattern in Fig. 1a is a TX-pattern where the dividing cut is vertical, and that in Fig. 1b is a TY-pattern where the dividing cut is horizontal. Here and in the following sections one should only consider the TX-pattern, and note that the TY-pattern appears by rotating the sheet and all items by 90°.

Homogenous T-shape patterns are a subset of three-stage cutting patterns. They are not a superset of two-stage cutting patterns, as two-stage cutting patterns allow general strips, each of which may include blanks of different types.

Homogenous T-shape patterns are interesting to OR practice for the reasons below. The first reason is that in the sheet metal industry, the sheet is often divided into blanks in two phases. First a guillotine shear cuts the sheet into strips. Then a stamping press punches out the blanks from the strips. The stamping phase requires that each strip include only blanks of the same type. The second reason is to simplify the cutting process. In developing countries, non-numerical controlled machines are widely used to divide the sheet into blanks. The cutting process of the T-shape patterns is simple, for that each strip consists of only blanks of the same type.

Many authors have investigated the CTDC problem. There exist two exact approaches: the top–down approach [1] and the bottom-up one [2–7]. The top–down approach uses a tree-search procedure. All possible cuts that can be made on the stock sheet are enumerated by means of a tree in which branching corresponds to guillotine cuts and nodes represent sub-rectangles obtained through the guillotine cut. The bottom–up approach is based on the observation that any pattern satisfying the guillotine

constraint can be obtained through horizontal and vertical builds of rectangles. All possible combinations of smaller rectangles are generated to obtain larger rectangles until no more guillotine patterns can be obtained. Both approaches can use upper and lower bounds to discard some non-promising branches.

Both homogenous T-shape patterns and two-stage patterns are the subsets of three-stage patterns. Algorithms for constrained two-stage patterns have been presented in the literature [8–10]. The author has not seen any report of the constrained homogenous T-shape patterns.

This paper presents an algorithm for CTDC problems. It can generate the optimal homogenous T-shape pattern. The computational results indicate that the algorithm is efficient both in material usage and in computation time.

2. Some basic concepts

To facilitate presentation, it is assumed that all blank types have fixed direction, and the sizes of the blanks and the stock sheet are integers.

Definition 1 (*X-strips, Y-strips, X-segments, and Y-segments*). An *X*-strip is a horizontal strip, and a *Y*-strip is a vertical one. A TX-pattern includes two segments (Fig. 1a). The left segment consisting of *X*-strips is an *X*-segment, and the right segment consisting of *Y*-strips is a *Y*-segment. The value of a strip is the sum of the values of the blanks included. The value of a segment is the sum of the values of the strips included. *X*-segment $x \times y$ means that in segment $x \times y$, the strip direction is along the side of length x. Similarly, *Y*-segment $x \times y$ means that the strip direction is along the side of length y.

Definition 2 (*Normal lengths*). For *X*-segments, a normal length is the multiple of a blank length. Normal lengths related to the ith blank type are

$$x_{ik} = kl_i, \quad k = 1, 2, \ldots, \min[d_i, \text{int}(L/l_i)], \quad i = 1, \ldots, m.$$

Assume that $\boldsymbol{P}$ is the set of normal lengths. Add 0 and L to this set, and arrange the elements in ascending order, namely $\boldsymbol{P} = \{p_1, \ldots, p_M\}$, $p_1 = 0$, $p_M = L$, and $p_i < p_{i+1}$ for $i = 1, \ldots, M-1$.

Normal lengths have the following property [5,11]: the optimal value of *X*-segment $x \times y$ is the same as that of *X*-segment $p(x) \times y$, where $p(x)$ is the maximum normal length not larger than x.

Definition 3 (*Normal segments and normal patterns*). Fig. 2 shows a normal *X*-segment, where the strips are arranged from bottom to top in non-increasing order of their lengths. The strips in a normal *Y*-segment are arranged from left to right in non-increasing order of their lengths. A T-shape pattern consisting of one normal *X*-segment and one normal *Y*-segment is referred to as a normal pattern. Any non-normal pattern can be transformed to a normal one through position interchanging of the strips, without changing the value of the pattern. That is to say, among the optimal patterns there exists a normal one. Therefore, only normal patterns will be considered in the searching process.

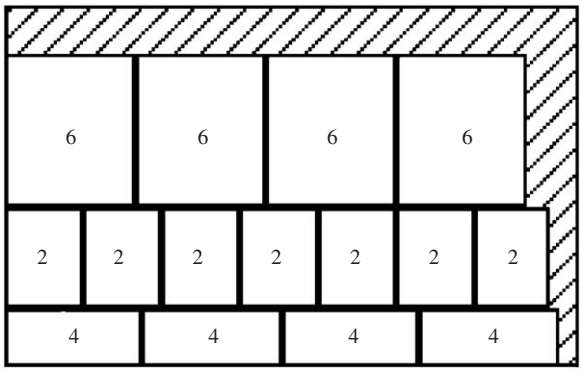

Fig. 2. A normal *X*-segment.

