J Glob Optim (2008) 41:267–281
DOI 10.1007/s10898-007-9252-z

T-shape homogenous block patterns for the two-dimensional cutting problem

Yaodong Cui · Zhiyong Liu

Received: 22 December 2005 / Accepted: 3 October 2007 / Published online: 27 October 2007

Abstract This paper presents an algorithm for unconstrained T-shape homogenous block cutting patterns of rectangular pieces. A vertical cut divides the stock sheet into two segments. Each segment consists of sections that have the same length and direction. A section contains a row of homogenous blocks. A homogenous block consists of homogenous strips of the same piece type. Each cut on the block produces just one strip. The directions of two strips cut successively from a block are either parallel or orthogonal. The algorithm uses a dynamic programming recursion to generate optimal blocks, solves knapsack problems to obtain the block layouts on the sections and the section layout on segments of various lengths, and optimally selects two segments to compose the cutting pattern. The computational results indicate that the algorithm is efficient in improving material usage, and the computation time is reasonable.

Keywords Cutting stock · Unconstrained two-dimensional cutting · Homogenous blocks

1 Introduction

The unconstrained two-dimensional cutting (UTDC) problem discussed is as follows [1]: m types of rectangular pieces are to be cut from a rectangular sheet $L \times W$ (length L and width W), where the cuts made are restricted to guillotine ones. The ith type has size $l_i \times w_i$, and value $c_i, i = 1, \ldots, m$. Assume that pattern A includes z_i pieces of type i, and $\boldsymbol{N}$ is the set of natural numbers. The mathematical model for the UTDC problem is:

$$\max \left\{ \sum_{i=1}^{m} c_i z_i; \quad A \text{ is a feasible pattern}; \quad z_i \in \boldsymbol{N}, i = 1, \ldots, m \right\}$$

Y. Cui (✉)
Department of Computer Science, Guangxi Normal University, Guilin 541004, China
e-mail: ydcui@263.net

Z. Liu
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

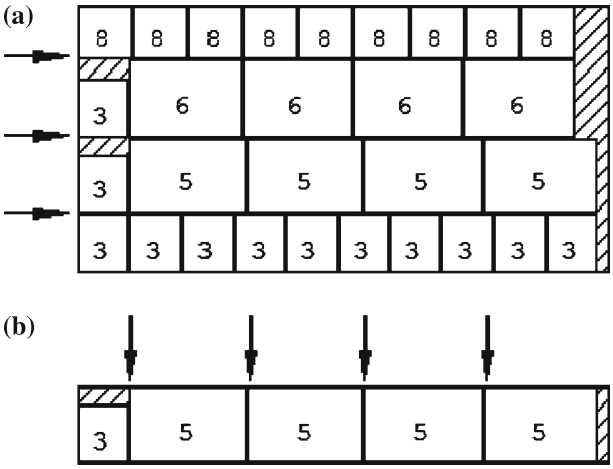

Fig. 1 A two-stage pattern and its cutting process. (**a**) The pattern and the first stage cuts and (**b**) the third strip and the second stage cuts

The patterns for the above general UTDC problem are called general patterns. Although exact algorithms [2,3] exist for general patterns, the computational complexities of them are unknown. The computational experiences in the literature have indicated that the time required by an exact algorithm to solve large-scale problems may become unaffordable. This has motivated people to study specialized, less-general types of patterns that can be efficiently solved to optimality. So there is a tradeoff between solution time and solution quality (closeness to the general solution).

A guillotine cut always divides a rectangle into two small rectangles. Staged patterns are typical specialized patterns. A staged pattern can be cut into pieces in several stages [4–6]. The direction of the cuts made at the next stage is perpendicular to that of the cuts made at the current stage. Figure 1 shows a two-stage pattern and its cutting process. Horizontal cuts denoted by the arrows divide the sheet into strips at the first stage, and vertical cuts divide the strips into pieces at the second stage. Figure 2 shows a three-stage pattern that can be cut into pieces in three stages. Vertical cuts divide the sheet into segments at the first stage.

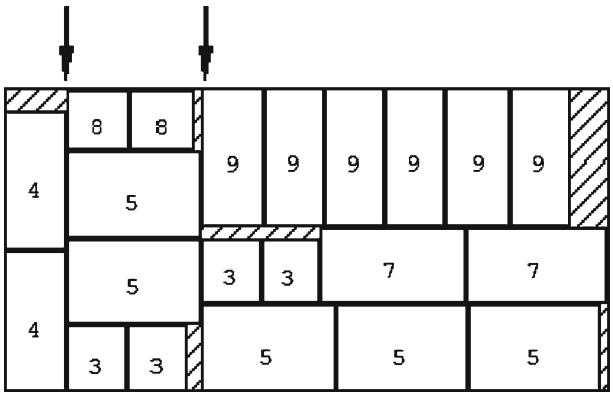

Fig. 2 A three-stage pattern and the first stage cuts

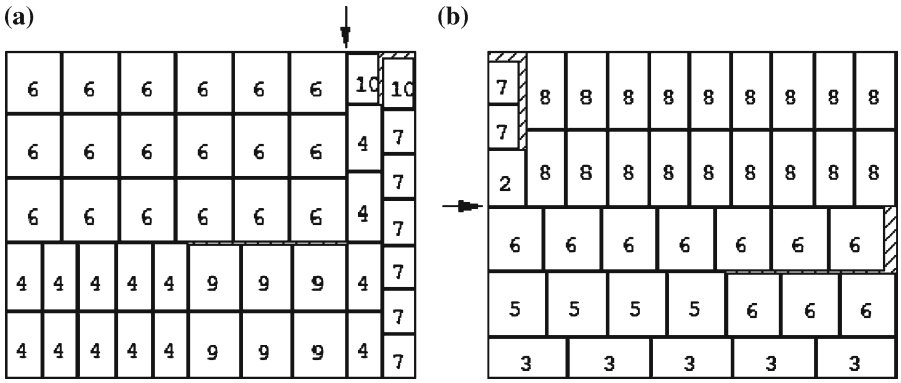

Fig. 3 T-shape patterns (copied from [1]). (**a**) The dividing cut is vertical and (**b**) the dividing cut is horizontal

Each segment contains a two-stage pattern that can be cut into pieces in the same way as that in Fig. 1. Algorithms for two- and three-stage patterns can be found in [4–6], and those for multistage patterns in [5–7]. They are all based on dynamic programming techniques.

