Journal of Zhejiang University-SCIENCE A (Applied Physics & Engineering)
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Heuristic algorithm based on the principle of minimum total potential energy (HAPE): a new algorithm for nesting problems

Xiao LIU, Jia-wei YE

(*School of Civil and Transportation Engineering, South China University of Technology, Guangzhou 510640, China*)

E-mail: liuxiao@scut.edu.cn; yjw4360@163.com

Received Feb. 18, 2011; Revision accepted Aug. 29, 2011; Crosschecked Sept. 28, 2011

Abstract: We present a new algorithm for nesting problems. Many equally spaced points are set on a sheet, and a piece is moved to one of the points and rotated by an angle. Both the point and the rotation angle constitute the packing attitude of the piece. We propose a new algorithm named HAPE (Heuristic Algorithm based on the principle of minimum total Potential Energy) to find the optimal packing attitude at which the piece has the lowest center of gravity. In addition, a new technique for polygon overlap testing is proposed which avoids the time-consuming calculation of no-fit-polygon (NFP). The detailed implementation of HAPE is presented and two computational experiments are described. The first experiment is based on a real industrial problem and the second on 11 published benchmark problems. Using a hill-climbing (HC) search method, the proposed algorithm performs well in comparison with other published solutions.

Key words: Packing, Cutting, Nesting, Irregular, Heuristic algorithm, Minimum total potential energy
doi:10.1631/jzus.A1100038 **Document code:** A **CLC number:** TH16

1 Introduction

The nesting problem is a two-dimensional cutting and packing problem dealing with irregular shaped pieces. It arises in many industries, e.g., shipbuilding, clothing, textiles, and furniture. The objective is usually to minimize the length of the sheet on which the pieces are placed satisfying the 'no overlapping' constraints. The problems we are going to deal with consider only one large rectangular sheet, having fixed width and infinite length, and the pieces are represented by polygons.

Art (1966) presented the earliest algorithm for nesting problems. They introduced the concept of the 'shape envelope' to describe the feasible no-overlapping positions in which two pieces can be placed. Albano and Sapuppo (1980) used the same concept in nesting problems, but re-named it 'no-fit-polygon' (NFP), a term which then gradually became accepted in the literature. Subsequently, two research directions have been followed. The first aims to study nesting strategies and evaluation criteria. Oliveira *et al*. (2000) presented three nesting strategies (minimizing area, minimizing length, and maximizing overlap) and three evaluation criteria (waste, overlap, and distance). Dowsland *et al*. (1998) proposed a jostling algorithm which can be regarded as a special nesting strategy. Dowsland *et al*. (2002) developed a bottom-left (BL) strategy to calculate the leftmost position for the piece by introducing the NFP. The second research direction concerns the search method of packing orderings. Gomes and Oliveira (2002) proposed a search technique based on a 2-exchange neighborhood generation mechanism. Burke *et al*. (2006) used hill climbing (HC) and tabu local search methods. Gomes and Oliveira (2006) used a simulated annealing algorithm to guide the search over the solution space.

Since the earliest work of Art (1966), the NFP has been regarded as a powerful geometric tool. It was embedded in nearly all the algorithms proposed for nesting problems. Many researchers have created

methods for computing NFPs. Bennell *et al*. (2001) proposed a method which avoids the need to decompose the pieces into suitable forms and the subsequent assembly of the required NFPs. Burke *et al*. (2007) introduced an orbital method for the creation of NFPs.

Although many methods have been proposed, the NFP calculation is still time-consuming. Take the benchmark problem SWIM, for example. Burke *et al*. (2007) declared that it will take 1/66 s to generate an NFP and that the total computing time is only $1/66\times(10\times2)^2$=6.1 s (10 shapes and two rotation angles per shape). While it may take only a short time in this condition, the execution time will increase to $1/66\times(10\times32)^2$=1551.5 s if each piece is allowed to have 32 orientations.

2 A new algorithm: HAPE

In this paper, we propose a new algorithm for nesting problems which replaces time-consuming NFP calculations with a series of polygon overlap tests. Because our proposed algorithm combines the physical similarities of the optimization force leading to a minimum length and the principle of minimum total potential energy, it is named HAPE (Heuristic Algorithm based on the principle of minimum total Potential Energy). Before giving detailed implementation procedures, some concepts and definitions must be introduced.

2.1 Principle of minimum total potential energy

The principle of minimum total potential energy is a fundamental concept which asserts that a structure or body shall deform or displace to a position that minimizes the total potential energy. The total potential energy, Π, is the sum of the elastic strain energy, U, stored in the deformed body and the potential energy, V, of the applied forces:

$$\Pi=U+V. \tag{1}$$

As the pieces are rigid in the packing problems, the elastic energy U is zero, and Eq. (1) should be rewritten as

$$\Pi=V=Gy, \tag{2}$$

where G is the force due to gravity and y is the vertical coordinate of the center of gravity.

The tendency of all weights to lower their position is a basic law of nature. This also applies to nesting problems: the piece always attempts to find an optimal attitude to keep its center of gravity as low as possible.

2.2 Definitions

Definition 1 (Reference point) The reference point is the point around which the piece is rotated. It can be any point in the 2D space. Usually it is suggested that one of the vertices of the polygon be chosen as the reference point (Fig. 1).

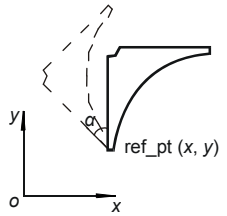

Fig. 1 Reference point and attitude

Definition 2 (Rotation angle) The rotation angle is the angle through which a piece is rotated around its reference point. It can be computed using the following formula:

$$\alpha=2\pi k/\mathrm{RN}, \tag{3}$$

where k=0, 1, …, RN−1, and RN is the rotation number that the piece is allowed to have.

Definition 3 (Attitude) The attitude consists of the above two terms: reference point and rotation angle (Fig. 1).

Definition 4 (Packing points and the distance between packing points) Packing points are the evenly spaced points on the sheet. The vertical or horizontal distance between the packing points is referred to as the packing point distance (PPD) (Fig. 2).

Definition 5 (Feasible attitude) Let a piece slide to a packing point (x, y) and rotate around it through the angle α. If the piece does not intersect with the sheet border or other pieces, the corresponding attitude can be regarded as a feasible attitude. In Fig. 2, two attitudes marked with solid lines are feasible, whereas those marked with dashed lines are infeasible.