3. The approach

The approach for generating the optimal homogenous T-shape patterns consists of the following steps: (1) Generate all possible *X*-strips and *Y*-strips; (2) Obtain all possible *X*-segments through vertical build of *X*-strips; (3) Obtain all possible *Y*-segments through horizontal build of *Y*-strips. (4) Generate all possible T-shape patterns and select the best one as the optimal solution. Each T-shape pattern is generated through the horizontal build of an *X*-segment and a *Y*-segment.

3.1. Possible X-strips and Y-strips

Definition 4 (*Possible X-strips and Y-strips for building up segments*). In order to obtain all possible *X*-segments through vertical build of *X*-strips, all possible *X*-strips must be considered. For the *j*th blank type, the width of the *X*-strip is w_j, the number of different *X*-strips is $g_{X,j} = \min[d_j, \text{int}(L/l_j)]$. The *k*th strip contains *k* blanks, and its value is kc_j, $1 \leqslant k \leqslant g_{X,j}$. Similarly, in order to obtain all possible *Y*-segments through horizontal build of *Y*-strips, all possible *Y*-strips must be considered. For the *j*th blank type, the width of the *Y*-strip is l_j, the number of different *Y*-strips is $g_{Y,j} = \min[d_j, \text{int}(W/w_j)]$. The *k*th strip contains *k* blanks, and its value is kc_j, $1 \leqslant k \leqslant g_{Y,j}$.

3.2. Builds of strips

Definition 5 (*Vertical build of X-strips*). Two *X*-strips $A = l_A \times w_A$ and $B = l_B \times w_B$ can form an *X*-segment $C = l_C \times w_C$ through vertical build, where $l_C = \max(l_A, l_B)$, $w_C = w_A + w_B$, $l_C \leqslant L$ and $w_C \leqslant W$. Moreover, the numbers of blanks in *C* must not exceed the demands. The segment in Fig. 3b is obtained from the vertical build of two strips in Fig. 3a.

Vertical build can be also performed between an *X*-segment and a strip. It should not be performed between two *X*-segments; otherwise duplicate segments will be produced. For example, a segment of four strips can be obtained through the vertical build of two segments, each of which contains two strips.

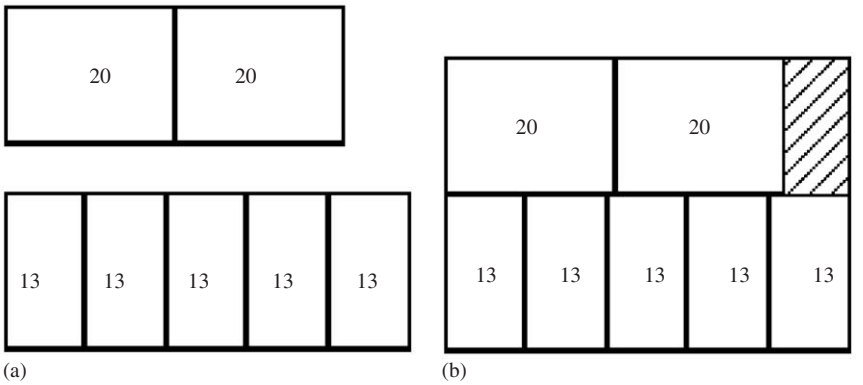

Fig. 3. Vertical build of *X*-strips. (a) Two *X*-strips. (b) An *X*-segment.

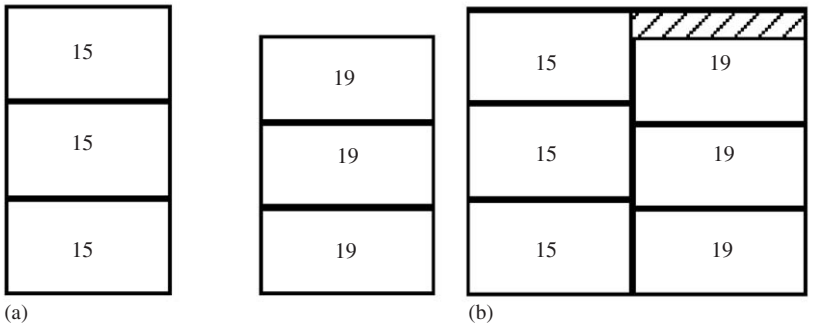

Fig. 4. Horizontal build of *Y*-strips. (a) Two *Y*-strips. (b) A *Y*-segment.

It can also be generated through the vertical build of a strip and a segment of three strips. Furthermore, from Definition 3 it is necessary only to generate normal *X*-segments.

Definition 6 (*Horizontal build of Y-strips*). Two *Y*-strips $A = l_A \times w_A$ and $B = l_B \times w_B$ can form a *Y*-segment $C = l_C \times w_C$ through horizontal build, where $l_C = l_A + l_B$, $w_C = \max(w_A, w_B)$, $l_C \leqslant L$ and $w_C \leqslant W$. Moreover, the numbers of blanks in C must not exceed the demands. The segment in Fig. 4b is obtained from the horizontal build of two strips in Fig. 4a. Horizontal build can be also performed between a *Y*-segment and a strip. Similarly to the vertical build of *X*-strips, horizontal build of *Y*-strips should not be performed between two *Y*-segments, and only normal segments should be generated.