T-shape patterns [8] also belong to specialized patterns. As shown in Fig. 3, a dividing cut denoted by the arrow divides a T-shape pattern into two segments. One contains horizontal strips, and the other consists of vertical strips. The algorithm in [8] is based on dynamic programming techniques. It determines the optimal position of the dividing cut and the strip layouts on the two segments. A T-shape pattern becomes a two-stage pattern if the dividing cut coincides with one edge of the sheet. In this case, the pattern contains only one segment. T-shape patterns are a super set of two-stage patterns, and a subset of three-stage patterns.

This paper proposes the T-shape homogenous block (TSHB) pattern for the UTDC problem. The TSHB pattern generalizes the T-shape pattern by allowing each *rectangle* in the T-shape pattern be a *block* of same-size rectangles called a "homogeneous block". Each homogenous block contains only pieces of the same type. Figure 4 shows two homogenous blocks that will be defined precisely in a later section.

The TSHB pattern generalizes both the two-stage pattern and the T-shape pattern, and so an exact solution to the TSHB problem must be at least as good as that of the optimal two-stage or the optimal T-shape pattern. On the other hand, the three-stage, and indeed any

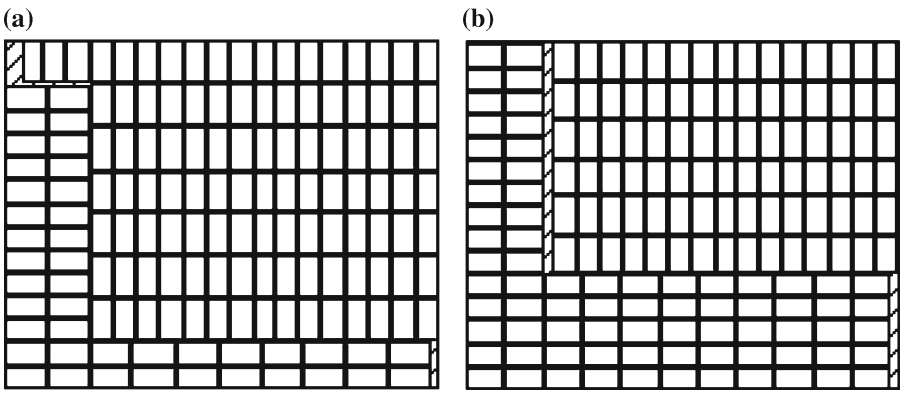

Fig. 4 Homogenous blocks. (**a**) The first strip is horizontal and (**b**) the first strip is vertical

given-stage pattern does not contain the TSHB as a sub-case, that is, it does not generalize the TSHB pattern. This will be explained later in the section that defines TSHB pattern. So it is possible for the optimal TSHB solution to be better than that of the three-stage solution.

In the rest of this paper, we will present an exact algorithm for solving the TSHB problem and present numerical results on a large set of test problems. For these test problems, the optimal TSHB solution was indeed better than the optimal three-stage solution for the large majority of the cases, while the computation time was actually shorter. Moreover, the solution value was often identical or close to the value of the general UTDC problem. The implications of these results for the cutting and packing community will be explored in Sect. 5.

2 Some concepts of the TSHB pattern

2.1 Homogenous blocks

Figure 5 shows three strip types often used in generating cutting patterns, where the numbers denote the piece types. The first is a general strip consisting of pieces of different widths; the second is a uniform strip containing pieces of the same width that is equal to the strip width; and the third is a homogenous strip containing pieces of the same type.

Figure 4 shows two homogenous blocks. A homogenous block consists of homogenous strips, where the same piece type is not only used within each strip, but also between strips. The strips in a homogenous block can be cut out using a succession of guillotine cuts. Each cut produces just one strip. Two strips that are cut successively are either parallel or orthogonal. The first strip in a block may be either horizontal (Fig. 4a) or vertical (Fig. 4b). The approaches proposed in [9–13] can be used for generating optimal homogenous blocks.

2.2 Sections, segments and TSHB patterns

A segment in a simple T-shape pattern (see Fig. 3) contains either horizontal strips (X-strips) or vertical strips (Y-strips), whereas in a TSHB pattern, a segment contains either X-sections or Y-sections. An X-section consists of a row of homogenous blocks that are arranged horizontally from left to right. The X-section in Fig. 6 contains three blocks, where the arrows denote the borders of the blocks. An X-section becomes a Y-section when it is rotated by 90°.

A segment contains sections that have the same length and direction. Figure 7 shows an X-segment consisting of X-sections (Fig. 7a), and a Y-segment consisting of Y-sections (Fig. 7b), where the arrows denote the borders of the sections. Each X-section in Fig. 7a contains two homogenous blocks. The four Y-sections in Fig. 7b contain respectively 1, 2, 2, and 4 homogenous blocks.