To estimate whether an attitude is feasible, an overlap test of polygons must be introduced.

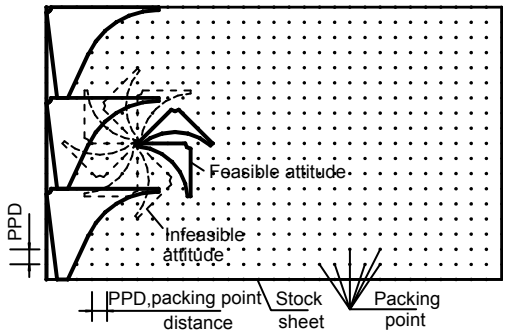

Fig. 2 Packing point and attitude

A feasible attitude does not intersect with the sheet border or other pieces

2.3 Polygon overlap test

When pieces A and B are separated, their attitudes are feasible (Fig. 3a). If piece B slides to the left (Fig. 3b), or rotates counter-clockwise (Fig. 3c), their attitudes become infeasible. Since the piece can be described as a polygon, the piece overlap test can be transformed into a polygon overlap test.

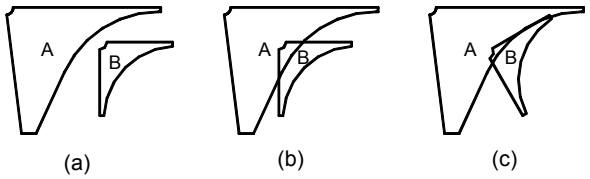

Fig. 3 Feasible and infeasible attitudes

When pieces A and B are separated (a), their attitudes are feasible. If piece B slides to the left (b), or rotates counter-clockwise (c), their attitudes become infeasible

2.3.1 Point-in-polygon test

The point-in-polygon test which queries whether a point lies within a polygon is a fundamental problem in geometry (Sun and Yang, 1995). The sum-of-included-angles algorithm is a basic solution for the point-in-polygon test. Consider a point P_0 and a polygon made up of n vertices P_i where i ranges from 1 to n (Fig. 4). Compute the sum of the angles made between the test point P_0 and each pair of points making up the polygon:

$$\alpha_i = \begin{cases} \angle P_i P_0 P_{i+1}, & i = 1, 2, ..., n-1, \\ \angle P_i P_0 P_1, & i = n, \end{cases} \tag{4}$$

$$\text{sum} = \sum_{i=1}^{n} \alpha_i \,. \tag{5}$$

If the sum is 2π then the point is an interior point; if 0 then the point is an exterior point.

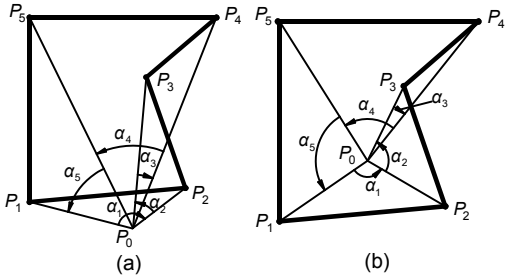

Fig. 4 Summation of angles

(a) $\sum_i \alpha_i = 0$, exterior point; (b) $\sum_i \alpha_i = 2\pi$, interior point

2.3.2 Polygon separation test

Logically, the inverse of a polygon overlap test is a polygon separation test which includes two subtests (suppose there are two polygons A and B): (1) All vertices of polygon A are exterior points of polygon B, and vice versa; (2) All line segments of polygon A do not cross those of B, and vice versa.

Note the following two points: (1) When vertices of A are outside of B, A may contain B completely (Fig. 5a); (2) When the first subtest of the separation tests is satisfied, polygons A and B may still be overlapping (Fig. 5b).

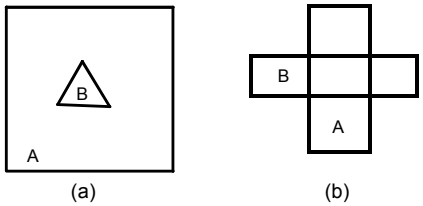

Fig. 5 Overlapped polygons satisfying only one of the subtests

(a) A contains B completely; (b) A and B overlap

2.4 Advance-or-retreat method for polygon touching

In Fig. 6a, A is fixed while B is sliding left to touch A (Fig. 6b). An advance-or-retreat method for polygon touching is described in Fig. 7.

2.5 Geometric center of a polygon

Suppose a polygon (Fig. 8) is composed of n vertices (i=1, 2, …, n) and n edges (l_i, i=1, 2, …, n). Its geometric center (x_c, y_c) can be calculated using Eq. (6):

