Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

HAPE3D—a new constructive algorithm for the 3D irregular packing problem*

Xiao LIU[†‡1], Jia-min LIU[2], An-xi CAO[3], Zhuang-le YAO[1]

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

([2]*School of Information Science and Engineering, Shenyang University of Technology, Shenyang 110870, China*)

([3]*College of Ocean Science and Engineer, Shanghai Maritime University, Shanghai 201306, China*)

[†]E-mail: liuxiao@scut.edu.cn

Received Dec. 8, 2014; Revision accepted Mar. 29, 2015; Crosschecked Apr. 10, 2015

Abstract: We propose a new constructive algorithm, called HAPE3D, which is a heuristic algorithm based on the principle of minimum total potential energy for the 3D irregular packing problem, involving packing a set of irregularly shaped polyhedrons into a box-shaped container with fixed width and length but unconstrained height. The objective is to allocate all the polyhedrons in the container, and thus minimize the waste or maximize profit. HAPE3D can deal with arbitrarily shaped polyhedrons, which can be rotated around each coordinate axis at different angles. The most outstanding merit is that HAPE3D does not need to calculate no-fit polyhedron (NFP), which is a huge obstacle for the 3D packing problem. HAPE3D can also be hybridized with a meta-heuristic algorithm such as simulated annealing. Two groups of computational experiments demonstrate the good performance of HAPE3D and prove that it can be hybridized quite well with a meta-heuristic algorithm to further improve the packing quality.

Key words: 3D packing problem, Layout design, Simulation, Optimization, Constructive algorithm, Meta-heuristics
doi:10.1631/FITEE.1400421 **Document code:** A **CLC number:** TP391.7

1 Introduction

The 3D irregular packing problem belongs to a general class of combinatorial optimization problems which are concerned with packing a set of irregular pieces into one or more large containers to minimize the waste or maximize profit. The 3D irregular packing problem occurs in many applications, namely, container loading, human occupied vehicle (HOV) design (Fig. 1), satellite module layout design (Huo *et al.*, 2006) (Fig. 2), building layout, and 3D laser cutting.

2 Literature review

For the 3D packing problem, most published studies deal with only polyhedrons with regular shapes, such as cubes (Wu *et al.*, 2010; Allen *et al.*, 2011), spheres (Al-Raoush and Alsaleh, 2007), cylinders (Stoyan and Chugay, 2009), and some other special shapes such as tablet-shaped particles (Song *et al.*, 2006). Bortfeldt and Wäscher (2013) stated that only 1.8% of the published works in the packing problem area are related to irregular pieces.

Since the earliest work of Art (1966), the no-fit polygon (NFP) has been the most powerful geometric tool used by most researchers worldwide to solve the 2D irregular packing problem. It has become so

‡ Corresponding author

* Project supported by the Natural Science Foundation of Guangdong Province, China (No. S2013040016594), the Natural Science Foundation of Liaoning Province, China (No. 201102164), and the Fundamental Research Funds for the Central Universities, China (No. 2013ZM0124)

ORCID: Xiao LIU, http://orcid.org/0000-0003-2975-4749

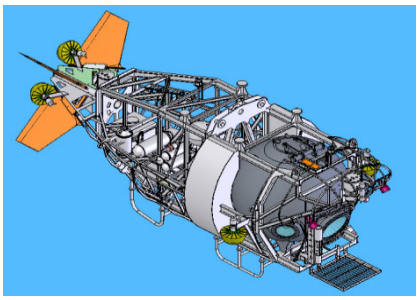

Fig. 1 A human occupied vehicle (HOV)

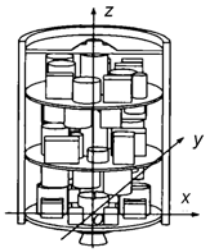

Fig. 2 A satellite module (Huo *et al.*, 2006)

prevailing that hundreds of methods have been created to generate NFPs (Bennell *et al.*, 2001; Liu and He, 2006; Burke *et al.*, 2007).