Figure 8 shows the TSHB patterns proposed. Each pattern contains two segments, one of which is an X-segment, and the other is a Y-segment. The arrow in each pattern denotes the dividing cut of the two segments. The X-segment is on the left in a TX-pattern (Fig. 8a), and on the bottom in a TY-pattern (Fig. 8b). The X-segment in Fig. 8a contains two X-sections,

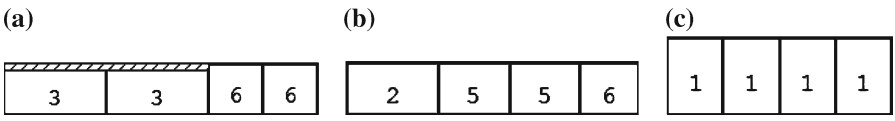

Fig. 5 Three strip types. (**a**) A general strip, (**b**) a uniform strip and (**c**) a homogenous strip

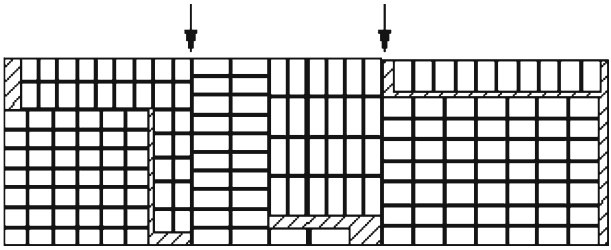

Fig. 6 An X-section

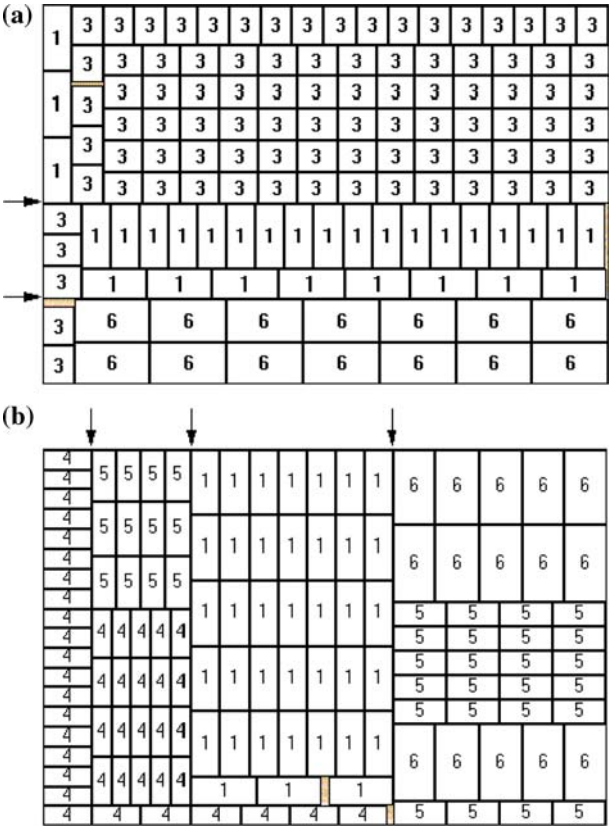

Fig. 7 Segments. (**a**) An X-segment and (**b**) a Y-segment

where the first section contains three homogenous blocks (related to piece types 5, 1, and 4 respectively), and the second contains two homogenous blocks (piece types 3 and 1). The Y-segment in Fig. 8a contains only one Y-section that consists of two homogenous blocks (piece types 2 and 1).

Although the TSHB pattern is also a staged pattern, it cannot be generalized by any given-stage pattern, because there is no constraint on the stage number of a homogenous block.

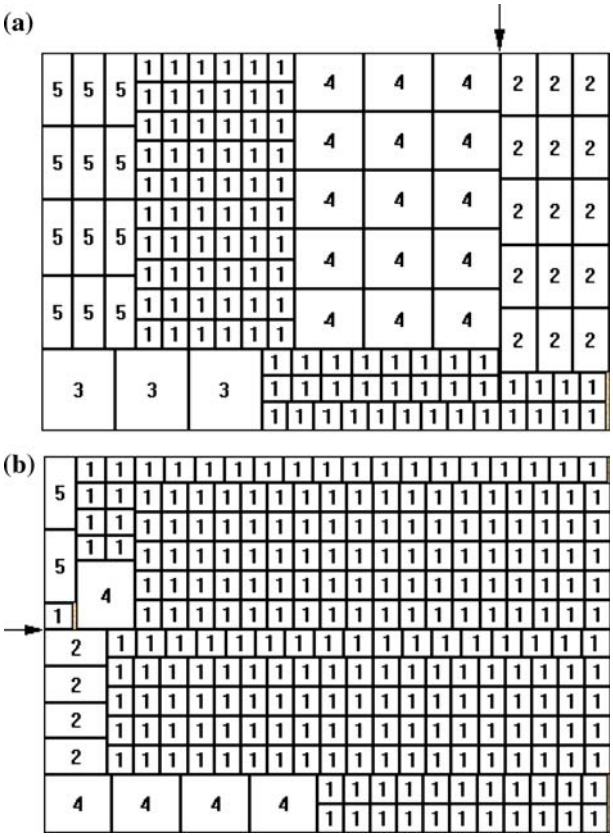

Fig. 8 TSHB patterns. (**a**) A TX-pattern and (**b**) a TY-pattern

3 The algorithm

Assume that the sizes of the stock sheet and the pieces are all integers. The pieces are allowed to rotate by 90°. As a result, a sheet of size $L \times W$ is equivalent to that of size $W \times L$. Therefore, it is assumed that $L \geq W$.

The approach will be only presented for the TX-pattern. It consists of the following steps: (1) Generating optimal blocks; (2) determining optimal block layouts on sections; (3) determining optimal section layouts on segments; and (4) selecting two segments with orthogonal directions to compose the cutting pattern. These steps will be presented one by one after the introduction of normal sizes.

3.1 Normal sizes

Normal sizes have been used by many authors [1,4,6,9] and must be defined according to the properties of the problem concerned. Take the case of a homogenous block as an example. The basic idea is that we can limit ourselves to blocks that are tightly packed, with all the rectangles touching, and so the lengths and widths can only take on a finite set of possible values (given a finite sheet), and these are the normal lengths and widths. This is actually a

key concept in the cutting and packing literature. One of the key results in this literature is as follows [13]: Assume that $g(x)$ is the maximum normal length not larger than x, $h(y)$ is the maximum normal width not larger than y, and $f(x, y)$ is the maximum value of rectangle $x \times y$ that contains smaller pieces, then $f(x, y) = f[g(x), h(y)]$. That is to say, considering only normal sizes is sufficient in solving cutting problems. Three types of normal sizes will be defined below.

