Simple heuristic for the constrained two-dimensional cutting problem

Y Cui* and **Q Chen**
School of Computer, Electronics and Information, Guangxi University, Nanning, China

The manuscript was received on 30 July 2011 and was accepted after revision for publication on 10 August 2011.

DOI: 10.1177/0954405411421996

Abstract: This paper presents a heuristic for the constrained two-dimensional cutting problem in which a guillotine divides a plate into rectangular pieces. The objective of the proposed heuristic is to maximize the pattern value (that is, the total value of the pieces produced from the plate) while observing the constraint that the number produced of a piece can not exceed the demand for that piece. The algorithm uses a simple recursion approach to consider a set of cutting patterns with specified geometric features, and uses a bound technique to discard unpromising branches. It can give solutions competitive with those of other heuristic algorithms. Its solutions to some benchmark instances are better than those currently reported in the literature.

Keywords: cutting stock, constrained two-dimensional cutting, guillotine cuts

1 INTRODUCTION

Optimization algorithms are widely used in manufacturing industry to solve practical problems [**1–3**] such as the two-dimensional (2D) cutting problem [**3**]. This paper presents a simple heuristic for the following constrained 2D cutting (CTDC) problem. A guillotine is used to divide stock plate $L \otimes W$ (length $\otimes$ width) into m different types of rectangular pieces so as to maximize the pattern value (the total value of the pieces produced from the plate), where the ith $(i = 1, \cdots, m)$ type has length l_i, width w_i, value c_i , and demand d_i; the frequency of each type should not exceed the demand for that type. It is assumed that both the plate and item sizes are integers. In this paper P is used to denote a cutting pattern, V^* the value of the optimal pattern, N the set of natural numbers, and z_i the frequency of piece type i $(i = 1, \cdots, m)$. The CTDC problem can then be formulated as

$$V^* = \max\left(\sum_{i=1}^{m} c_i z_i\right) \quad z_i \leqslant d_i \quad \text{and} \quad z_i \in N, \quad i = 1, \cdots, m$$

**Corresponding author: School of Computer, Electronics and Information, Guangxi University, Nanning, 530004, China email:ydcui@263.net*

Exact CTDC algorithms [**4–8**] are not a practical method to solve this problem because the computation cost tends to become exorbitant for medium or large-scale instances. Heuristics [**9–13**] are often used to extend the size of the problems that can be treated using this approach: these include an algorithm based on a strip generation procedure [**9**], a tabu search-based algorithm [**10**], an algorithm that uses dynamic programming and hill-climbing techniques [**11**], a recursive algorithm [**12**], and an algorithm based on dynamic programming and and/or graph search [**13**]. The schema of these algorithms will not be explained further in this short communication. Algorithms for strip packing may also be of use as a source of ideas for the design of CTDC algorithms. The interested reader is referred to [**14**] and [**15**].

The algorithm presented in this paper has the following advantages.

1. It is simple to code. This will allow its use in practical applications.
2. It is much faster than existing algorithms that yield comparable solutions.
3. It is competitive with other algorithms in solution quality. For some benchmark instances the algorithm can yield solutions better than the best ones obtained to date using a heuristic.

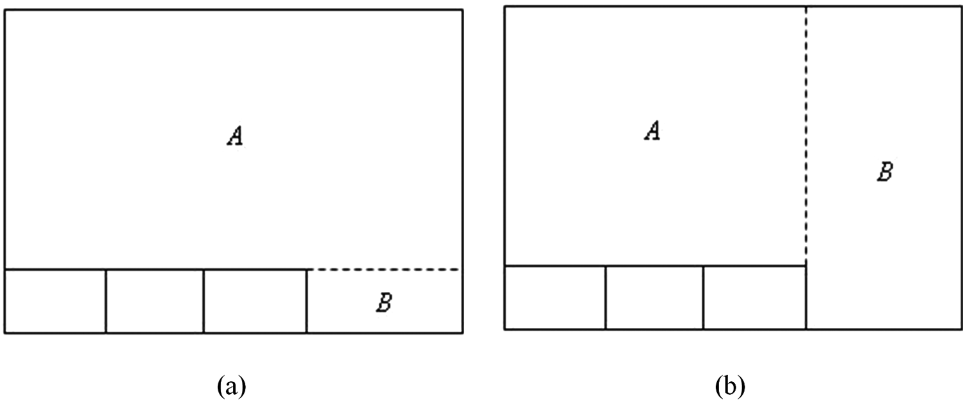


Fig. 1 Patterns related with a row of k pieces of type i (a) pattern P_{RH}^{ik}, and (b) pattern P_{RV}^{ik}

4. It has the all-capacity property that is useful in the solution of the 2D cutting stock (TDCS) problem. This will be further discussed in section 2.