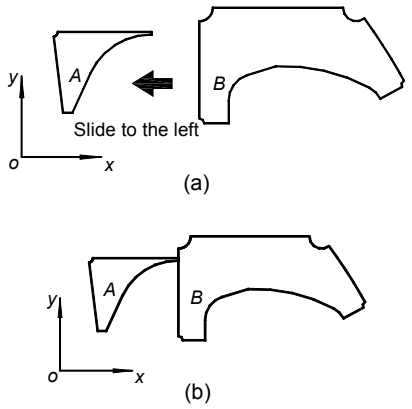

Fig. 6 Contact between pieces A and B
(a) Before contact; (b) After contact

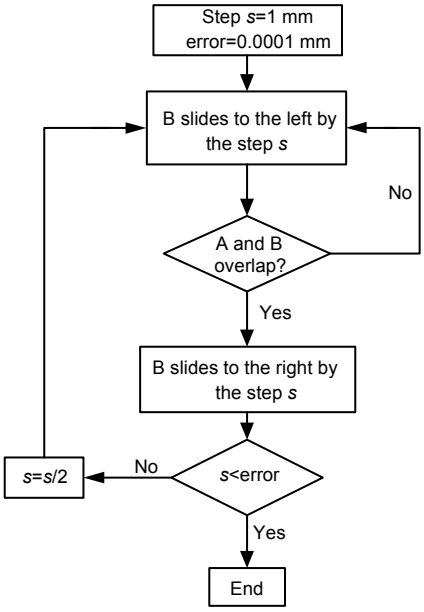

Fig. 7 Flowchart of the advance-or-retreat method for polygon touching

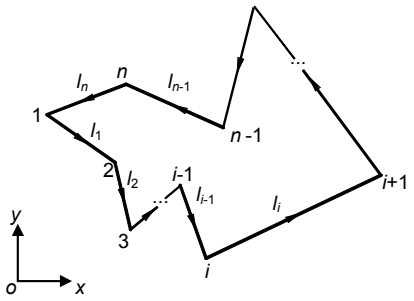

Fig. 8 A polygon which is composed of *n* vertices and *n* edges

$$\begin{cases} x_c = \dfrac{M_y}{A} = \dfrac{1}{3}\dfrac{\sum_{i=1}^{n}(y_{i+1}-y_i)(x_i^2+x_ix_{i+1}+x_{i+1}^2)}{\sum_{i=1}^{n}(x_i-x_{i+1})(y_i+y_{i+1})}, \\ y_c = \dfrac{M_x}{A} = \dfrac{1}{3}\dfrac{\sum_{i=1}^{n}(x_i-x_{i+1})(y_i^2+y_iy_{i+1}+y_{i+1}^2)}{\sum_{i=1}^{n}(x_i-x_{i+1})(y_i+y_{i+1})}, \end{cases} \tag{6}$$

where $x_{n+1}=x_1$, and $y_{n+1}=y_1$. A is the area of the polygon, M_x is the first moment of the polygon in the x direction, M_y is the first moment of the polygon in the y direction, and these three values can be derived using the Green formula (Hibbeler, 2011).

2.6 The implementation procedure of HAPE

The implementation procedure of HAPE is similar to that of the TOPOS algorithm proposed by Oliveira *et al*. (2000), but needs no NFP calculation.

An assumption and a term must be introduced first.

Assumption 1 For the convenience of plotting arrangements, we assume that the direction of the force of gravity is level to the left. Therefore, Eq. (2) becomes

$$\Pi=Gx, \tag{7}$$

where x is the horizontal coordinate of the center of the piece.

Attitude

The attitude of the piece (Fig. 1) can be described in C programming language:

```
struct Attitude
{
double x, y;  /*coordinate of the reference point*/
double alpha;  /*rotation angle*/
};
```

Now, HAPE can be described as follows:

1. The pieces are sorted in order of decreasing area.

2. The 'current' piece is moved to each packing point and rotated around it by RN angles.

3. Calculate the x coordinate of the center of the current piece for each packing attitude according to

Eq. (6).

4. Find the optimal attitude with the smallest x coordinate, and place the current piece on the sheet accordingly.

5. HAPE stops when all the pieces have been placed.

The formal process of HAPE can be stated as the following:

```
Input: Point[0…PPN−1],  // Set PPN (packing point number)
                        // packing points on the sheet (Fig. 2)
       Piece[0…quantity−1]
       // Pieces have been sorted in order of decreasing area
Begin
for (int i=0; i<quantity; i++)
{
   PackOnePieceOnSheet(Piece[i]);
}
End
```

The flowchart of packing one piece on the sheet is shown in Fig. 9.

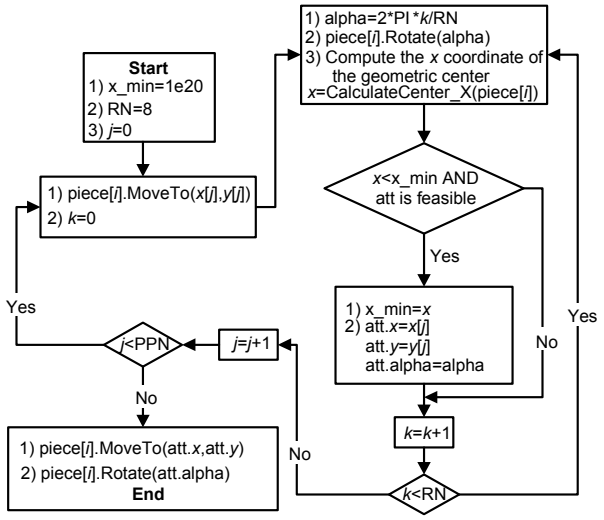

Fig. 9 Flowchart showing the packing of one piece on the sheet

The above flowchart has been simplified for ease of understanding. Some modifications can be added to accelerate the computing. For example, after one piece is placed, it will occupy some packing points which will be infeasible for the next piece. In Fig. 2, the points in the three placed triangular pieces have been eliminated. When the fourth piece is ready to be placed, it can skip these occupied points.

2.7 Hybridizing HAPE with HC

The pre-defined ordering of the pieces does not usually lead to a better layout. Thus, we hybridize HAPE with HC, which is used to search for an optimal ordering (Burke *et al*., 2006). If an improved neighbor is found, it is adopted as the current solution and the search continues. If the neighbor is not an improvement over the current solution, it is discarded and the search continues to find another neighbor. The best solution is returned at the end of the search. We apply operator iOpt (i=1, 2, …, N, where N is the piece quantity) throughout the searching process. 1Opt randomly chooses two pieces and swaps their position in the order; i.e., 1Opt means one swapping operation is applied to the order (Fig. 10). This is extended to NOpt, where N swapping operations are carried out and which is likely to produce a radically different solution, and thus diversify the search. Each operator has a different chance of selection—from 1Opt which has the largest chance of being selected, to NOpt, which has a much lower chance of being selected. This is because the fewer radical operators allow us to concentrate our search, and the highly radical operators, e.g., NOpt, enable us to escape from local optima.

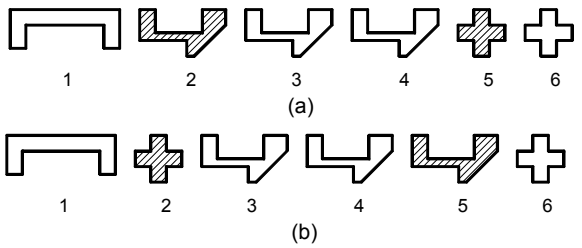

Fig. 10 1Opt: one swapping operation of two randomly chosen pieces from the packing ordering
(a) Before swapping; (b) After swapping

The pseudo codes for HAPE+HC are listed as follows:

```
Input: Pieces, RN, PPD, sheet size, MaxIterationNum
Begin
Current.Ordering=SortOrdering(DecreasingArea);
Current.PackingLength=HAPE(Current.Ordering);
IterationNum=0;
while (IterationNum<MaxIterationNum)
{
   // randomly generate an integer from 1 to N
   Opt=SelectOperator();
```

```
  Neighbor.Ordering=GenerateNeighbor(Current.Ordering,
Opt);
  Neighbor.PackingLength=HAPE(Neighbor.Ordering);
  if (Neighbor.PackingLength<Current.PackingLength)
    Current=Neighbor;
  IterationNum=IterationNum+1;
}
return Current;
End
```

3 Computational experiments

To evaluate the performance of HAPE, we carried out two experiments which were run in the Visual C++ 6.0 environment using a computer with a 2.66 GHz Celeron® CPU and the Windows XP operating system.