Definition 1 (*Normal sizes of blocks*) Each block contains pieces of the same type. Therefore, normal sizes of blocks are defined for each piece type. For piece type i, $i = 1, \ldots, m$, assume that the set of normal sizes is $P^{(i)}$, then:

$$P^{(i)} = \{x = z_1 l_i + z_2 w_i;\ z_1, z_2 \in \boldsymbol{N};\ 0 \le x \le L\}$$

That is to say, normal block sizes are the combinations of the piece length and width. To facilitate the presentation, add L and W to $P^{(i)}$ if they are not in the set. A block is a normal one if both its length and width are normal sizes.

Definition 2 (*Normal sizes of sections*) A section contains a row of blocks (Fig. 6). As mentioned in Sect. 2.2, the section direction may be either horizontal (X-section) or vertical (Y-section). The width of an X-section is measured vertically, and that of a Y-section is measured horizontally. Therefore, the maximum length and width of the sections are both L. If the width of a section does not equal to any normal block width, it may be shortened to the nearest normal block width, without changing the block layout on the section. Therefore, the set of normal section widths is defined as:

$$Q_{\text{section}} = \bigcup_{i=1}^{m} P^{(i)} = \left\{q_1^{\text{sec}}, q_2^{\text{sec}}, \ldots, q_N^{\text{sec}}\right\}$$

in which N is the number of elements in the set.

Normal section lengths are the combinations of all block sizes, or the combinations of all piece lengths and widths. They are denoted by the following set:

$$\begin{aligned} P_{\text{section}} &= \left\{x = \sum_{i=1}^{m} (z_{1i} l_i + z_{2i} w_i);\ z_{1i}, z_{2i} \in \boldsymbol{N}, i = 1, \ldots, m;\ 0 \le x \le L\right\} \\ &= \{p_1^{\text{sec}}, p_2^{\text{sec}}, \ldots, p_M^{\text{sec}}\} \end{aligned}$$

in which M is the number of elements in the set.

A section is a normal one if its length belongs to P_{section}, and its width belongs to Q_{section}.

Definition 3 (*Normal sizes of segment*) A segment contains sections in the same direction. The length of the segment is the same as that of the sections included. Assume that P_{segment} is the set of normal segment sizes. It is the same as the set of section lengths, namely $P_{\text{segment}} = P_{\text{section}}$. A segment is a normal one if both its length and width belong to P_{segment}.

3.2 Optimal blocks

Assume that $B(x, y)$ is the global optimal value of block $x \times y$, $x \in P_{\text{segment}}$, $y \in P_{\text{segment}}$ and $y \le W$. The steps to determine $B(x, y)$ are:

Step 1. Let $B(x, y) = 0$ for all $x \in P_{\text{segment}}$, $y \in P_{\text{segment}}$ and $y \le W$. Let $i = 1$.

Step 2. Using the algorithm in [13] to determine $n(x, y)$, the number of pieces of type i in block $x \times y$, $x \in P^{(i)}$, $y \in P^{(i)}$ and $y \leq W$.

Step 3. Consider the normal block sizes of piece type i one by one. Go to Step 4 when finished. For each block $x \times y$ currently considered, let $v = n(x, y) \times c_i$ and $B(x, y) = \max[B(x, y), v]$.

Step 4. Let $i = i + 1$. Go to Step 2 if $i \leq m$; Otherwise go to Step 5.

Step 5. Normalize $B(x, y)$ so that $B(x, y) \geq B(x_0, y_0)$ holds if $x_0 \leq x$ and $y_0 \leq y$, where x, x_0, y, y_0 are all normal segment sizes.

3.3 Optimal block layouts on sections

A section contains a row of blocks. The following model determines $u(x, y)$, the optimal value of section $x \times y$ (length x and width y), $x \in P_{\text{section}}$ and $y \in Q_{\text{section}}$:

$$u(x, y) = \left\{ \max \left[\sum_{i=1}^{M} z_i B(p_i^{\text{sec}}, y) \right]; \quad \sum_{i=1}^{M} z_i p_i^{\text{sec}} \leq x; \quad z_i \in \boldsymbol{N} \right\} \tag{1}$$

Model (1) is an unbounded knapsack problem, and good algorithms exist for it [14].

3.4 Optimal section layouts on segments

Assume that $f_Y(x, W)$ is the value of Y-segment $x \times W$, and $f_X(x, W)$ is the value of X-segment $x \times W$. The following model determines $f_Y(x, W)$, $x \in P_{\text{segment}}$:

$$f_Y(x, W) = \left\{ \max \left[\sum_{i=1}^{N} z_i u(W, q_i^{\text{sec}}) \right]; \quad \sum_{i=1}^{N} z_i q_i^{\text{sec}} \leq x; \quad z_i \in \boldsymbol{N} \right\} \tag{2}$$

The following model determines $f_X(x, W)$, $x \in P_{\text{segment}}$:

$$f_X(x, W) = \left\{ \max \left[\sum_{i=1}^{N} z_i u(x, q_i^{\text{sec}}) \right]; \quad \sum_{i=1}^{N} z_i q_i^{\text{sec}} \leq W; \quad z_i \in \boldsymbol{N} \right\} \tag{3}$$

Models (2) and (3) are also unbounded knapsack problems. Model (3) must be solved for each X-segment. Recall that each section in a segment of a TSHB pattern is equivalent to a strip in a segment of a simple pattern (such as the simple T-shape pattern [8]). Arranging the segments in increasing order of their lengths and using the techniques presented in [16] can reduce the computation time greatly.

3.5 The optimal TSHB pattern

For the TX-pattern, assume that division x means that the dividing cut is at position x ($0 \leq x \leq L$), and the related pattern value is $U(x)$. Division x divides the stock sheet into two segments: The X-segment $x \times W$ on the left, and the Y-segment $(L - x) \times W$ on the right. The pattern value is determined as $U(x) = f_X(x, W) + f_Y(L - x, W)$. After considering all possible divisions, the one with maximum pattern value is the optimal division. From the property of normal sizes, only those divisions that are normal segment lengths should be considered, namely $x \in P_{\text{segment}}$. Assume that $g(x)$ is the maximum normal segment size not

larger than x, and V_X is the value of the optimal TX-pattern. We have the following equation:

$$x_Y = g(L - x), \quad U(x) = f_X(x, W) + f_Y(x_Y, W), \quad x \in P_{\text{segment}};$$
$$V_X = \max\{U(x) | x \in P_{\text{segment}}\} \tag{4}$$

Assume that the value of the optimal TY-pattern is V_Y, which can be determined in a similar way. The value of the optimal T-shape pattern is $V = \max(V_X, V_Y)$.