The proposed algorithm is described in section 2. The results of computational experiments are presented in section 3 and conclusions are drawn in section 4.

2 THE ALGORITHM

Several pieces of the same type are placed at the (left-bottom) corner of a sub-plate $x \otimes y$, $x = 0, \cdots, L$ and $y = 0, \cdots, W$. These pieces are referred to as the main pieces and the related type is denoted as the main type. The main pieces form either a row (Fig. 1), Figs 1(a) and (b), or a column (Fig. 2), Figs 2(a) and (b). The number of main pieces of type i in the row is between one and $\min\{\lfloor x/l_i \rfloor, d_i\}$, and that in the column is between one and $\min\{\lfloor y/w_i \rfloor, d_i\}$. It should be noted that the patterns of this type form a superset of the patterns used in [**12**], because the former type allows several main pieces be placed at the corner whereas the later type allows only one main piece.

When k, the number of main pieces, is specified, the following four pattern types should be considered (see Figs 1 and 2): P_{RH}^{ik}, P_{RV}^{ik}, P_{CH}^{ik}, and P_{CV}^{ik}, where the superscript ik denotes that k pieces of type i are placed at the corner of the sub-plate; subscript R denotes the row of the main pieces, and C denotes the column; subscripts H and V denote that the unoccupied region is divided horizontally (Figs 1(a) and 2(a)) and vertically (Figs 1(b) and 2(b)), respectively. For example, P_{RH}^{ik} indicates that a row of k pieces of type i is placed at the corner and the unoccupied region is divided horizontally into two small sub-plates A and B for further consideration (Fig. 1(a)). The patterns are referred to as extended block patterns because they form a superset of the simple block patterns [**12**, **16**], in which the number of main pieces for a sub-plate is restricted to one.

Let $I(x, y) = \{i | x \geqslant l_i, y \geqslant w_i, 1 \leqslant i \leqslant m\}$. Let $F(x, y)$ be the value of sub-plate $x \otimes y$, that is, the total value of the items included. The following recursion determines $F(x, y)$

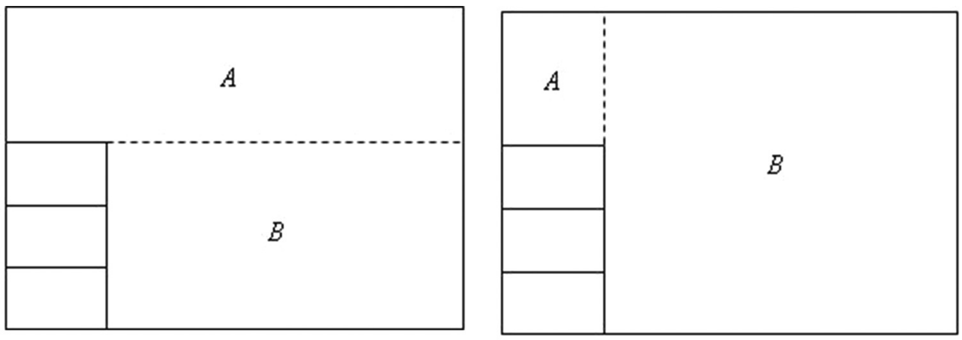
$$F(x,y) = \max\begin{cases} F(x-1,y), F(x,y-1) \\ V_{\mathrm{RH}}^{ik}, V_{\mathrm{RV}}^{ik} | i \in I(x,y), k = 1, \cdots, \min\{\lfloor x/l_i \rfloor, d_i\} \\ V_{\mathrm{CH}}^{ik}, V_{\mathrm{CV}}^{ik} | i \in I(x,y), k = 1, \cdots, \min\{\lfloor y/w_i \rfloor, d_i\} \end{cases} \tag{1}$$

where V_{RH}^{ik}, V_{RV}^{ik}, V_{CH}^{ik} , and V_{CV}^{ik} are the values of patterns P_{RH}^{ik}, P_{RV}^{ik}, P_{CH}^{ik}, and P_{CV}^{ik}, respectively. The initial value of $F(x, y)$ is set to be the largest one out of $F(x-1, y)$ and $F(x, y-1)$. Then four pattern types are considered for improvement.