3.1 Experiment 1

The first test was taken from the shipbuilding industry and included 66 pieces with four different types of shapes (Fig. 11, the detailed data can be found in Appendix A). The sheet was 3050 mm wide, on which we set a lot of packing points (PPD=100 mm). At each point, the piece was allowed to have eight orientations (RN=8), corresponding to rotation angles of 0°, 45°, 90°, …, 315°. All pieces were placed one by one in order of decreasing area using the HAPE algorithm (Fig. 12).

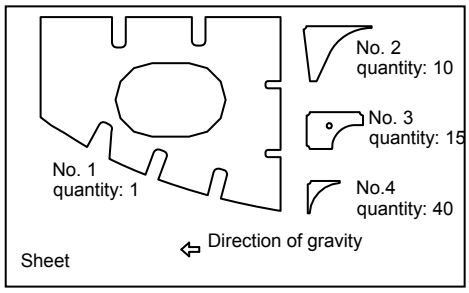

Fig. 11 Sheet and pieces
The sheet is 3050 mm wide, on which a lot of packing points are set with the packing point distance (PPD) being 100 mm. At each point, the piece is allowed to have eight orientations, corresponding to rotation angles of 0°, 45°, 90°, …, 315°

HAPE achieved a compact layout confirming that HAPE is capable of hole-filling and packing concave shaped and jigsaw-type pieces (Fig. 12). Nevertheless, there were still a lot of gaps between pieces. We can eliminate them using the advance-or-retreat method. After the vertical and horizontal sliding of each piece (Fig. 13), the layout will become more compact (Fig. 14).

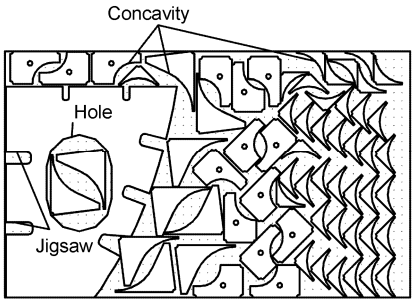

Fig. 12 Gap between pieces
Packing length: 4900 mm; packing density: 61.61%

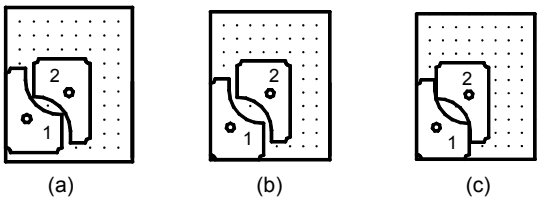

Fig. 13 Eliminating gaps by vertical/horizontal sliding of each piece
(a) Gaps between pieces 1, 2 and the bottom of the sheet; (b) Gap between piece 1 and the bottom is eliminated by vertical sliding of piece 1; (c) Gap between pieces 1 and 2 is eliminated by horizontal sliding of piece 2

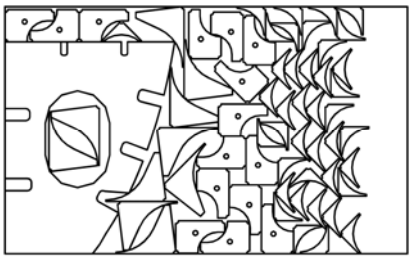

Fig. 14 Layout which becomes more compact after eliminating gaps
Computation time: 0.42 s; packing length: 4521.08 mm; packing density: 66.77%

To achieve a more optimal layout, we used HC as a search mechanism to generate new input orderings for piece placement and HAPE to transform the ordering into a layout. This problem was run for 500 iterations; i.e., there were 500 HAPEs executed during the whole searching procedure. The packing density improved greatly, to 72.45% (Fig. B3).

Obviously, RN and PPD are two important variables that affect the computing speed and packing efficiency. Thus, we made some comparisons under different RNs and PPDs. In Table 1, the computational results are summarized. Due to space limitations, we list the detailed layout results of only PPD=25 mm (Figs. B1–B5).

Larger RNs and smaller PPDs lead to extended execution times (Table 1 and Figs. 15b and 15d). The lines in Fig. 15b are close to straight, which indicates a linear relationship between the execution time and RN. In Fig. 15d five parabolic lines describe the quadratic relationship between the execution time and PPD.

As to the relationship between packing density (PD) and RN/PPD, the situation is complicated. The first conclusion, drawn from Fig. 15c, is that a smaller PPD leads to a larger PD, except for RN=4. But a larger RN does not always produce a more compact layout. PD does not always increase with increasing RN (Fig. 15a). Three PD-RN lines reach a peak at RN=4 and RN=16. The PD can reach a relatively high value at a certain RN, which we call the sweet RN. This problem has two sweet RNs.

Table 1 Packing density and execution time with different RNs and PPDs

RN	Packing length (mm)			Packing density (%)			Execution time (s)		
	PPD=100 mm	50 mm	25 mm	100 mm	50 mm	25 mm	100 mm	50 mm	25 mm
2	4681.39	4391.34	4218.28	64.48	68.74	71.56	114	272	799
4	4397.47	4159.01	4167.96	68.65	72.58	72.43	167	422	1327
8	4391.95	4274.98	4166.79	68.73	70.61	72.45	240	741	2607
16	4361.64	4193.77	4068.06	69.21	71.98	74.20	458	1427	5136
32	4400.55	4221.47	4144.43	68.60	71.51	72.84	866	2724	10278