3.6 Non-promising segments

If Model (2) is solved for $x = L$, the values of all Y-segments will be obtained. That is to say, it is sufficient to solve only one large knapsack problem to obtain $f_Y(x, W)$ for all $x \in P_{\text{segment}}$. Model (3) does not have this property. It must be solved for each X-segment. The following discussion indicates that some non-promising X-segments may be skipped in searching for the optimal TX-pattern.

Considering two divisions x_1 and x_2, where $x_1 < x_2$. Assume that $z_1 = g(L - x_1)$ and $z_2 = g(L - x_2)$, then $z_1 \geq z_2$. The following equations hold from Eq. 4 and Model (3):

$$U(x_1) = f_X(x_1, W) + f_Y(z_1, W); \quad U(x_2) = f_X(x_2, W) + f_Y(z_2, W)$$
$$f_X(x_1, W) \leq f_X(x_2, W)$$

From these equations, $U(x_1) \leq U(x_2)$ holds if $f_Y(z_1, W) = f_Y(z_2, W)$. As a result, X-segment $x_1 \times W$ is referred to as a non-promising segment and can be skipped.

3.7 The algorithm for generating the optimal TSHB pattern

The algorithm contains the following steps:

Step 1. Determine all types of normal sizes according to Definitions 1 to 3.
Step 2. Determine the optimal blocks according to Sect. 3.2.
Step 3. Determine the optimal sections from Model (1).
Step 4. Determine the optimal segments from models (2) and (3).
Step 5. Determine the optimal TSHB pattern according to Sect. 3.5.

4 The computational results

To the author's knowledge, there are no algorithms for the TSHB pattern. As mentioned previously, the TSHB pattern generalizes the T-shape pattern [8], and cannot generalize the two-segment pattern [15,16] and the three-stage pattern [4]. This section compares the TSHB pattern with the three-stage pattern and the general pattern. Four groups of benchmark problems are used. For each problem, the algorithm of this paper generates the optimal TSHB pattern, Hifi's algorithm [4] generates the optimal three-stage pattern, and the algorithm in [7] determines the optimal general pattern. These three algorithms are referred to as the TSHBA, 3STAGE, and GENERAL respectively. Let v_{TSHBA} and t_{TSHBA} be the value and computation time of the TSHB pattern respectively. Let v_{3STAGE} and t_{3STAGE} be those of the three-stage pattern, and v_{GENERAL} and t_{GENERAL} be those of the general pattern. The problem scale of the second group is larger than those of the other two groups. The problems of the first two groups are available on the Internet at http://www.laria.u-picardie.fr/hifi/OR-Benchmark/. Once the paper is published, the problems of the other two groups will be made available on the author's homepage (http://www.gxnu.edu.cn/Personal/ydcui/). The three algorithms

were all run on a computer with Pentium 4 CPU, clock rate 2.8 GHz, and main memory 512 MB.

4.1 The computational results of the first group

The first group includes 41 problems whose details are available from [4]. Table 1 shows the computational results, where the mark "▲" denotes that the pattern value is equal to that of the optimal general pattern. The number of problems solved to optimality is 34 for the TSHBA, and 30 for the 3STAGE. Compared with the 3STAGE, the pattern value of the TSHBA is larger in 11 problems, equal in 28 problems, and smaller in 2 problems. On the average, the TSHBA can perform better than the 3STAGE in improving material usage. Both algorithms can give solutions very close to optimal.

The total computation time is 18.845 s for the TSHBA, 31.766 s for the 3STAGE, and 1708.265 s for the GENERAL. The average computation times of the three algorithms are respectively 0.460, 0.775, and 41.665 s. The computation time of the GENERAL is much longer than those of the TSHBA and the 3STAGE. The computation time of the TSHBA is shorter than that of the 3STAGE.

4.2 The computational results of the second group

The second group includes 20 problems whose details are available from [17]. Table 2 shows the computational results. Both the TSHBA and the 3STAGE can give solutions very close to optimal. Compared with the 3STAGE, the solutions of the TSHBA are better in 15 problems, and equal in five problems. The TSHBA performs better in improving material usage than the 3STAGE.

The total computation time is 41.389 s for the TSHBA, 681.942 s for the 3STAGE, and 1841.532 s for the GENERAL. The average computation times of the three algorithms are respectively 2.069, 34.097, and 92.077 s. It should be noted that the first 10 problems are unweighted where the value of each piece equals to its area, and the last 10 problems are weighted where the piece value does not equal to its area. The computation time of the TSHBA is comparable with that of the 3STAGE for unweighted problems, and is much shorter than that of the 3STAGE for weighted problems.

4.3 The computational results of the third and fourth group

The third group includes 50 unweighted problems that are the same as those in the first group of [7]. The sheet size is $3{,}000 \times 1{,}500$. Each problem includes 30 piece types. The length and width of each piece were generated by sampling two integers from the uniform distribution [200, 700]. The value of each piece is equal to the piece area. Table 3 shows the computational results. The total computation time is 42.019 s for the TSHBA, and 61.125 s for the 3STAGE. The average computation times of the two algorithms are respectively 0.840 and 1.223 s. Compared with the 3STAGE, the solutions of the TSHBA are better in 20 problems, equal in 29 problems, and worse in one problem (Problem 18).

The problems in the fourth group are weighted problems. They are the same as those in the third group except the piece values. The piece value is equal to the piece area multiplied by a coefficient uniformly distributed in [0.5, 1.0]. Table 4 shows the computational results. The total computation time is 17.811 s for the TSHBA, and 564.031 s for the 3STAGE. The average computation times of the two algorithms are respectively 0.356 and 11.281 s.