The frequency of piece type j in sub-plate $x \otimes y$, $j = 1, \cdots, m$ is denoted as $n(x, y, j)$. Let $x_{\mathrm{A}} \otimes y_{\mathrm{A}}$ and $x_{\mathrm{B}} \otimes y_{\mathrm{B}}$ be the sizes of sub-plates A and B (see Figs 1 and 2), respectively. Before considering a pattern type, the following two formulas should be checked

$$\begin{aligned} &\text{value promising (VP)}: \quad kc_i + F(x_{\mathrm{A}}, y_{\mathrm{A}}) + F(x_{\mathrm{B}}, y_{\mathrm{B}}) > F(x,y) \\ &\text{frequency feasible (FF)}: \quad k + n(x_{\mathrm{A}}, y_{\mathrm{A}}, i) + n(x_{\mathrm{B}}, y_{\mathrm{B}}, i) \leqslant d_i \end{aligned}$$

where $kc_i + F(x_{\mathrm{A}}, y_{\mathrm{A}}) + F(x_{\mathrm{B}}, y_{\mathrm{B}})$ is the upper bound of the related pattern value, and $k + n(x_{\mathrm{A}}, y_{\mathrm{A}}, i) + n(x_{\mathrm{B}}, y_{\mathrm{B}}, i)$ is the frequency of type i. A VPFF (combined VP and FF) pattern will be considered by the following function that improves $F(x, y)$.



(a) (b)

Fig. 2 Patterns related with a column of k pieces of type i (a) pattern P_{CH}^{ik}, and (b) pattern P_{CV}^{ik}

$\mathrm{ImprovePat}(x, y, x_A, y_A, x_B, y_B, k, i)$:

1 *Let* $b_j = \min\{d_j, n(x_A, y_A, j) + n(x_B, y_B, j)\}$, $j = 1, \cdots, m$. *Let* $b_i = k + b_i$.

2 *Let* $V = c_1 b_1 + \cdots + c_m b_m$. *Skip the current branch if* $V \le F(x, y)$.

3 *Let* $F(x, y) = V$. *Let* $n(x, y, j) = b_j$, $j = 1, \cdots, m$.

Line 1 counts the frequency of the different pieces and any surplus is taken to be waste. Lines 2 and 3 determine the value of the current branch and update the solution if an improvement is obtained.

The presented algorithm is designed according to recursion (1) and is referred to as the heuristic for constrained extend block patterns (HCEB). It is a greedy algorithm that can be written in pseudo-code as follows for the case where $l_{\min} = \min\{l_i\}$ and $w_{\min} = \min\{w_i\}$.

Line 1 determines the solutions related with $x < l_{\min}$ and $y < w_{\min}$. Lines 2 and 3 indicate that the sub-plates are considered according to the ascending order of their sizes. Lines 4 to 6 set the initial solution of sub-plate $x \otimes y$ to be the better one between the two solutions of smaller sub-plates $(x - 1) \otimes y$ and $x \otimes (y - 1)$. Lines 9 to 14 consider the four sub-pattern types to improve the solution.

Considering only normal sizes is sufficient in designing CTDC algorithms [**11**, **13**]. The HCEB

1 *Let* $F(x, y) = 0$, $n(x, y, j) = 0$, $j = 1, \cdots, m$, $x = 0, \cdots, l_{\min} - 1$, $y = 0, \cdots, w_{\min} - 1$.

2 *For* $x = l_{\min}$ *to* L

3 *For* $y = w_{\min}$ *to* W

4 *If* $F(x-1, y) \ge F(x, y-1)$ *then let*

5 $F(x, y) = F(x-1, y)$, $n(x, y, j) = n(x-1, y, j)$, $j = 1, \cdots, m$;

6 *Else let* $F(x, y) = F(x, y-1)$, $n(x, y, j) = n(x, y-1, j)$, $j = 1, \cdots, m$.