RN: rotation number of piece; PPD: packing point distance

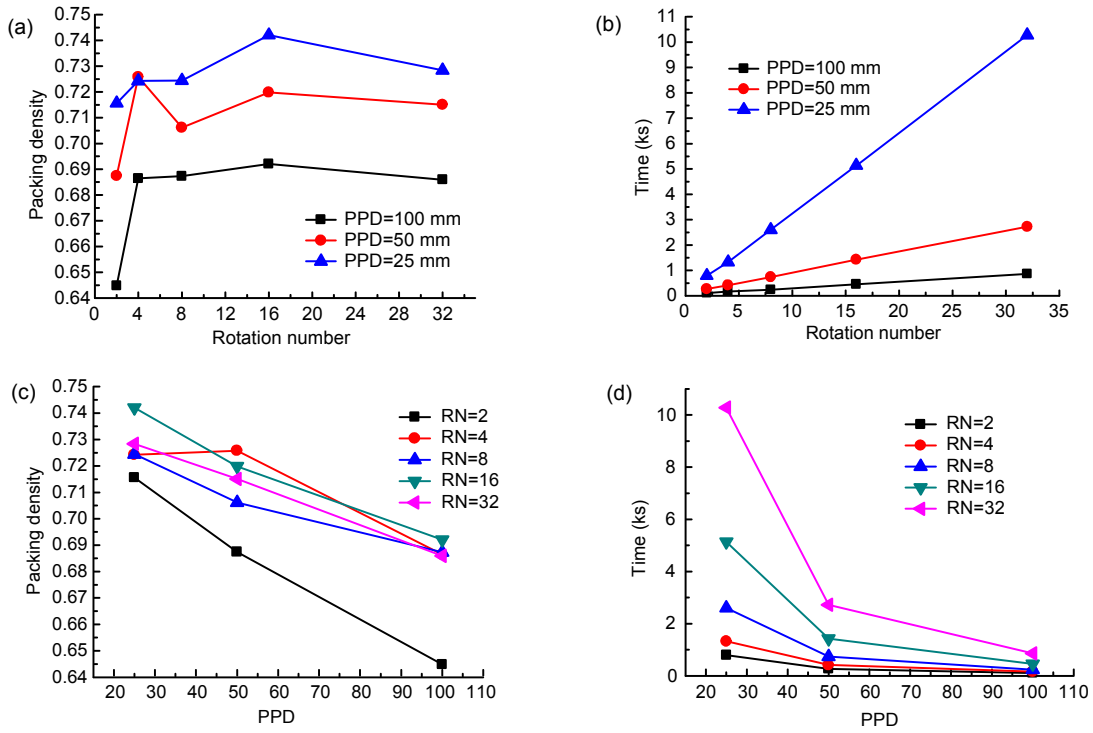

Fig. 15 Packing density and execution time with different rotation numbers of piece (RNs) and packing point distances (PPDs)

(a) Packing density vs. RN; (b) Execution time vs. RN; (c) Packing density vs. PPD; (d) Execution time vs. PPD

3.2 Experiment 2

To assess the performance of HAPE, we gathered 11 well-known problems from the literature, which were downloaded from the EURO Special Interest Group on Cutting and Packing (ESICUP) website http://paginas.fe.up.pt/~esicup/tiki-index.php (Table 2).

Once this website was open to the public, researchers around the world competed to test their algorithms with these standard problems. The reported packing lengths are becoming shorter and shorter (Table 3). Taking Blaz1 for example, there are four different length records from the years 2000 to 2006. The longest record (L_{max}=28.9) was created by Oliveira *et al*. (2000) while the shortest (L_{min}=25.84) was from Gomes and Oliveira (2006).

As in Experiment 1, each problem in Table 2 was run for 500 iterations (HC+HAPE). The packing lengths and execution times are summarized in Table 3 (for details see Appendix B, Figs. B6–B16).

The first observation about HAPE is its high speed which allows a fairly wide search of the solution space. Referring to the last column in Table 3, the shortest execution time for HAPE was 18/500=0.04 s and the longest was 1078/500=2.16 s.

The second observation is the good performance of HAPE in combination with HC. Although the proposed approach does not surpass any of the newly created records, the solutions are located within the reasonable range (except for SHAPES0/SHAPES1). Take problem Blaz1 for example, L_{hh}=28.34$\in$[L_{min}=25.84, L_{max}=28.90]. This can be seen more clearly by defining the relative length, L_i/L_{hh} (L_i=L_{min}, L_{max}, or L_{hh}) (Fig. 16).

Table 2 The 11 benchmark problems from the literature

Reference	Problem name	Piece quantity	Sheet width	Rotation number
Błażewicz *et al.* (1993)	Blaz1	28	15	2
Ratanapan and Dagli (1997)	Dagli	30	60	4
Fujita *et al*. (1993)	Fu	12	38	4
Jakobs (1996)	Jakobs1	25	40	4
Jakobs (1996)	Jakobs2	25	70	4
Marques *et al*. (1991)	Marques	24	104	4
Oliveira *et al*. (2000)	SHAPES0	43	40	1
Oliveira *et al*. (2000)	SHAPES1	43	40	2
Oliveira *et al*. (2000)	SHIRTS	99	40	2
Oliveira *et al*. (2000)	SWIM	48	5752	2
Oliveira *et al*. (2000)	TROUSERS	64	79	2

Table 3 Published length records and our results

Problem	Length 1	Reference	Length 2	Reference	Length 3	Reference	Length 4	Reference	L_{min}	L_{max}	L_{hh}	Execution time (s)
Blaz1	27.30	a	27.20	e	25.84	f	28.90	g	25.84	28.90	28.34	18
Dagli	65.60	b	60.57	e	58.20	f	−	−	58.20	65.60	60.75	219
Fu	34.00	c	32.80	e	31.33	f	−	−	31.33	34.00	33.00	38
Jakobs1	13.20	b	11.86	e	12.00	f	−	−	11.86	13.20	13.00	59
Jakobs2	28.20	b	25.80	e	24.97	f	−	−	24.97	28.20	26.77	183
Marques	83.60	b	80.00	e	78.48	f	−	−	78.48	83.60	82.30	369
SHAPES0	63.00	d	65.00	e	60.00	f	66.75	g	60.00	66.75	67.55	79
SHAPES1	59.00	a	58.40	e	56.00	f	61.00	g	56.00	61.00	64.04	159
SHIRTS	63.13	a	63.00	e	62.21	f	66.44	g	62.21	66.44	65.97	217
SWIM	6568.00	b	6462.40	e	5948.37	f	−	−	5948.37	6568.00	6525.24	805
TROUSERS	245.75	a	243.40	e	242.11	f	263.20	g	242.11	263.17	251.47	1078