3.3. Generating T-shape patterns

Definition 7 (*Horizontal build of two segments*). *X*-segment $A = l_A \times w_A$ and *Y*-segment $B = l_B \times w_B$ can form a T-shape pattern $C = l_C \times w_C$ through horizontal build, where $l_C = l_A + l_B$, $w_C = \max(w_A, w_B)$, $l_C \leqslant L$ and $w_C \leqslant W$. Moreover, the numbers of blanks in C must not exceed the demands. Fig. 5 shows

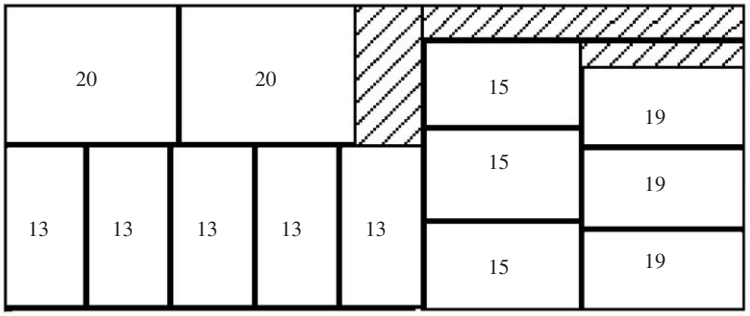

Fig. 5. A T-shape pattern obtained from the horizontal build of two segments.

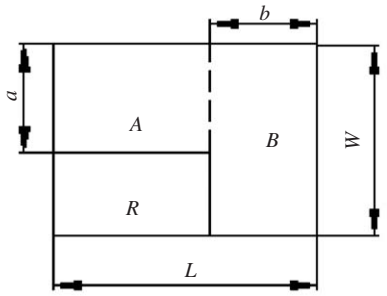

Fig. 6. Upper bound for a pattern containing X-segment R.

the T-shape pattern obtained from the horizontal build of the X-segment in Fig. 3b and the Y-segment in Fig. 4b.

3.4. The upper bound of the patterns related to an X-segment

All possible X-segments can be generated through the vertical build of X-strips according to Definition 5. Some of them may not appear in the optimal solution. To cut down the number of segments explicitly considered, it is helpful to use tighter upper bound.

Assume that $val(R)$, the value of X-segment $R = l_R \times w_R$, is already known. Let $uboundX(R)$ be the upper bound on the value of a pattern that contains R. The strips that will be combined with R later through vertical build have lengths not longer than l_R, for that only normal X-segments are to be considered. Divide the sheet into the three regions shown in Fig. 6, where $a = W - w_R$ and $b = L - l_R$. When the final T-shape pattern is generated, region A contains only X-strips, and region B contains only Y-strips. Let U_A be the upper bound of the total value of the blanks included in X-segment $l_R \times a$, and U_B be that in Y-segment $b \times W$. Then

$$unboundX(R) = val(R) + U_A + U_B, \tag{1}$$

in which U_A and U_B must be determined before $uboundX(R)$ can be estimated.

First consider how to estimate U_A. There is one X-strip width related to each blank type. Assume that the value of X-strip $x \times w_i$ is $u(i, x)$, which is determined as

$$u(i, x) = \min[d_i, \text{int}(x/l_i)] \times c_i.$$

Assume that z_i is the number of X-strip $x \times w_i$ appearing in X-segment $x \times y$. Let $F_X(x, y)$ be the maximum value of X-segment $x \times y$, without considering the constraint of the blank demands, then

$$F_X(x, y) = \max\left[\sum_{i=1}^{m} u(i, x)z_i\right];$$

$$\text{s.t.} \sum_{i=1}^{m} w_i z_i \leqslant y; \quad z_i \text{ non-negative integer}, \quad i = 1, \ldots, m; \quad x \in \boldsymbol{P}, \quad 0 \leqslant y \leqslant W. \tag{2}$$

Model (2) is an unbounded knapsack problem, and can be solved in $O(my)$ time [12,13]. After solving the model, it is reasonable to let $U_A = F_X(l_R, a)$.