Table 1 The computational results of the problems in the first group

ID	v_{GENERAL}	v_{TSHBA}	v_{3STAGE}	t_{GENERAL}	t_{TSHBA}	t_{3STAGE}
H	12,387	▲	12,348	0.031	0.000	0.000
HZ1	5,226	▲	▲	0.016	0.000	0.000
M1	15,550	▲	15,468	0.000	0.000	0.000
M2	73,176	▲	73,080	0.046	0.000	0.016
M3	147,366	▲	147,054	0.000	0.000	0.000
M4	273,991	▲	▲	0.047	0.00	0.015
M5	590,012	588,315	586,879	0.062	0.016	0.016
B	9,000,000	▲	▲	295.313	2.704	2.906
U1	22,435,030	22,419,021	22,416,630	408.062	1.844	1.500
U2	20,446,684	▲	20,382,215	28.938	0.765	0.563
UU1	246,046	▲	▲	0.047	0.015	0.031
UU2	595,655	▲	▲	0.141	0.031	0.047
UU3	1,089,308	▲	▲	0.156	0.047	0.047
UU4	1,188,638	▲	▲	0.328	0.078	0.125
UU5	1,878,253	▲	▲	0.578	0.125	0.219
UU6	2,951,202	▲	▲	0.250	0.110	0.187
UU7	2,949,043	▲	▲	0.891	0.188	0.328
UU8	3,974,828	3,964,638	▲	0.859	0.203	0.360
UU9	6,117,826	▲	6,110,442	1.000	0.282	0.437
UU10	12,004,474	11,996,982	11,996,982	2.094	0.547	0.797
UU11	13,170,382	13,169,380	13,161,640	958.547	8.203	3.937
HZ2	8,523	▲	8,394	0.000	0.000	0.000
MW1	3,916	▲	▲	0.000	0.000	0.000
MW2	24,950	▲	▲	0.015	0.000	0.016
MW3	39,637	▲	▲	0.000	0.000	0.000
MW4	64,044	▲	▲	0.032	0.015	0.016
MW5	190,937	▲	▲	0.031	0.015	0.015
BW	2,379,786	▲	2,370,620	3.266	0.686	17.313
W1	168,834	168,289	168,289	4.234	1.267	1.156
W2	37,621	▲	▲	0.656	0.281	0.250
UW1	6,696	▲	▲	0.016	0.016	0.016
UW2	9,732	▲	▲	0.047	0.015	0.015
UW3	7,188	6,500	▲	0.031	0.016	0.032
UW4	8,452	▲	▲	0.125	0.062	0.062
UW5	8,398	▲	▲	0.110	0.047	0.063
UW6	6,937	▲	▲	0.156	0.110	0.093
UW7	11,585	▲	▲	0.297	0.156	0.172
UW8	8,088	▲	▲	0.547	0.266	0.297
UW9	7,527	▲	▲	0.390	0.281	0.234
UW10	8,172	▲	▲	0.875	0.438	0.454
UW11	18,200	▲	▲	0.031	0.016	0.031

Table 2 The computational results of the problems in the second group

ID	v_{GENERAL}	v_{TSHBA}	v_{3STAGE}	t_{GENERAL}	t_{TSHBA}	t_{3STAGE}
ATP10	3,591,980	3,591,813	3,591,395	208.172	5.844	27.469
ATP11	4,190,481	4,189,704	4,189,406	241.579	5.391	3.797
ATP12	5,162,097	5,157,782	5,157,280	93.297	1.344	1.922
ATP13	3,498,302	3,498,240	3,496,776	220.719	5.640	7.031
ATP14	4,466,844	4,466,098	4,466,098	124.656	1.859	1.875
ATP15	6,054,955	6,054,230	6,053,226	172.156	2.750	3.000
ATP16	7,573,596	7,573,404	7,571,638	237.359	3.781	3.922
ATP17	4,537,842	▲	4,536,836	253.047	5.844	4.000
ATP18	5,835,996	5,834,516	5,834,516	75.313	1.266	2.782
ATP19	6,833,281	6,832,852	6,831,801	180.062	2.234	2.546
ATP20	5,717,092	▲	5,691,316	3.484	0.735	31.344
ATP21	3,582,310	3,573,678	3,556,714	1.531	0.468	125.796
ATP22	4,190,116	4,185,891	4,179,696	5.969	0.625	1.469
ATP23	3,568,354	3,567,248	3,557,391	5.156	0.609	21.641
ATP24	4,078,132	▲	▲	3.219	0.594	3.457
ATP25	3,546,813	3,538,507	3,531,646	6.141	0.531	373.235
ATP26	2,723,840	2,723,248	2,694,022	1.406	0.359	46.859
ATP27	2,458,038	▲	▲	2.610	0.359	0.594
ATP28	4,113,349	4,112,224	4,112,224	3.672	0.625	5.000
ATP29	3,688,965	▲	3,676,018	1.984	0.531	14.203

The former is only about three percent of the later. Compared with the 3STAGE, the solutions of the TSHBA are better in 11 problems, equal in 36 problems, and worse in three problems.

4.4 Solution to a an example

Table 5 shows the piece data that was taken from a factory that makes passenger cars. There are 49 piece types. The sheet size is $4{,}000 \times 3{,}000$. The value of each blank is equal to its area. Algorithms TSHBA and GENERAL were used to solve the example. The computational results are as follows: $v_{\text{TSHBA}} = v_{\text{GENERAL}} = 12,000,000$, $t_{\text{TSHBA}} = 82.89\,\text{s}$, and $t_{\text{GENERAL}} = 3423.36\,\text{s}$. Both algorithms generated optimal patterns, and the computation time of the TSHBA is much shorter than that of the GENERAL.