7 *For* $i = 1$ *to* m

8 *Skip type* i *if* $x < l_i$ *or* $y < w_i$.

9 *For* $k = 1$ *to* $\min\{\lfloor x / l_i \rfloor, d_i\}$

10 *If* P_{RH}^{ik} *is VPFF then* $\mathrm{ImprovePat}(x, y, x, y - w_i, x - kl_i, w_i, k, i)$.

11 *If* P_{RV}^{ik} *is VPFF then* $\mathrm{ImprovePat}(x, y, kl_i, y - w_i, x - kl_i, y, k, i)$

12 *For* $k = 1$ *to* $\min\{\lfloor y / w_i \rfloor, d_i\}$

13 *If* P_{CH}^{ik} *is VPFF then* $\mathrm{ImprovePat}(x, y, x, y - kw_i, x - l_i, kw_i, k, i)$

14 *If* P_{CV}^{ik} *is VPFF then* $\mathrm{ImprovePat}(x, y, l_i, y - kw_i, x - l_i, y, k, i)$

Table 1 Comparing HCEB with FHZ

Instance	Optimum	FHZ	HCEB	Instance	Optimum	FHZ	HCEB
OF1	2737	2713	Δ	CHW1	2892	2731	Δ
OF2	2690	2586	Δ	CHW2	1860	1740	Δ
W	2721	Δ	Δ	TH1	4620	Δ	Δ
CU1	12 330	12 312	Δ	TH2	9700	9529	Δ
CU2	26 100	25 806	Δ	CW1	6402	Δ	Δ
CU3	16 723	16 608	Δ	CW2	5354	Δ	Δ
CU4	99 495	98 190	Δ	CW3	5689	5148	5623
CU5	173 364	171 651	Δ	CW4	6175	6168	Δ
CU6	158 572	Δ	Δ	CW5	11 659	11 550	11 644
CU7	247 150	246 860	Δ	CW6	12 923	12 403	12 907
CU8	433 331	432 198	Δ	CW7	9898	9484	Δ
CU9	657 055	Δ	Δ	CW8	4605	4504	Δ
CU10	773 772	764 696	Δ	CW9	10 748	10 748	Δ
CU11	924 696	913 387	Δ	CW10	6515	6116	Δ
				CW11	6321	6084	6084

used in the computational tests is enhanced with this property, that is, it considers only normal sizes.

HCEB has the following all-capacity property. Once the solution is obtained for plate $L \otimes W$, the solutions to all sub-plates $x \otimes y$ are also known, $x = 0, \cdots, L$ and $y = 0, \cdots, W$. This property is useful to solve the TDCS problem from the following description. The sequential heuristic procedure (SHP) can be used to solve the TDCS [**17**], especially when the average demand of the piece types is small. During the solution process, the pieces are classified as either fulfilled or unfulfilled, and initially all pieces are unfulfilled. The SHP follows the following steps to solve the TDCS:

Step 1: call the CTDC algorithm to generate a new pattern, using the unfulfilled pieces.
Step 2: update the unfulfilled pieces by moving those fulfilled by the new pattern.
Step 3: repeat until all pieces are fulfilled.

Assume that M plate sizes can be used. Then it is necessary to generate M patterns (each of which is on a plate of specified size) in step 1 to select the new pattern. In doing so the CTDC algorithm must be called M times if it does not have the all-capacity property. When HCEB is used, it is sufficient to generate only one pattern on a pseudo plate whose length and width are the maximum among the plate lengths and widths, all the M patterns are thus obtained because of the all-capacity property. This reduces the required computation time.

3 COMPUTATIONAL RESULTS

Benchmark instances taken from the literature are used to compare HCEB with other heuristic algorithms [**9–13**]. The instances used in sections 3.1 and 3.2 are available from the Library of Instances [**18**], and those in section 3.3 from the ESICUP [**19**]. The features of the instances will not be detailed due to space considerations. The computational tests were performed on a PC with a 2.66 GHz CPU and 3.37 GB of RAM.