Now consider how to estimate U_B. A constrained solution will be used. In a constrained solution, the numbers of blanks must not exceed the demands. Assume that $E_Y(x)$ is the value of the optimal constrained solution for Y-segment $x \times W$. To determine $E_Y(x)$, it is necessary to consider all possible Y-strips according to Definition 4. Index all possible Y-strips from 1 to n. Let $S = \{s_1, \ldots, s_n\}$ be the set of the widths and $T = \{t_1, \ldots, t_n\}$ be the set of values of these strips. Assume that z_i is the number of the ith strip included in the segment. The number of the kth blank type in the ith strip is δ_{ik}, $k = 1, \ldots, m$. Recall that each strip contains only one blank type. For a specific i, there exists only one integer j so that $\delta_{ik} > 0$ for $k = j$, and $\delta_{ik} = 0$ for $k \neq j$. The following model determines the constrained solution of the segment.

$$E_Y(x) = \max\left(\sum_{i=1}^{n} t_i z_i\right); \quad \text{s.t.} \sum_{i=1}^{n} s_i z_i \leqslant x; \quad \sum_{i=1}^{n} z_i \delta_{ik} \leqslant d_k, \quad k = 1, \ldots, m. \tag{3}$$

Model (3) is a bounded knapsack problem and the algorithms for it can be easily found in the literature [12–14]. It can be solved in $O\left(x\sum_{k=1}^{m} d_k\right)$ time. After solving the model, it is reasonable to let $U_B = E_Y(b)$.

3.5. The upper bound of the patterns related to a Y-segment

Assume that $val(R)$, the value of Y-segment $R = l_R \times w_R$, is already known. Let $uboundY(R)$ be the upper bound on the value of a pattern that contains R. Some strips may be combined with R later through horizontal build of Y-strips, and the resulting Y-segment may be combined with an X-segment to form a T-shape pattern. That is to say, the pattern containing R can be divided into two regions T and R shown in Fig. 7. Region $T = (L - l_R) \times W$ may contain a T-shape pattern. Assume that $F_T(x)$ is the upper bound of the T-shape pattern on partial sheet $x \times W$, then

$$F_T(x) = \max_{0 \leqslant z \leqslant x} [F_X(z, W) + E_Y(x - z)],$$

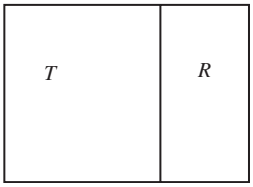

Fig. 7. Upper bound for a pattern containing *Y*-segment *R*.

where $F_X(z, W)$ is the value of *X*-segment $z \times W$ determined from Model (2), and $E_Y(x - z)$ is that of *Y*-segment $(x - z) \times W$ obtained from Model (3). Considering the property of normal lengths, the equation above can be written as

$$F_T(x) = \max_{z \in \boldsymbol{P}, z \leqslant x} [F_X(z, W) + E_Y(x - z)]. \tag{4}$$

A reasonable estimate for $uboundY(R)$ is

$$uboundY(R) = val(R) + F_T(L - l_R). \tag{5}$$

3.6. The initial lower bound

During the searching process, when the upper bound for a pattern containing segment *R* is less than or equal to the value of the best pattern obtained so far, segment *R* can be discarded. Initially the value of the best pattern is set to the value of a feasible pattern, which is referred to as the initial lower bound. Starting from a good initial lower bound can shorten the searching space greatly. The initial lower bound is determined in the following way.

Using the same set of strips as that in Model (2). Assume that $E_X(x)$ is the value of the heuristic constrained solution for *X*-segment $x \times W$, z_i is the number of the *i*th strip in the segment, and δ_i is the number of blanks in this strip, $i = 1, \ldots, m$. Then.

$$E_X(x) = \max \left[\sum_{i=1}^{m} u(i, x) z_i \right]; \quad x \in \boldsymbol{P},$$

$$\text{s.t.} \sum_{i=1}^{m} w_i z_i \leqslant W; \quad z_i \delta_i \leqslant d_i, \quad z_i \text{ non-negative integer}, \quad i = 1, \ldots, m. \tag{6}$$

Let $n_X(i, x)$ be the number of the *i*th blank type included in the heuristic constrained solution for *X*-segment $x \times W$, which can be obtained from the solution to Model (6), $n_Y(i, x)$ be that included in the optimal constrained solution for *Y*-segment $x \times W$, which is determined from Model (3). Let $Q = \{x | x \in \boldsymbol{P}; n_X(i, x) + n_Y(i, L - x) \leqslant d_i, i = 1, \ldots, m\}$. Assume that V_0 is the initial lower bound, which is determined from the following equation:

$$V_0 = \max_{x \in Q} [E_X(x) + E_Y(L - x)]. \tag{7}$$

3.7. The algorithm

Assume that $p_{\max} = \max_i \{c_i/(l_i w_i)\}$ is the maximum value of the unit area of the blanks. Use blocks to denote both strips and segments. Let V^* be the value of the best T-shape pattern obtained so far, *ubound*(R) be the upper bound on the value of the pattern containing block R, *Open*, *ClosedX* and *ClosedY* be three sets to keep blocks. *ClosedX* contains *X*-strips and *X*-segments. *ClosedY* contains *Y*-strips and *Y*-segments. The algorithm contains the following steps:

Step 1: Solve Model (2) to obtain $F_X(x, y)$, $x \in \boldsymbol{P}$ and $0 \leqslant y \leqslant W$. Solve Model (3) to obtain $E_Y(x)$, $0 \leqslant x \leqslant L$. Get $F_T(x)$ from Eq. (4), $0 \leqslant x \leqslant L$. Get $E_X(x)$ from Model (6), $x \in \boldsymbol{P}$. Obtain the initial lower bound V_0 from Eq. (7).