5 Conclusions

Although exact algorithms exist for general patterns, the computational complexities of them are unknown. The computational experiences in the literature have indicated that the time required by an exact algorithm to solve large-scale problems may become unaffordable. Two approaches have been used to deal with large-scale problems. The first approach is to apply patterns of a specific type, such as staged patterns, T-shape patterns, and two-segment patterns. The exact algorithms for specific pattern types usually have higher time efficiency, and

Table 3 The computational results of the problems in the third group

ID	v_{TSHBA}	v_{3STAGE}	t_{TSHBA}	t_{3STAGE}	ID	v_{TSHBA}	v_{3STAGE}	t_{TSHBA}	t_{3STAGE}
1	4,492,074	4,491,180	1.109	1.344	26	4,497,306	4,497,306	0.923	1.390
2	4,492,500	4,492,500	0.890	1.266	27	4,490,638	4,490,638	0.609	0.891
3	4,497,594	4,496,682	0.969	1.156	28	4,493,799	4,493,368	0.750	4.015
4	4,492,558	4,490,964	0.922	1.250	29	4,496,904	4,496,904	0.845	1.079
5	4,493,361	4,493,361	0.906	1.047	30	4,498,350	4,498,350	0.875	1.046
6	4,496,824	4,496,695	0.922	1.406	31	4,492,556	4,492,556	0.703	0.969
7	4,492,669	4,486,588	0.625	0.953	32	4,492,959	4,490,720	0.828	1.078
8	4,491,640	4,491,640	0.765	1.031	33	4,488,720	4,488,720	0.671	0.969
9	4,494,636	4,492,875	0.813	1.157	34	4,492,631	4,492,528	0.767	1.188
10	4,496,072	4,496,072	1.078	1.375	35	4,493,946	4,493,946	0.687	0.921
11	4,491,138	4,491,138	0.780	1.250	36	4,495,380	4,495,380	0.953	1.204
12	4,493,506	4,493,506	0.829	1.140	37	4,492,172	4,492,172	0.969	1.156
13	4,490,166	4,490,166	0.828	1.016	38	4,492,964	4,492,964	0.906	1.094
14	4,496,160	4,493,740	0.891	1.141	39	4,492,234	4,492,234	0.797	1.031
15	4,498,844	4,497,501	0.969	1.281	40	4,492,576	4,490,549	0.563	0.828
16	4,497,000	4,495,572	0.718	1.047	41	4,496,185	4,496,185	0.827	1.172
17	4,490,865	4,490,865	0.656	0.953	42	4,495,964	4,495,230	1.094	1.187
18	4,491,790	4,492,768	0.845	1.359	43	4,492,347	4,492,347	1.188	1.250
19	4,494,660	4,494,061	1.125	1.219	44	4,492,378	4,492,378	0.813	0.985
20	4,495,538	4,495,538	0.859	1.937	45	4,491,417	4,491,417	0.750	1.093
21	4,491,848	4,490,944	0.734	1.235	46	4,494,330	4,494,330	0.907	1.141
22	4,488,809	4,488,064	0.609	0.812	47	4,496,259	4,492,869	0.812	1.141
23	4,496,100	4,496,100	0.922	1.782	48	4,494,828	4,494,090	1.000	1.937
24	4,493,346	4,491,184	0.843	1.156	49	4,493,488	4,493,488	0.860	1.094
25	4,494,030	4,494,030	0.687	0.969	50	4,491,086	4,491,086	0.718	0.984

may be appropriate for practical applications. The second approach is to develop heuristics for general patterns. These heuristics may also be applicable in practice.

The TSHBA can act as an exact algorithm to generate optimal TSHB patterns. As a specific pattern type, the TSHB pattern can yield more efficient material usage than two-stage pattern and the classic T-shape pattern. Although it cannot generalize the three-stage pattern, the computational results of this paper indicate that for most problems, the material usage of the TSHB pattern is higher than that of the three-stage pattern. The TSHB pattern may also yield better material usage than the two-segment pattern that is generalized by the three-stage pattern.

The TSHBA can also act as a heuristic. The computational results indicate that the values of the optimal TSHB patterns are very close to those of the optimal general patterns. The computation time of the TSHBA is much shorter than that of the GENERAL [7]. Compared with the 3STAGE [4], the average computation time of the TSHBA is shorter for unweighted problems, and is much shorter for weighted problems.

This paper provides a research area for the cutting and packing community. Although the paper presents an algorithm for generating optimal TSHB patterns, and the computation time

Table 4 The computational results of the problems in the fourth group

ID	v_{TSHBA}	v_{3STAGE}	t_{TSHBA}	t_{3STAGE}	ID	v_{TSHBA}	v_{3STAGE}	t_{TSHBA}	t_{3STAGE}
1	4,288,132	4,288,132	0.422	42.031	26	4,423,971	4,423,971	0.359	4.016
2	4,331,992	4,336,733	0.3910	17.157	27	4,386,609	4,341,723	0.298	0.422
3	4,391,908	4,391,908	0.3750	1.281	28	4,203,619	4,186,933	0.343	3.828
4	4,389,770	4,389,770	0.3750	1.859	29	4,273,564	4,273,564	0.344	0.703
5	4,305,809	4,316,162	0.4060	33.844	30	4,260,020	4,260,020	0.359	8.219
6	4,328,627	4,328,627	0.360	0.828	31	4,390,372	4,390,372	0.313	0.578
7	4,353,716	4,353,716	0.296	4.625	32	4,402,439	4,402,439	0.343	21.203
8	4,319,984	4,318,801	0.376	0.969	33	4,245,930	4,232,260	0.313	6.657
9	4,323,212	4,323,212	0.359	260.594	34	4,279,744	4,279,744	0.328	1.890
10	4,327,333	4,327,333	0.437	14.578	35	4,274,391	4,274,391	0.328	1.313
11	4,310,695	4,310,695	0.359	6.094	36	4,340,758	4,340,758	0.375	6.640
12	4,178,060	4,173,588	0.345	30.984	37	4,385,565	4,385,565	0.375	28.516
13	4,295,478	4,295,478	0.406	0.531	38	4,314,348	4,320,956	0.359	2.422
14	4,281,648	4,281,648	0.422	0.500	39	4,254,618	4,254,618	0.328	2.297
15	4,334,688	4,334,688	0.359	1.141	40	4,201,616	4,201,616	0.312	0.812
16	4,413,317	4,396,440	0.360	5.765	41	4,366,725	4,366,725	0.360	0.500
17	4,298,987	4,298,987	0.343	5.547	42	4,286,672	4,286,672	0.359	1.141
18	4,392,600	4,392,600	0.344	0.422	43	4,304,300	4,304,300	0.375	9.359
19	4,391,624	4,387,800	0.344	0.672	44	4,398,526	4,398,526	0.344	0.485
20	4,330,513	4,316,705	0.359	6.734	45	4,374,096	4,374,096	0.312	4.484
21	4,281,328	4,281,328	0.344	0.547	46	4,218,840	4,218,840	0.375	0.906
22	4,395,975	4,395,975	0.312	0.625	47	4,323,138	4,311,156	0.390	4.625
23	4,305,795	4,300,815	0.391	7.360	48	4,398,880	4,398,880	0.328	0.735
24	4,346,817	4,330,720	0.359	0.578	49	4,300,400	4,300,400	0.3440	3.812
25	4,367,335	4,367,335	0.375	1.906	50	4,373,114	4,373,114	0.328	1.281