L_{hh}: packing length generated by HAPE+HC. a: Gomes and Oliveira (2002); b: Hopper (2000); c: Fujita *et al*. (1993); d: Dowsland and Dowsland (1993); e: Burke *et al*. (2006); f: Gomes and Oliveira (2006); g: Oliveira *et al*. (2000)

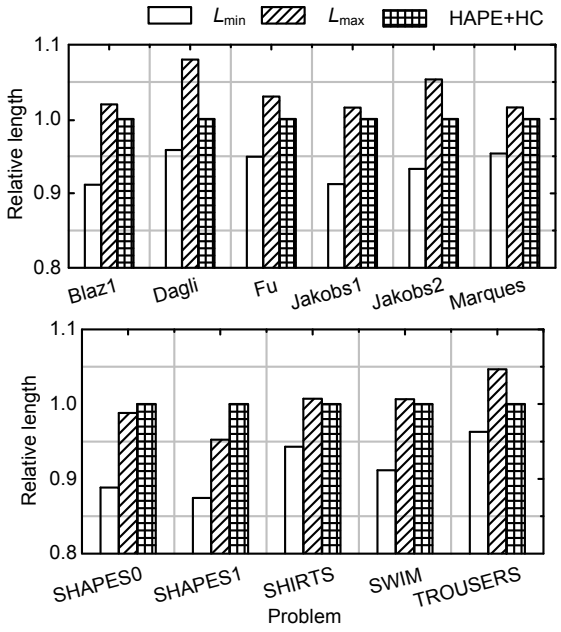

Fig. 16 Relationship between L_{min}, L_{max}, and L_{hh}
Relative length=L_i/L_{hh}, where L_i=L_{min}, L_{max}, or L_{hh}

Table 3 and Fig. 16 show that our method works poorly on SHAPES0/SHAPES1. To improve the packing density, we assigned three larger values to RN, from 4 to 16, and then produced the corresponding layouts (Figs. 17–19). In Fig. 18, the packing length reaches the optimal value 56.66 (very close to L_{min}=56.00 in Table 3) at RN=8, which can be regarded as the sweet RN for problem SHAPES.

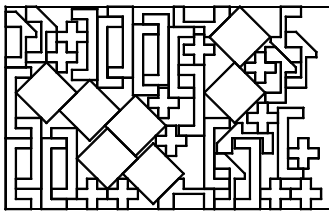

Fig. 17 SHAPES2
HC (500 iterations)+HAPE (RN=4, PPD=0.5). Length=62.09; density=64.26%; execution time=220 s

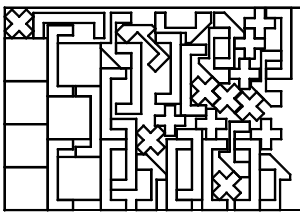

Fig. 18 SHAPES3
HC (500 iterations)+HAPE (RN=8, PPD=0.5). Length=56.66; density=70.42%; execution time=415 s

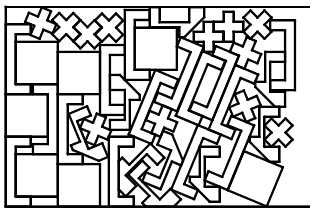

Fig. 19 SHAPES4
HC (500 iterations)+HAPE (RN=16, PPD=0.5). Length=58.93; density=67.71%; execution time=805 s

4 Summary

In this paper we have presented a new algorithm, named HAPE, for nesting problems. This algorithm is quite different from others, because it avoids the time-consuming calculation of NFP by using a geometric technique of an overlap test for polygons. Our computational experiments showed that HAPE is a credible algorithm for nesting problems. HAPE also allows each piece to have up to 32 orientations while the execution time is reasonable. The initial study also demonstrated HAPE's potential for hybridization with other meta-heuristics.

Although the main advantage of HAPE is its ability to rotate the pieces through many angles, it is not suitable to assign a big value to RN without limitation. We suggest using a sweet RN and a reasonably small PPD to obtain a compact packing layout in an acceptable time. But how to find the sweet RN is still not clear. This could be an interesting research question for future studies.

Acknowledgements

The authors would like to thank Guangzhou Wenchong Shipyard Co., Ltd. (GWS) and Guangzhou Shipyard International Co., Ltd. (GSI) for supporting this study. We also thank the anonymous reviewers for their constructive comments.

References

Albano, A., Sapuppo, G., 1980. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *IEEE Transactions on Systems, Man and Cybernetics*, **10**(5):242-248. [doi:10.1109/TSMC.1980.4308483]

Art, J.R.C., 1966. An Approach to the Two Dimensional

Irregular Cutting Stock Problem. Technical Report No. 36.Y08, IBM Cambridge Scientific Center, Massachusetts, USA.

Bennell, J.A., Dowsland, K.A., Dowsland, W.B., 2001. The irregular cutting-stock problem—a new procedure for deriving the no-fit polygon. *Computers & Operations Research*, **28**(3):271-287. [doi:10.1016/S0305-0548(00)00021-6]

Błażewicz, J., Hawryluk, P., Walkowiak, R., 1993. Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, **41**(4):313-325. [doi:10.1007/BF02022998]

Burke, E., Hellier, R., Kendall, G., Whitwell, G., 2006. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, **54**(3):587-601. [doi:10.1287/opre.1060.0293]

Burke, E.K., Hellier, R.S.R., Kendall, G., Whitwell, G., 2007. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, **179**(1):27-49. [doi:10.1016/j.ejor.2006.03.011]

Dowsland, K.A., Dowsland, W.B., 1993. Heuristic Approaches to Irregular Cutting Problems. Technical Report, University College of Swansea, Swansea, UK.

Dowsland, K.A., Dowsland, W.B., Bennell, J.A., 1998. Jostling for position: local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, **49**(6):647-658. [doi:10.1057/palgrave.jors.2600563]

Dowsland, K.A., Vaid, S., Dowsland, W.B., 2002. An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research*, **141**(2):371-381. [doi:10.1016/S0377-2217(02)00131-5]

Fujita, K., Akagji, S., Kirokawa, N., 1993. Hybrid Approach for Optimal Nesting Using a Genetic Algorithm and a Local Minimisation Algorithm. Proceedings 19th Annual ASME Design Automation Conference, Part 1, **65**:477-484.