Although numerous methods have been proposed, NFP calculation is still time-consuming. Taking the benchmark problem SWIM as an example, Burke *et al.* (2007) declared that it needs 1/66 s to generate one NFP. The total execution time is only $1/66\times(10\times 2)^2$=6.06 s (10 pieces, each with two rotation angles), which appears to be a short time under this condition. However, if the number of pieces is increased to 50 and each piece is allowed to have four orientations, then the execution time will increase to $1/66\times(50\times4)^2$ =606.06 s. As for the 3D packing problem, the long-time calculation for NFPs is much more intolerable: $1/66\times(50\times4\times3)^2$=5454.55 s (approximately 1.5 h). To make matters worse, even if people do not mind the long time waiting for computing the NFP, no researcher has ever successfully provided a promising method to generate the NFP for the 3D packing problem.

The NFP generating barrier forces researchers to consider other flexible ways. Stoyan and Chugay (2009) modeled a polyhedron by a union of axis-aligned bounding boxes (AABBs, Fig. 3), whose overlap test is fast. However, if the polyhedron is rotated, then the AABB will be transformed into an oriented bounding box (OBB), whose overlap test is much more complicated than that of AABB (Ericson, 2004). This means Stoyan *et al.* (2004)'s method will be slowed down if the polyhedron is allowed to rotate.

Zhang *et al.* (2008) used spheres to represent polyhedrons (Fig. 4). Their method is similar to that of Stoyan *et al.* (2004), but is more convenient for collision detection because such a process can be easily achieved among spheres. Furthermore, a polyhedron does not need to be remodeled into spheres after being rotated.

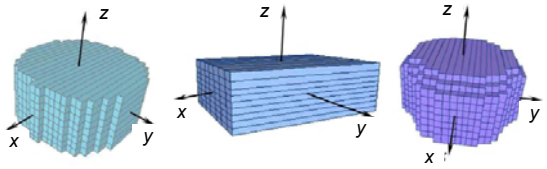

Fig. 3 Approximate representation of objects (reprinted from Stoyan and Chugay (2009), Copyright 2009, with permission from Elsevier)

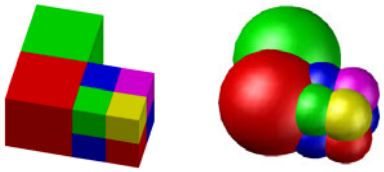

Fig. 4 Approximation representation of spheres, modified from Zhang *et al.* (2008)

From the previous discussions, it can be observed that both Stoyan *et al.* (2004) and Zhang *et al.* (2008) required disassembling a polyhedron into many smaller components. Hence, a trade-off between accuracy and speed raises a dilemma for researchers.

In the current study, we propose a new constructive algorithm, called HAPE3D, which is a heuristic algorithm based on the principle of minimum total potential energy for the 3D packing problem. It requires neither the NFP nor disassembling a polyhedron. HAPE3D can deal with irregularly shaped pieces with holes and cavities. Furthermore, it allows a polyhedron to have multiple orientations.

Although HAPE3D itself has exhibited good performance during the following experiments, it should be hybridized with a heuristic to search for a better solution (Allen *et al.*, 2011). Various methods based on simulated annealing (SA) (Szykman and Cagan, 1995; Cagan *et al.*, 1998) have been considered for solving the 3D packing problem. In the

current study, we also present a hybrid algorithm called HAPE3D+SA.

3 A new constructive algorithm: HAPE3D

Liu and Ye (2011) proposed a new constructive algorithm based on the principle of minimum total potential energy (HAPE) for the irregular 2D packing problem. HAPE does not need to calculate the NFP and is capable of dealing with an irregularly shaped polygon which is allowed to rotate. The algorithm presented in this study is based on the work of Liu and Ye (2011). Therefore, the algorithm is called HAPE3D. Before presenting the implementation procedures, several concepts are introduced in the following sections.