Step 2: Let $V^* = V_0$, go to Step 8 if $V^* = F_T(L)$. Generate all possible strips from Definition 4, and add them to *Open*. Let the upper bound of each initial strip in *Open* be $p_{\max} LW$.

Step 3: Go to Step 8 if *Open* is empty. Otherwise take one block R_A from *Open* having the highest upper bound *ubound*(R_A). Go to Step 8 if *ubound*(R_A) $\leqslant V^*$.

Step 4: Transfer R_A from *Open* to *ClosedX* if it is an *X*-block; otherwise to *ClosedY*. Go to step 6 if R_A is a *Y*-block.

Step 5: Vertical build: Consider the blocks in *ClosedX* one by one, until *ubound*(R_A) $\leqslant V^*$ or all blocks have been considered. When finished, go to step 7. For each block R_B taken from *ClosedX*, skip it if *ubound*(R_B) $\leqslant V^*$. Otherwise perform vertical build between R_A and R_B according to Definition 5. Determine the upper bound of the resulting block R_C according to Eq. (1). Discard R_C if *ubound*(R_C) $\leqslant V^*$. Otherwise add R_C to *Open*, and let $V^* = \max[V^*, val(R_C)]$.

Step 6: Horizontal build: Consider the blocks in *ClosedY* one by one, until *ubound*(R_A) $\leqslant V^*$ or all blocks have been considered. For each block R_B taken from *ClosedY*, skip it if *ubound*(R_B) $\leqslant V^*$. Otherwise perform horizontal build between R_A and R_B according to Definition 6. Determine the upper bound of the resulting block R_C according to Eq. (5). Discard R_C if *ubound*(R_C) $\leqslant V^*$. Otherwise add it to *Open*, and let $V^* = \max[V^*, val(R_C)]$.

Step 7: Go to Step 3 if *ubound*(R_A) $\leqslant V^*$. Let *Closed* = *ClosedY* if R_A is an *X*-block, otherwise let *Closed* = *ClosedX*. Consider the blocks in *Closed* one by one, until *ubound*(R_A) $\leqslant V^*$ or all blocks have been considered. For each block R_B taken from *Closed*, skip it if *ubound*(R_B) $\leqslant V^*$. Otherwise from Definition 7, obtain T-shape pattern R_C through the horizontal build of R_A and R_B. Let $V^* = \max[V^*, val(R_C)]$.

Step 8: Output V^* and stop.

The search for the optimal solution starts from an initial feasible lower bound V_0 (Step 2). If this lower bound equals to $F_T(L)$ that is the value of the optimal unconstrained solution for sheet $L \times W$, the initial feasible solution is optimal and it is not necessary to perform the searching process.

Steps 3–7 are performed repeatedly. Each time a block is selected from *Open*. If it is an *X*-block, vertical build should be made between it and each block in *ClosedA* to form new *X*-segments (Step 5), and it is combined with each *Y*-block in *ClosedB* to form new T-shape patterns (Step 7). If it is a *Y*-block, horizontal build should be made between it and each block in *ClosedB* to form new *Y*-segments (Step 6), and it is combined with each *X*-block in *ClosedA* to form new T-shape patterns (Step 7). The T-shape patterns generated in Step 7 will not be considered further. The *X*-blocks and *Y*-blocks generated in Steps 5 and 6 should be add to *Open* for further consideration, if they have upper bounds larger than the value of the best pattern obtained so far. The searching process ends when *Open* is empty or the maximum

value of the upper bounds of the blocks in *Open* is not larger than the value of the best pattern obtained so far (Step 4).

3.8. Inferior segments

Assume that segment $A = l_A \times w_A$ and segment $B = l_B \times w_B$ are two segments of the same type (either X-segments or Y-segments). Assume that $D_A(i)$ is the number of blank type i in segment A, and $D_B(i)$ is that in segment B. Refer A as an inferior segment if $l_A \geqslant l_B$, $w_A \geqslant w_B$, and $D_A(i) \leqslant D_B(i)$ holds for any i between 1 and m. In the searching process, inferior segments should not be added to *Open*.

4. The computational results

The computations were performed on a computer with Pentium 4 CPU, clock rate 2.8 GHz, and main memory 512 MB. Fifty test problems were randomly generated. There are 20 blank types for each problem. The value of each blank is equal to its area. The variable ranges are: sheet length in (2000, 2600), sheet width in (1000, 1300), blank length and width both in (50, 450), blank demand in (1, 10). The variables are uniformly distributed in the ranges. Once the paper is published, these problems will be made available on the Internet (http://www.gxnu.edu.cn/Personal/ydcui/Index.asp). We use material usage u to measure the quality of the solution. It is determined as

$$u = (\text{Total area of blanks in the pattern})/(\text{Area of the sheet}) \times 100\%.$$

4.1. General test

Only TX-patterns were generated. When the direction of the blanks is fixed, the average material usage is 96.3368%, the average computation time is 0.156 s, and the average number of blocks generated is 983. The algorithm can be easily adapted to the case where rotation of the blanks by 90° is allowed. When rotation of blanks by 90° is allowed, the average material usage is 97.9426%, the average computation time is 0.527 s, and the average number of blocks generated is 1851. Table 1 lists the first five test problems. Table 2 shows the computational results. These test problems are of middle scale. The computation time is reasonable for practical use.