Table 5 The piece data of the example ($l_i \times w_i$)

990 × 235, 220 × 113, 288 × 250, 1,160 × 868, 1,000 × 868, 868 × 598, 1,160 × 258, 1,000 × 96, 1,000 × 152, 185 × 160, 1,183 × 521, 815 × 521, 1,012 × 571, 455 × 180, 540 × 192, 555 × 210, 253 × 143, 442 × 420, 240 × 90, 330 × 182, 800 × 133, 800 × 403, 803 × 333, 778 × 116, 778 × 78, 1,200 × 241, 1,020 × 262, 1,300 × 121, 1,300 × 83, 800 × 120, 700 × 133, 1,200 × 85, 480 × 480, 965 × 705, 800 × 390, 680 × 185, 840 × 410, 800 × 73, 800 × 40, 930 × 78, 930 × 50, 600 × 88, 1,200 × 146, 822 × 465, 876 × 318, 755 × 147, 460 × 180, 270 × 117, 215 × 145

is reasonable for some practical applications, there are still other cases where high efficiency algorithms may be required. Future researchers can construct more efficient algorithms to meet various requirements coming from practice.

For practical applications, the demand of the pieces is usually constrained, that is, there is an upper bound on the number of pieces of each type that can be used in the pattern. The linear programming (LP) approach is often applied to solve the cutting problem when the demand is relatively large [8,15]. It obtains the solution through iteration. In each cycle of the iteration,

an algorithm of the UTDC problem must be utilized to generate an unconstrained cutting pattern. A large number of cutting patterns must be generated before the LP approach finds a solution close to optimal. This number may reach 100–1,000s. Recall that the computation time for the GENERAL to generate a cutting pattern may reach 100–1,000s seconds. Exact algorithms may be inadequate because of their long computation time. The TSHBA may be used as a candidate for solving the UTDC when applying the LP approach, because it can generate patterns close to optimal in much shorter time.

Acknowledgements This research is supported in part by the National Natural Science Foundation Projects (60763011 and 60752001) and Guangxi Science Foundation (0728100).

References

1. Cui, Y., Zhang, X.: Two-stage general block patterns for the two-dimensional cutting problem. Comput. Oper. Res. **34**, 2882–2893 (2007)
2. Seong, G.G., Kang, M.K.: A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems. Oper. Res. Lett. **31**, 301–307 (2003)
3. Hifi, M., Zissimopoulos, V.: A recursive exact algorithm for weighted two-dimensional cutting. Eur. J. Oper. Res. **91**, 553–564 (1996)
4. Hifi, M.: Exact algorithms for large-scale unconstrained two and three staged cutting problems. Comput. Optim. Appl. **18**, 63–88 (2001)
5. Gilmore, P.C., Gomory, R.E.: Multistage cutting problems of two and more dimensions. Oper. Res. **13**, 94–119 (1965)
6. Beasley, J.E.: Algorithms for unconstrained two-dimensional guillotine cutting. J. Oper. Res. Soc. **36**, 297–306 (1985)
7. Cui, Y., Wang, Z., Li, J.: Exact and heuristic algorithms for staged cutting problems. J. Eng. Manufacture **219**(B2), 201–208 (2005)
8. Cui, Y.: Generating optimal T-shape cutting patterns for rectangular blanks. J. Eng. Manufacture **218**, 857–866 (2004)
9. Agrawal, P.K.: Minimizing trim loss in cutting rectangular blanks of a single size form a rectangular sheet using orthogonal guillotine cuts. Eur. J. Oper. Res. **64**, 410–422 (1993)
10. Tarnowski, A.G., Terno, J., Scheithauer, G.: A polynomial time algorithm for the guillotine pallet-loading problem. INFOR **32**, 275–287 (1994)
11. Arslanov, M.Z.: Continued fractions in optimal cutting of a rectangular sheet into equal small rectangles. Eur. J. Oper. Res. **125**, 239–248 (2000)
12. Cui, Y., Zhou, R.: Generating optimal cutting patterns for rectangular blanks of a single size. J. Oper. Res. Soc. **53**, 1338–1346 (2002)
13. Cui, Y.: Dynamic programming algorithms for the optimal cutting of equal rectangles. Appl. Math. Model. **29**, 1040–1053 (2005)
14. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
15. Cui, Y., He, D., Song, X.: Generating optimal two-section cutting patterns for rectangular blanks. Comput. Oper. Res. **33**, 1505–1520 (2006)
16. Fayard, D., Hifi, M., Zissimopoulos, V.: An efficient approach for large-scale two-dimensional guillotine cutting stock problems. J. Oper. Res. Soc. **49**, 1270–1277 (1998)
17. Alvarez-Valdes, R., Parajon, A., Tamarit, J.M.: A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. Comput. Oper. Res. **29**, 925–947 (2002)