3.1 Comparing HCEB with FHZ

FHZ is an algorithm presented in [**9**]. Table 1 shows the computational results of 29 instances, where Δ denotes that the value is the same as the optimum. An instance is unweighted if the piece value is equal to the area; weighted otherwise. The 14 instances in column 1 are unweighted, and the others are weighted. The values of FHZ solutions were obtained from Table 4 of [**10**]. The number of instances solved to optimality is 25 for HCEB, and six for FHZ. The former is much larger than the latter. This indicates that HCEB is more efficient than FHZ in solution quality.

The average computation time of an instance was 0.175 s for HCEB. The computation time for FHZ is not presented since the computer used in that work was not as powerful as the one used in this paper and thus the solution times are not directly comparable in any meaningful sense.

3.2 Comparing HCEB with TS500, TDH2, and REC

Three groups of instances are used to compare HCEB with the algorithms TS500 [**10**], TDH2 [**11**], and REC [**12**]. The solution values of the three algorithms were obtained from [**12**].

The first group included 26 weighted instances. Table 2 shows the computational results. The number of instances solved to optimality is 20 for TS500, 23 for TDH2, 24 for REC, and 14 for HCEB. HCEB is not as efficient as other algorithms in this group of instances.

Table 2 Results for the first group

Instance	Optimum	TS500	TDH2	REC	HCEB
CHW1	2892	Δ	Δ	Δ	Δ
CHW2	1860	Δ	Δ	Δ	Δ
CW1	6402	Δ	Δ	Δ	Δ
CW2	5354	Δ	Δ	Δ	Δ
CW3	5689	Δ	Δ	Δ	5623
CW4	6175	6170	Δ	Δ	Δ
CW5	11 659	11 644	Δ	Δ	11 644
CW6	12 923	Δ	Δ	Δ	12 907
CW7	9898	Δ	Δ	Δ	Δ
CW8	4605	Δ	Δ	Δ	Δ
CW9	10 748	Δ	Δ	Δ	Δ
CW10	6515	Δ	Δ	Δ	Δ
CW11	6321	Δ	Δ	Δ	6084
2	2892	Δ	Δ	Δ	Δ
3	1860	Δ	Δ	Δ	1840
A1	2020	Δ	Δ	Δ	1940
A2	2505	2455	Δ	Δ	2455
STS2	4620	Δ	Δ	Δ	Δ
STS4	9700	Δ	Δ	Δ	Δ
CHL1	8671	8660	8660	8660	8660
CHL2	2326	Δ	Δ	Δ	2292
CHL3	5283	Δ	Δ	Δ	Δ
CHL4	8998	Δ	Δ	Δ	Δ
Hch11	11 303	11 089	11 255	11 235	11 228
Hch12	9954	9918	9953	Δ	9891
Hch19	5240	Δ	Δ	Δ	5220

The second group includes 36 unweighted instances. The computational results are listed in Table 3. The number of instances solved to optimality is 20 for TS500, 28 for TDH2, 28 for REC, and 31 for HCEB. HCEB is the most efficient for this group of experiments.

It should be noted that the value of the optimal solution to Hch15s was reported as 45 361 in the literature [**10–12**]. The value of 45 410 in Table 3 was obtained from HCEB. This indicates that the value of the optimal solution cannot be 45 361, it should be at least 45 410. Figure 3 gives the pattern obtained from HCEB, so that the reader can check for its feasibility.

The third group includes 20 instances, where the first ten are unweighted and the others are weighted. Table 4 shows the computational results. The symbol * in the column Opt./upper means that the reported value denotes the upper bound of the treated instance, without proof of optimality. The bold numbers denote the best solution values obtained from the algorithms. It is seen that the