3.1 Principle of minimum total potential energy

The principle of minimum total potential energy is a fundamental concept which asserts that a structure or body will be deformed or displaced to a position that minimizes 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. It is expressed as follows:

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

Given that pieces are rigid in packing problems, the elastic energy U is zero. Consequently, Eq. (1) should be rewritten as follows:

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

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

The tendency of all weights to lower their positions is a basic law of nature, which also applies to packing problems. A piece always attempts to attain an optimal attitude to keep its center of gravity as low as possible.

As shown in Fig. 5a, four rectangles are piled up like an upside-down pyramid, which is evidently unstable, and likely to collapse under a slight disturbance (Fig. 5b). If the disturbance, caused by shaking of the container or a blow of the wind, becomes more 'intense', it will drive the rectangles to move around and orient to a new direction. Finally, the rectangles will 'lay out' in an attitude which is more stable (Fig. 5c). Note that both the location and orientation of the rectangle could be referred to as 'attitude'. There are infinite attitudes available, but the piece always prefers to choose the one with the lowest gravity center.

The center height of the combination of the four rectangles can be computed as follows:

$$h=\sum_i A_i h_i \Big/ \sum_i A_i, \tag{3}$$

where A_i and h_i are the area and center height of rectangle i, respectively.

Evidently, as h decreases, the combination will be increasingly stable and compact (Fig. 5c).

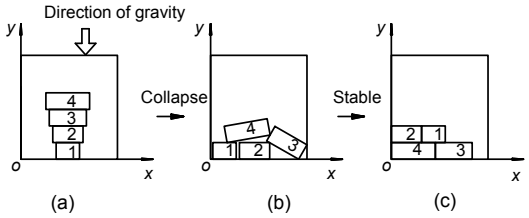

Fig. 5 Collapsing procedure of the upside-down pyramid
(a) Upside-down pyramid; (b) Stable attitude; (c) More stable attitude

3.2 Polyhedron separation test

Logically, the inverse of overlap is separation. Therefore, a separation test is proposed, which consists of two subsets as follows:

1. All vertices of polyhedron A are exterior points of polyhedron B, and vice versa.

2. All line segments of A do not cross any face of B, and vice versa.

Two aspects must be noted as follows: (1) When the vertices of A are outside B, A may contain B completely (Fig. 6a). (2) If only the first subset is satisfied, then polyhedrons A and B may still be overlapping (Fig. 6b).

3.3 Point-in-polyhedron test (PIPT)

Testing whether a point is inside a polyhedron is to test how many times a ray line, starting from this point and going any direction, intersects the face of

the polyhedron. If the intersection number is even, then the point is outside the polyhedron. Otherwise, the point is inside the polyhedron. As shown in Fig. 7, the polyhedron is a combination of two tetrahedrons, namely, $v_0v_1v_2v_3$ and $v_1v_2v_3v_4$. A ray, starting from point P_{test}, intersects face $v_0v_3v_2$ at point P_1 and face $v_2v_4v_3$ at point P_2. Because the intersection number is two, an even number, P_{test} can be regarded as an outer point of the polyhedron. Cui *et al.* (2011) introduced an algorithm for PIPT, named threshold-based ray-crossing (TBRC), which is efficient and robust compared with other classical algorithms. Hence, TBRC is used as the PIPT algorithm in HAPE3D.

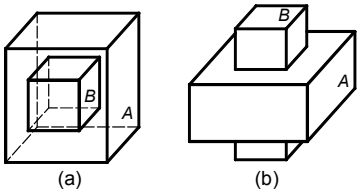

Fig. 6 Overlapped polyhedrons satisfying only one subset of separation test: (a) *A* contains *B*; (b) *A* and *B* are overlapping

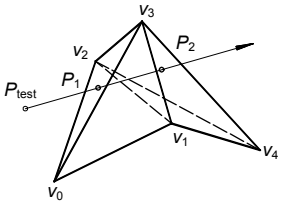

Fig. 7 Point-in-polyhedron test (PIPT)

3.4 Advance-or-retreat method for polyhedron contact

As shown in Fig. 8a, it is supposed that polyhedron *A* is fixed, whereas *B* slides to the left to come in contact with *A* (Fig. 8b). We propose an advance-or-retreat method for polyhedron contact (Fig. 9).