Gomes, A.M., Oliveira, J.F., 2002. A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, **141**(2):359-370. [doi:10.1016/S0377-2217(02)00130-3]

Gomes, A.M., Oliveira, J.F., 2006. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, **171**(3):811-829. [doi:10.1016/j.ejor.2004.09.008]

Hibbeler, R.C., 2011. Statics and Mechanics of Materials. Prentice Hall, New Jersey, USA, p.261-272.

Hopper, E., 2000. Two-Dimensional Packing Utilising Evolutionary Algorithm and Other Meta-Heuristic Methods. PhD Thesis, University of Wales, Cardiff, UK.

Jakobs, S., 1996. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, **88**(1):165-181. [doi:10.1016/0377-2217(94)00166-9]

Marques, V.M.M., Bispo, C.F.G., Sentieiro, J.J.S., 1991. A System for the Compaction of Two-Dimensional Irregular Shapes Based on Simulated Annealing. Proceedings International Conference on Industrial Electronics, Control and Instrumentation, p.1911-1916. [doi:10.1109/IECON.1991.239050]

Oliveira, J.F., Gomes, A.M., Ferreira, J.S., 2000. TOPOS—a new constructive algorithm for nesting problems. *OR Spectrum*, **22**(2):263-284. [doi:10.1007/s002910050105]

Ratanapan, K., Dagli, C.H., 1997. An Object-Based Evolutionary Algorithm for Solving Irregular Nesting Problems. Proceedings Artificial Neural Networks in Engineering Conference, p.383-388.

Sun, J., Yang, C., 1995. Computer Graphics. Tsinghua University Press, Beijing, China, p.390-391 (in Chinese).

Appendix A: Data set of problem SomeParts from shipbuilding industry

Plate length=5000
Plate width=3050
Part
Name
Part1
Quantity
1
OutLoop

93.589,1651.729	93.589,1050.924
209.909,962.031	354.716,877.905
477.743,798.551	598.471,726.769
732.067,972.383	749.897,991.143
774.718,998.476	799.880,992.415
844.681,968.046	853.856,961.629
861.373,953.331	866.855,943.568
870.027,932.830	870.730,921.656
868.929,910.605	864.714,900.232
842.916,596.559	979.581,532.350
1118.216,472.471	1235.201,425.637
1326.210,669.060	1341.240,690.130
1364.791,700.864	1390.554,698.383
1436.451,681.223	1446.433,676.153
1455.037,668.988	1461.831,660.088
1466.474,649.900	1468.734,638.934
1468.497,627.740	1465.775,616.880
1374.585,372.976	1518.983,321.731
1714.168,259.200	1912.952,202.611
1977.557,454.205	1990.263,476.753
2012.541,489.928	2038.422,490.198
2085.882,478.011	2096.346,474.030
2105.663,467.819	2113.364,459.692
2119.063,450.054	2122.475,439.390
2123.428,428.234	2121.875,417.146
2057.134,165.022	2238.542,121.724
2445.524,78.184	2653.588,40.141
2707.589,31.157	2707.589,616.729
2567.589,616.729	2549.911,624.052
2542.589,641.729	2542.589,669.729
2549.911,687.407	2567.589,694.729

2707.589,694.729 2707.589,1365.729
2567.589,1365.729 2549.911,1373.052
2542.589,1390.729 2542.589,1418.729
2549.911,1436.407 2567.589,1443.729
2707.589,1443.729 2707.589,2139.729
1889.589,2139.729 1889.589,1879.729
1882.890,1854.729 1864.589,1836.428
1839.589,1829.729 1790.589,1829.729
1765.589,1836.428 1747.287,1854.729
1740.589,1879.729 1740.589,2139.729
1021.589,2139.729 1021.589,1859.729
1014.890,1834.729 996.589,1816.428
971.589,1809.729 920.589,1809.729
895.589,1816.428 877.287,1834.729
870.589,1859.729 870.589,2139.729
93.589,2139.729 EndOutLoop
InnerLoop
1709.245,837.960 1909.245,891.550
2055.655,1037.960 2109.245,1237.960
2055.655,1437.960 1909.245,1584.371
1709.245,1637.960 1309.245,1637.960
1109.245,1584.371 962.835,1437.960
909.245,1237.960 962.835,1037.960
1109.245,891.550 1309.245,837.960
EndInnerLoop
EndPart
Part
Name
Part2
Quantity
10
OutLoop
215.943,595.621 286.602,30.347
356.602,30.347 492.443,321.659
540.235,403.064 602.476,474.036
676.947,532.044 760.993,575.022
851.619,601.437 945.595,610.347
955.595,610.347 955.595,630.347
246.602,630.347 245.590,621.991
242.612,614.119 237.840,607.184
231.552,601.590 224.109,597.658
EndOutLoop
EndPart
Part
Name
Part3
Quantity
15
OutLoop
58.095,402.023 56.902,392.965
53.406,384.523 47.843,377.274
40.595,371.712 32.153,368.216
23.095,367.023 23.095,37.023
47.843,26.772 58.095,2.023
298.095,2.023 304.390,68.642
328.131,131.204 367.617,185.228
420.018,226.843 481.581,253.068
547.895,262.023 633.095,262.023
633.095,367.023 608.346,377.274
598.095,402.023
EndOutLoop
InnerLoop
286.799,275.478 261.419,281.271
241.066,265.040 241.066,239.007
261.419,222.775 286.799,228.568
298.095,252.023 EndInnerLoop
EndPart
Part
Name
Part4
Quantity
40
OutLoop
135.869,21.829 155.869,21.829
170.844,105.429 205.515,182.960
257.840,249.858 324.738,302.183
402.269,336.853 485.869,351.829
485.869,371.829 170.869,371.829
160.617,347.080 135.869,336.829
EndOutLoop
EndPart
#FileEnd

Appendix B: Layouts with different RNs and PPDs

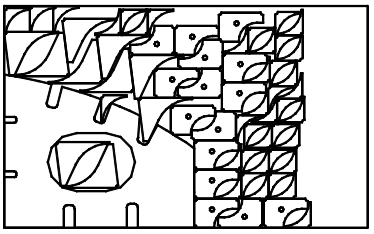

Fig. B1 SomeParts (RN=2, PPD=25)
HC (500 iterations)+HAPE. Length=4218.28; density=71.56%; execution time=799 s