4.2. Specialized test

This section illustrates the effects of upper bounds and the initial lower bound on the time efficiency of the algorithm. First introduce a valid upper bound. The upper bound of any block $R = x \times y$ can be written as

$$ubound(R) = p_{\max} xy, \tag{8}$$

where $p_{\max}$ is the maximum value of the unit area of the blanks. The following four cases are discussed:

(1) The initial lower bound $V_0 = E_Y(L)$, which is the value of the optimal constrained solution for Y-segment $L \times W$ obtained from Model (3). The upper bounds for region A in Fig. 6 and region T in Fig. 7 are both determined from Eq. (8). The upper bound for region B in Fig. 6 is $U_B = E_Y(b)$, which is the value of the optimal constrained solution for Y-segment $b \times W$.

Table 1
The first five test problems

ID	$L \times W$	$l_i \times w_i \times d_i$
1	2002 × 1001	53 × 67 × 1, 135 × 194 × 1, 119 × 407 × 7, 160 × 382 × 5, 194 × 410 × 8, 205 × 86 × 6, 297 × 319 × 3, 449 × 414 × 8, 81 × 223 × 2, 195 × 78 × 9, 400 × 172 × 4, 429 × 216 × 1, 170 × 217 × 4, 386 × 275 × 8, 51 × 324 × 4, 424 × 77 × 9, 92 × 416 × 4, 240 × 111 × 5, 383 × 160 × 8, 269 × 203 × 9
2	2449 × 1201	442 × 136 × 9, 300 × 64 × 10, 194 × 378 × 6, 103 × 301 × 4, 218 × 252 × 8, 310 × 89 × 10, 230 × 349 × 6, 352 × 341 × 10, 341 × 392 × 5, 316 × 113 × 7, 134 × 82 × 2, 144 × 334 × 3, 254 × 233 × 7, 111 × 167 × 1, 357 × 75 × 1, 426 × 399 × 2, 419 × 352 × 3, 193 × 311 × 6, 347 × 198 × 6, 243 × 190 × 1
3	2275 × 1191	132 × 209 × 4, 382 × 313 × 10, 129 × 97 × 6, 448 × 123 × 6, 247 × 299 × 6, 80 × 51 × 3, 195 × 239 × 7, 91 × 287 × 9, 251 × 77 × 5, 317 × 196 × 1, 397 × 115 × 75, 169 × 293 × 1, 70 × 260 × 10, 360 × 141 × 6, 389 × 305 × 5, 381 × 322 × 9, 51 × 336 × 6, 235 × 335 × 1, 287 × 128 × 1, 365 × 234 × 4
4	2222 × 1116	305 × 279 × 6, 343 × 384 × 5, 103 × 376 × 5, 268 × 210 × 3, 441 × 86 × 6, 302 × 93 × 2, 236 × 54 × 6, 120 × 173 × 5, 95 × 150 × 10, 215 × 392 × 5, 339 × 246 × 10, 73 × 267 × 2, 73 × 80 × 9, 96 × 124 × 9, 87 × 390 × 7, 160 × 445 × 1, 109 × 187 × 3, 231 × 58 × 9, 340 × 118 × 4, 446 × 237 × 3
5	2239 × 1036	271 × 428 × 5, 234 × 68 × 7, 225 × 327 × 7, 233 × 351 × 3, 396 × 255 × 1, 361 × 77 × 4, 252 × 239 × 4, 369 × 380 × 7, 207 × 144 × 1, 439 × 293 × 2, 164 × 116 × 2, 145 × 268 × 2, 84 × 245 × 6, 385 × 117 × 9, 445 × 390 × 6, 417 × 204 × 7, 329 × 150 × 4, 60 × 94 × 6, 191 × 78 × 1, 441 × 205 × 10

Table 2
The computational results for the first five test problems

Problem ID	1	2	3	4	5
Fixed direction					
Value	1 936 009	2 865 550	2 596 658	2 370 496	2 281 600
Material usage (%)	96.61	97.43	95.83	95.59	98.36
Computation time (s)	0.063	0.078	0.047	0.437	0.047
Number of blocks	352	623	236	2895	156
Rotation allowed					
Value	1 961 686	2 888 688	2 659 901	2 430 998	2 281 600
Material usage (%)	97.89	98.21	98.17	98.03	98.36
Computation time (s)	0.219	0.234	0.328	0.188	0.109
Number of blocks	490	770	1916	904	404

(2) The initial lower bound $V_0 = E_Y(L)$. The upper bound for region A is $U_A = F_X(l_R, a)$, which is the value of the optimal unconstrained solution for X-segment $l_R \times a$ obtained from Model (2). The upper bound for region B is $U_B = E_Y(b)$. The upper bound for region T is $F_T(L - l_R)$, which is obtained from Eq. (4).