3.5 Implementation procedure of HAPE3D

HAPE3D can be described as follows:

1. All pieces are sorted in order of decreasing volume and allocated individually into the container in this order.

2. Equally spaced points are set in the container. The vertical and horizontal distances between these points are referred to as the packing point distance (PPD).

3. The 'current' piece ready to be allocated into the container is moved to visit the packing points one by one.

4. The center height (*z* coordinate) of the current piece at each point is calculated. The optimal point with the smallest *z*, is determined and the current piece is accordingly allocated at the optimal point.

5. Because the packing points are set in the container 'discretely', many gaps will appear between the current piece and allocated pieces. These gaps can be eliminated by the advance-or-retreat method (Fig. 9). After vertical and horizontal sliding, the current piece will come in contact with other allocated ones or with the inner face of the container.

6. HAPE3D stops when all pieces are allocated.

The formal process of HAPE3D can also be stated by using the following pseudo code:

Input:
Point[0…PPN−1]; // Set PPN (packing point number)
// packing points in the container
Piece[0…quantity−1]; // Pieces have been sorted in order
// of decreasing volume
Begin
for (int *i*=0; *i*<quantity; *i*++)

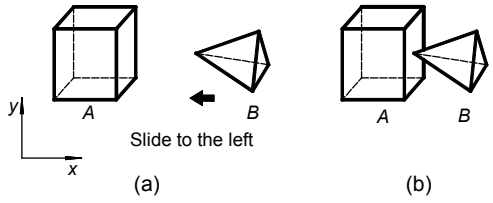

Fig. 8 Contact between pieces *A* and *B*
(a) Before contact; (b) Contacted

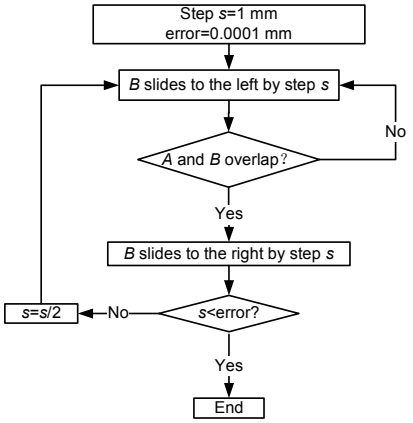

Fig. 9 Flowchart of the advance-or-retreat method for polyhedron contact

```
  {
    PackOnePiece(Piece[i]);
  }
End
```

The flowchart of packing a single piece is presented in Fig. 10.

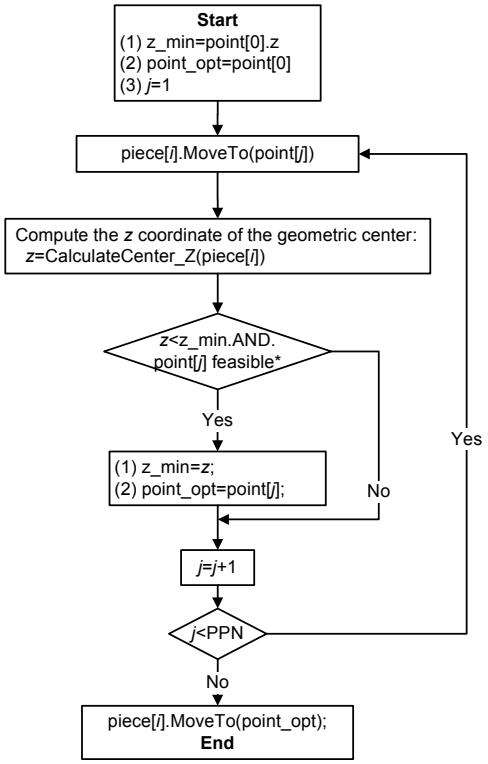

Fig. 10 Flowchart of packing one piece

'feasible*' indicates that the piece does not overlap with others at point[*j*]

The flowchart in Fig. 10 has been simplified for ease of understanding. In reality, necessary modifications have to be added to increase execution speed. For example, after the 'current' piece has been allocated in the container, it will occupy some packing points which are useless for the following pieces. Therefore, these points should be removed before allocating the next piece.