Table 3 Results for the second group

Instance	Optimum	TS500	TDH2	REC	HCEB
OF1	2737	2713	Δ	Δ	Δ
OF2	2690	2586	Δ	Δ	Δ
W	2721	Δ	Δ	Δ	Δ
CU1	12 330	Δ	Δ	Δ	Δ
CU2	26 100	Δ	Δ	Δ	Δ
CU3	16 723	16 679	Δ	Δ	Δ
CU4	99 495	99 366	Δ	Δ	Δ
CU5	173 364	Δ	Δ	Δ	Δ
CU6	158 572	Δ	Δ	Δ	Δ
CU7	247 150	Δ	Δ	Δ	Δ
CU8	433 331	432 714	Δ	Δ	Δ
CU9	657 055	Δ	Δ	Δ	Δ
CU10	773 772	773 485	Δ	773 485	Δ
CU11	924 696	922 161	Δ	924 311	Δ
2s	2778	Δ	Δ	Δ	Δ
3s	2721	Δ	Δ	Δ	Δ
A1s	2950	Δ	Δ	Δ	Δ
A2s	3535	Δ	Δ	Δ	Δ
STS2s	4653	Δ	Δ	Δ	Δ
STS4s	9770	Δ	Δ	Δ	Δ
CHL1s	13 099	Δ	Δ	Δ	Δ
CHL2s	3279	Δ	Δ	Δ	3266
CHL3s	7402	Δ	Δ	Δ	Δ
CHL4s	13 932	Δ	Δ	Δ	Δ
CHL5	390	Δ	Δ	Δ	Δ
CHL6	16 869	Δ	Δ	Δ	Δ
CHL7	16 881	16 838	Δ	16 840	Δ
Hch13s	12 215	12 208	12 214	12 214	12 214
Hch14s	12 202	11 967	11 993	Δ	11 964
Hch15s	45 410	45 223	45 361	45 313	Δ
Hch16s	61 040	61 002	61 002	Δ	Δ
Hch17s	63 112	62 802	63 029	63 102	63 102
Hch18s	911	904	904	904	876
A3	5451	5436	5436	Δ	Δ
A4	6179	Δ	Δ	Δ	Δ
A5	12 985	12 929	12 976	12 976	Δ

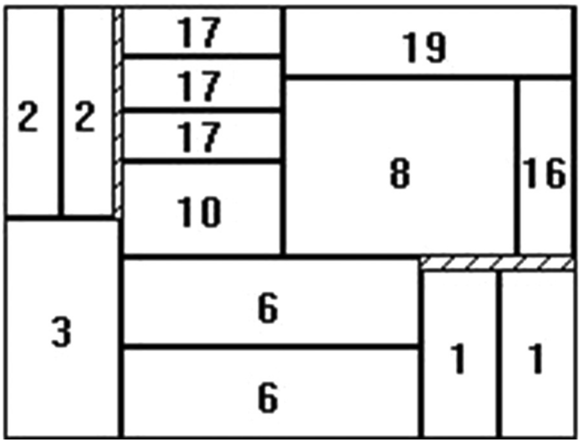


Fig. 3 Solution to Hch15s (value 45 410)

number of best heuristic solutions is two for TS500, 15 for TDH2, 12 for REC, and 13 for HCEB. This indicates that HCEB is competitive with TDH2 and REC, and more efficient than TS500. The HCEB solution to instance ATP34 is shown in Fig. 4. It is the best heuristic solution obtained to date.

Table 5 lists the average computation times for HCEB, TDH2, and REC. The three algorithms were executed on different computers that had different processing powers and thus it is not possible to compare the results in any meaningful manner. While it cannot be confirmed it also cannot be disproved that HCEB is much faster than TDH2 and REC.

3.3 Comparing HCEB with DP_AOG

DP_AOG is an algorithm that was presented in [**13**] and was run on a PC with a 2.99 GHz CPU to solve 450 unweighted instances; the results are described in section 6.2 of [**13**]. The average solution value of HCEB is 9678, and that of DP_AOG is 9679. The former is about 99.99 per cent of the latter. The average computation time is 0.01 s for HCEB and 264.7 s for DP_AOG. Thus, HCEB is competitive with DP_AOG in solution quality, and its computation time is negligible.

4 CONCLUSIONS

When a CTDC algorithm is used jointly with the SHP to solve the TDCS, it must be called numerous times before the solution is obtained; a new pattern being generated in each call. Exact CTDC algorithms are not practical when the problem of interest is medium or large in size since they generally require inordinate amounts of computer time. Heuristic algorithms can be used to simplify these problems and thus allow the scale of the instances that can be treated to be extended. The HCEB presented in this paper is worthy of consideration for the following reasons.