(3) The initial lower bound is determined from Eq. (7). The upper bounds for regions A, B and T are the same as those in case (1).

Table 3
The effects of upper bounds and the initial lower bound

ID	t_F	n_F	t_R	n_R
1	4.173	7349	52.822	21 658
2	0.863	3947	10.506	10 405
3	0.639	2554	4.368	5118
4	0.156	983	0.527	1851

(4) The initial lower bound is determined from Eq. (7). The upper bounds for regions A, B and T are the same as those in case (2). The bounds for this case are the same as those for the general test.

The differences of the four cases can be summarizes as: For cases (2) and (1): The upper bounds for regions A and T in case (2) are tighter than those in case (1). For cases (3) and (1): The initial lower bound in case (3) is tighter than that in case (1). For cases (4) and (2): The initial lower bound in case (4) is tighter than that in case (2). For cases (4) and (3): The upper bounds for regions A and T in case (4) are tighter than those in case (3).

Table 3 shows the computational results, where t_F is the average computation time and n_F is the average number of blocks generated when the direction of the blanks is fixed, t_R and n_R are those when rotation of the blanks is allowed. The time unit is seconds. The results indicate that tighter bounds are useful for improving the time efficiency of the algorithm.

4.3. Test on benchmark problems

As pointed out in the introduction, using homogenous T-shape patterns can simplify the cutting process, and meet the requirement of the stamping phase. The algorithm is not designed to compete with other algorithms in material usage. The test on benchmark problems below is only used to indicate the ability of the algorithm to deal with relatively larger scale problems.

Hifi and Roucairol [8] presented an exact algorithm (the HRA for short) for generating staged (two-stage) patterns for CTDC problems. In Table 5 of Ref. [8], they gave the computational results of 38 test problems where general strips are applied. The computer used is of clock rate 0.25 GHz and main memory 128 MB. The total computation time for generating the horizontal two-stage patterns is 9469.4 s. The algorithm of this paper (the CA for short) solved these problems in 0.234 s, which is much shorter than that of the HRA. Assume that V_{HRA} is the value of the optimal two-stage pattern, V_{CA} is that of the optimal homogenous T-shape pattern, then $V_{CA} > V_{HRA}$ for 12 problems, $V_{CA} = V_{HRA}$ for 3 problems, and $V_{CA} < V_{HRA}$ for 23 problems. Although the computers used by the two algorithms are different, it is still obviously that the time required to generate an optimal homogenous T-shape pattern is much shorter than that to generate an optimal two-stage pattern.

Other algorithms exist for CTDC problems. Generally speaking, the time required to generate a non-staged pattern is much longer than that required for generating a staged pattern, and the time for generating a staged pattern is much longer than that for a homogenous T-shape pattern. Therefore, the CA can solve problems of relatively larger scale.

4.4. Solution to a real cutting problem

The example below is taken from a factory that makes passenger cars. Blanks of 49 types are to be cut from a stock sheet of 2600 × 1300. Table 4 shows the blank data. The value of each blank is equal to its

Table 4
The blank data of the example ($l_i \times w_i \times d_i$)

990 × 235 × 6, 220 × 113 × 6, 288 × 250 × 6, 1160 × 868 × 3, 1000 × 868 × 3, 868 × 598 × 6, 1160 × 258 × 3, 1000 × 96 × 15, 1000 × 152 × 15, 185 × 160 × 3, 1183 × 521 × 3, 815 × 521 × 3, 1012 × 571 × 3, 455 × 180 × 3, 540 × 192 × 3, 555 × 210 × 3, 253 × 143 × 3, 442 × 420 × 3, 240 × 90 × 3, 330 × 182 × 3, 800 × 133 × 3, 800 × 403 × 6, 803 × 333 × 3, 778 × 116 × 3, 778 × 78 × 3, 1200 × 241 × 3, 1020 × 262 × 3, 1300 × 121 × 3, 1300 × 83 × 3, 800 × 120 × 3, 700 × 133 × 3, 1200 × 85 × 3, 480 × 480 × 6, 965 × 705 × 3, 800 × 390 × 3, 680 × 185 × 3, 840 × 410 × 3, 800 × 73 × 3, 800 × 40 × 3, 930 × 78 × 3, 930 × 50 × 3, 600 × 88 × 3, 1200 × 146 × 6, 822 × 465 × 3, 876 × 318 × 3, 755 × 147 × 3, 460 × 180 × 3, 270 × 117 × 3, 215 × 145 × 3