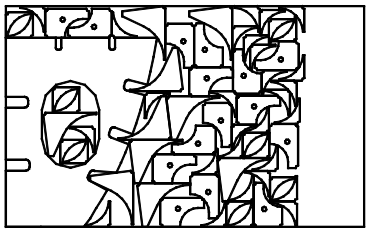

Fig. B2 SomeParts (RN=4, PPD=25)
HC (500 iterations)+HAPE. Length=4167.96; density=72.43%; execution time=1327 s

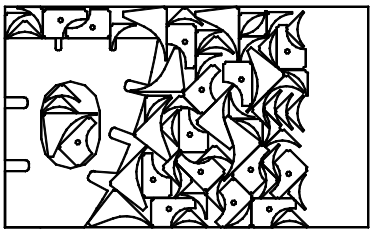

Fig. B3 SomeParts (RN=8, PPD=25)
HC (500 iterations)+HAPE. Length=4166.79; density=72.45%; execution time=2607 s

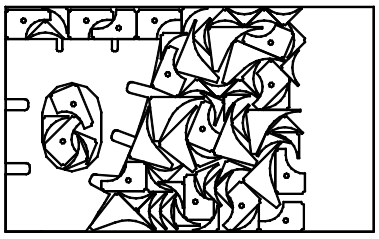

Fig. B4 SomeParts (RN=16, PPD=25)
HC (500 iterations)+HAPE. Length=4068.06; density=74.20%; execution time=5136 s

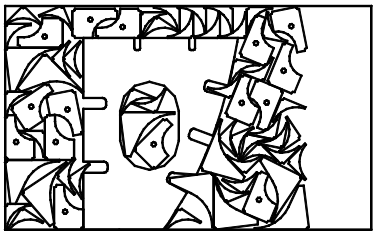

Fig. B5 SomeParts (RN=32, PPD=25)
HC (500 iterations)+HAPE. Length=4144.43; density=72.84%; execution time=10278 s

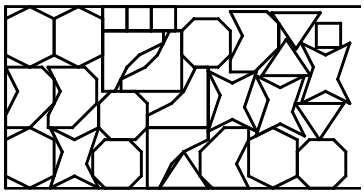

Fig. B6 Blaz1
HC (500 iterations)+HAPE (RN=2, PPD=0.5). Length= 28.34; density=76.21%; execution time=18 s

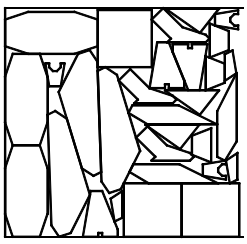

Fig. B7 Dagli
HC (500 iterations)+HAPE (RN=4, PPD=0.5). Length=60.75; density=83.48%; execution time=219 s

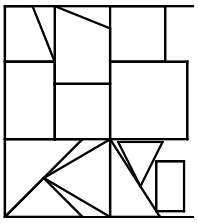

Fig. B8 Fu
HC (500 iterations)+HAPE (RN=4, PPD=0.5). Length=33.00; density=86.36%; execution time=38 s

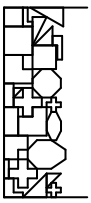

Fig. B9 Jakobs1
HC (500 iterations)+HAPE (RN=4, PPD=0.5). Length=13.00; density=75.38%; execution time=59 s

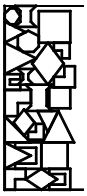

Fig. B10 Jakobs2
HC (500 iterations)+HAPE (RN=4, PPD=0.5). Length=26.77; density=72.11%; execution time=183 s

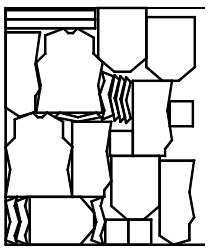

Fig. B11 Marques
HC (500 iterations)+HAPE (RN=4, PPD=0.5). Length= 82.30; density=84.05%; execution time=369 s

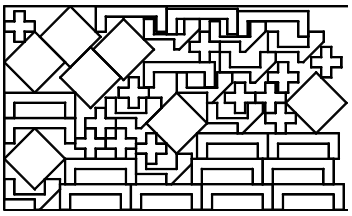

Fig. B12 SHAPES0
HC (500 iterations)+HAPE (RN=1, PPD=0.5). Length=67.55; density=59.06%; execution time=79 s

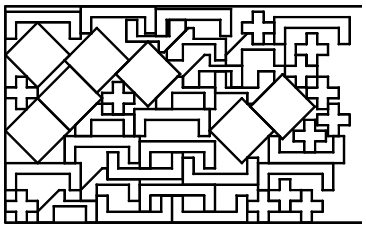

Fig. B13 SHAPES1

HC (500 iterations)+HAPE (RN=2, PPD=0.5). Length=64.04; density=62.31%; execution time=159 s

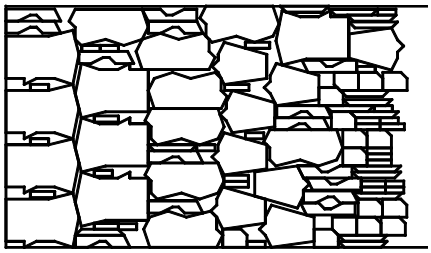

Fig. B14 SHIRTS

HC (500 iterations)+HAPE (RN=2, PPD=0.5). Length=65.97; density=81.86%; execution time=217 s

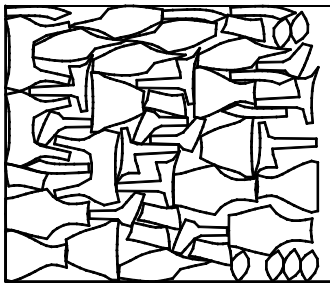

Fig. B15 SWIM

HC (500 iterations)+HAPE (RN=2, PPD=50). Length=6525.24; density=67.79%; execution time=805 s

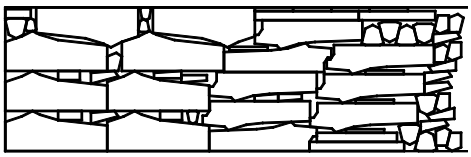

Fig. B16 TROUSERS

HC (500 iterations)+HAPE (RN=2, PPD=0.5). Length=251.47; density=86.61%; execution time=1078 s