1. It strikes a good balance between solution quality and computation time. Its solutions are often competitive with those of other heuristic algorithms.
2. It is simple to translate the pseudo codes of HCEB into a computer program.
3. When HCEB is used jointly with SHP to solve the TDCS for multiple stock sizes, the computation time can be significantly reduced because of its all-capacity property.

Table 4 Results for the third group

Instance	Opt./upper*	TS500	TDH2	REC	HCEB
ATP30	140 904	140 144	**140 904**	**140 904**	**140 904**
ATP31	824 931*	814 081	**823 976**	823 674	**823 976**
ATP32	38 068	38 030	**38 068**	**38 068**	**38 068**
ATP33	236 818*	234 920	**236 611**	**236 611**	**236 611**
ATP34	362 520*	360 084	361 167	361 197	**361 357**
ATP35	622 644*	620 700	**621 021**	**621 021**	**621 021**
ATP36	130 744	130 338	**130 744**	**130 744**	**130 744**
ATP37	387 276	381 966	387 118	387 118	**387 276**
ATP38	261 698*	259 380	**261 395**	**261 395**	**261 395**
ATP39	268 750	267 168	**268 750**	268 355	**268 750**
ATP40	67 654*	66 362	**67 154**	66 656	**67 154**
ATP41	215 699*	**206 542**	**206 542**	**206 542**	**206 542**
ATP42	34 098*	33 435	33 503	**34 015**	33 566
ATP43	222 570*	214 651	214 651	**214 840**	214 651
ATP44	74 887*	73 410	**73 868**	**73 868**	73 438
ATP45	75 888*	74 691	74 691	**75 808**	74 691
ATP46	151 813*	**149 911**	**149 911**	**149 911**	**149 911**
ATP47	153 747*	148 764	**150 234**	150 043	148 540
ATP48	170 914*	166 927	**167 660**	167 535	167 427
ATP49	226 346*	215 728	**218 388**	217 683	216 749

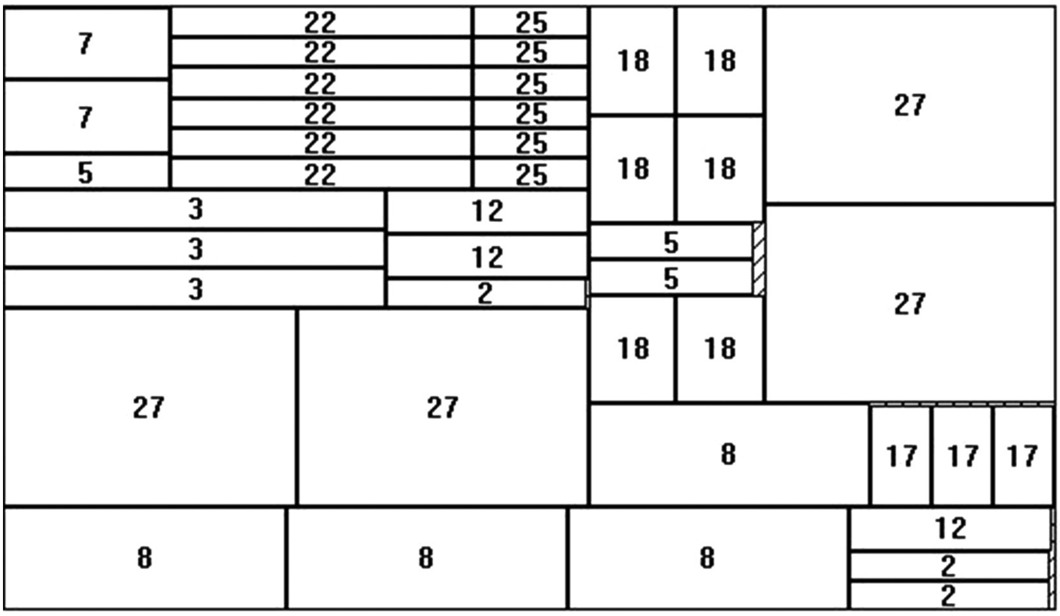


Fig. 4 Solution to ATP34 (value 361 357)

Table 5 Average computation time for an instance (in seconds)

	t – HCEB	t – TDH2	t – REC
First group	0.100	16.96	8.135
Second group	0.086	11.68	5.117
Third group	0.797	75.56	35.219

4. It may be used to provide good initial solution for exact algorithms. This would reduce the computation time required for a solution and extend the scale of the instances that can be treated.