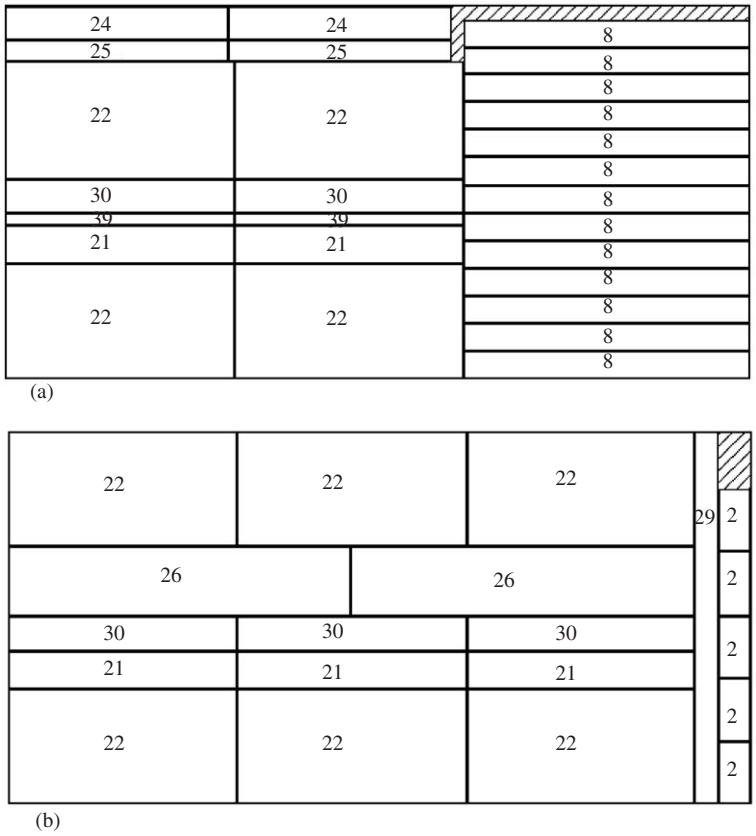

Fig. 8. The cutting pattern of the real example. (a) Fixed direction, computation time 0.656 s, value 3 308 264, material usage 97.88%. (b) Rotation allowed, computation time 1.141 s, value 3 352 200, material usage 99.18%.

area. Fig. 8 shows the cutting patterns and the computation results. In interpreting the computation times, it should be noted that for this example, the original main memory (512 MB) is exhausted and virtual memory is created in the hard disk, a portion of the computation time is consumed in interchanging data between the main memory and the virtual memory.

5. Conclusions

Because of the simplicity of the cutting process, cutting patterns of homogenous strips have been widely used in practice. They are especially appropriate for the case where non-numerical controlled guillotine shears are used, or the shearing and punching process is applied.

Other exact algorithms exist for the CTDC problems. Usually the patterns generated do not consist of strips, and are not adequate for the shearing and punching process. The staged patterns of general or uniform strips are not adequate either. If cutting patterns consisting of homogenous strips are demanded, the algorithm of this paper may be helpful.

Let t_1 be the time required to generate a non-staged pattern discussed in [1–7], t_2 to generate a two-stage pattern of general strips [8], and t_3 to generate a homogenous T-shape pattern. Usually $t_1 \gg t_2 \gg t_3$. That is to say, the algorithm of this paper has the ability to deal with relatively larger scale problems in a given time period.

Acknowledgements

This paper is part of the project supported by Guangxi Science Foundation (Grant 0236017). The authors wish to express their appreciation to the supporter.

References

[1] Christofides N, Whitlock C. An algorithm for two-dimensional cutting problems. Operations Research 1977;25:31–44.
[2] Wang PY. Two algorithms for constrained two-dimensional cutting stock problems. Operations Research 1983;32(3): 573–86.
[3] Viswanathan KV, Bagehi A. Best-first search methods for constrained two-dimensional cutting stock problems. Operations Research 1993;41(4):768–76.
[4] Hifi M. An improvement of Viswanathan and Bagchi's exact algorithm for constrained two-dimensional cutting stock. Computers & Operations Research 1997;24(8):727–36.
[5] Cung V-D, Hifi M, Cun BL. Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. International Transactions in Operations Research 2000;7:185–210.
[6] Amaral ARS, Wright M. Efficient algorithm for the constrained two-dimensional cutting stock problem. International Transactions in Operations Research 2001;8:3–13.
[7] Daza VP, Alvarenga AG, Diego J. Exact solutions for constrained two-dimensional cutting problems. European Journal of Operational Research 1995;84:633–44.
[8] Hifi M, Roucairol C. Approximate and exact algorithms for constrained (un)weighted two-dimensional two-staged cutting stock problems. Journal of Combinatorial Optimization 2001;5:465–94.
[9] Hifi M, Hallah RM. An exact algorithm for constrained two-dimensional two-staged cutting problems. Operations Research 2005;53:140–50.
[10] Lodi A, Monaci M. Integer linear programming models for 2-staged two-dimensional knapsack problems. Mathematical Programming 2003;94:257–78.
[11] Hifi M. Exact algorithms for large-scale unconstrained two and three staged cutting problems. Computational Optimization and Applications 2001;18:63–88.
[12] Andonov R, Poirrez V, Rajopadhye S. Unbounded knapsack problem: dynamic programming revisited. European Journal of Operational Research 2000;123:394–407.
[13] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. Berlin: Springer; 2004.
[14] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. Chichester, UK: Wiley; 1990.