FUNDING

This research was supported in part by the National Natural Science Foundation under grant 61063031.

REFERENCES

1 **Razfar, M. R., Asadnia, M., Haghshenas, M.,** and **Farahnakian, M.** Optimum surface roughness prediction in face milling X20Cr13 using particle swarm optimization algorithm. *Proc. IMechE, Part B: J. Engng Mf.*, 2010, **224**, 1645–1653.

2 **Sedighi, M.** and **Hadi, M.** Preform optimization for reduction of forging force using a combination of neural network and genetic algorithm. *Proc. IMechE, Part B: J. Engng Mf.*, 2010, **224**, 1717–1724.

3 **Cui, Y., Zhang, X.,** and **Wang, Q.** An algorithm for the two-dimensional cutting problem of punched strips with blade length constraint. *Proc. IMechE, Part B: J. Engng Mf.*, 2008, **222**, 1443–1451.

4 **Wang, P. Y.** Two algorithms for constrained two-dimensional cutting stock problems. *Oper. Res.*, 1983, **32**, 573–586.

5 **Viswanathan, K. V.** and **Bagehi, A.** Best-first search methods for constrained two-dimensional cutting stock problems. *Oper. Res.*, 1993, **41**, 768–776.

6 **Cung, V.-D., Hifi, M.,** and **Cun, B. L.** Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *Int. Trans. Oper. Res.*, 2000, **7**, 185–210.

7 **Amaral, A. R. S.** and **Wright, M.** Efficient algorithm for the constrained two-dimensional cutting stock problem. *Int. Trans. Oper. Res.*, 2001, **8**, 3–13.

8 **Vasko, F. J.** and **Bartkowski, C. L.** Using Wang's two-dimensional cutting stock algorithm to optimally solve difficult problems. *Int. Trans. Oper. Res.*, 2009, **16**, 829–838.

9 **Fayard, D., Hifi, M.,** and **Zissimopoulos, V.** An efficient approach for large-scale two-dimensional guillotine cutting stock problem. *J. Oper. Res. Soc.*, 1998, **49**, 1270–1277.

10 **Alvarez-Valdés, R., Parajón, A.,** and **Tamarit, J. M.** A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Comput. Oper. Res.*, 2002, **29**, 925–947.

11 **Hifi, M.** Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems. *J. Comb. Optim.*, 2004, **8**, 65–84.

12 **Chen, Y.** A recursive algorithm for constrained two-dimensional cutting problems. *Comput. Optim. Appl.*, 2008, **41**, 337–348.

13 **Morabito, R.** and **Vitória Pureza, R.** A heuristic approach based on dynamic programming

and and/or-graph search for the constrained two-dimensional guillotine cutting problem. *Ann. Oper. Res.*, 2010, **179**, 297–315.

14 **Araya, I., Neveu, B.,** and **Riff, M. C.** An efficient hyperheuristic for strip-packing problems. In *Adaptive and Multilevel Metaheuristics* (Eds C. Cotta, M. Sevaux, and K. Sorrenson) 2008, pp. 61–76 (Springer Verlag, Germany).

15 **Riff, M. C., Bonnaire, X.,** and **Neveu, B.** A revision of recent approaches for two dimensional strip-packing problems. *Engng Appl. Artif. Intell.*, 2009, **22**, 823–827.

16 **Cui, Y.** Simple block patterns for the two-dimensional cutting problem. *Math. Comput. Model.*, 2007, **45**, 943–953.

17 **Suliman, S. M. A.** A sequential heuristic procedure for the two-dimensional cutting-stock problem. *Int. J. Prod. Econ.*, 2006, **99**, 177–185.

18 Library of Instances, available from http://www.laria.u-picardie.fr/hifi/OR-Benchmark/ (access date May 10, 2011).

19 **ESICUP**, available from http://paginas.fe.up.pt/~esicup/tiki-index.php (access date May 10, 2011).