3.6 Hybridizing HAPE3D with simulated annealing

If the packing problem is represented by a permutation that is interpreted as the order in which the pieces are packed, then the constructive packing algorithm HAPE3D can be used to decode and evaluate the quality of the permutation in terms of packing density. Often, the predefined ordering, e.g., the order of decreasing volume, can produce a good packing layout. However, usually it is not good enough. Hence, a heuristic is required to search for another packing order which may give a better result. SA is considered as an excellent heuristic which has been successfully applied to solve the 3D packing problem (Szykman and Cagan, 1995; Cagan *et al.*, 1998). Consequently, we also choose SA to be hybridized with HAPE3D.

SA can be considered as a variant of the hill-climbing (HC) method. The most outstanding improvement of SA is that it can avoid being trapped in local minima. Instead of accepting only neighboring solutions that result in an improvement, worse solutions may also be accepted randomly with a certain probability. This probability depends on the temperature in physical annealing. When the temperature is high, the probability that the worse solutions are accepted is also high. During the annealing process, temperature gradually decreases based on the cooling schedule. The finding indicates that the algorithm becomes increasingly selective in accepting new solutions. By the end of the process, only moves that result in an improvement are accepted. Hence, compared with HC, SA behaves similarly when it reaches the lower bound of temperature.

We apply operator *i*Opt (i=1, 2, …, N, where N is the number of pieces) throughout the searching process. 1Opt randomly chooses two pieces and swaps their positions in the order; i.e., 1Opt means one swapping operation is applied to the order. This operator is extended to *N*Opt, where N swapping operations are performed and are likely to produce a radically different solution, which diversifies the search. Each operator has a different chance for selection—from 1Opt, which has the highest chance of being selected, to *N*Opt, which has a low chance of being selected. Few radical operators allow us to focus our search. Highly radical operators, e.g., *N*Opt, enable us to bypass local optima.

The pseudo codes for HAPE3D+SA are listed as follows:

```
Input:
  Pieces, RN, PPD, Container size, MaxIterationNum, T0, T
Begin
  Current.Ordering=SortOrderingbyDecreasingVolume( );
  Current.PackingDensity=HAPE3D(Current.Ordering);
  Best=Current; // Best Solution=Current Solution
  k=0;   // k: iteration number
  T0=0.01; // Initial temperature
```

```
while (k<MaxIterationNum)
{
// Calculate current temperature
   T=(MaxIterationNum−k)*T0/MaxIterationNum;
   // randomly generate an integer from 1 to N
   Opt=SelectOperator();
   Neighbor.Ordering=GenerateNeighbor(Current.
      Ordering, Opt);
   Neighbor.PackingDensity=HAPE3D(Neighbor.
      Ordering);
   if (Neighbor.PackingDensity>Current.PackingDensity)
      {Current=Neighbor;}
   else
   {
      // Calculate accepting probability
      P=exp(−(Current.PackingDensity−Neighbor.
         PackingDensity)/T);
      // if P is greater than the random number produced
      // by computer
      if (P>random())
         {Current=Neighbor;}
   }
   if (Neighbor.PackingDensity>Best.PackingDensity)
      {Best=Neighbor;}
   k=k+1;
}
return Best;
End
```

4 Computational experiments

To evaluate the performance of HAPE3D and the corresponding hybrid algorithm HAPE3D+SA, two experiments were conducted using a computer with a 2.0 GHz CPU. The data sets in the first experiment are listed in the appendix and those in the second experiment are obtained from Stoyan *et al.* (2004).

4.1 Experiment 1

Thirty-six polyhedrons with five types, as shown in Fig. 11 and Table 1, were to be allocated into a box-shaped container (20 mm×20 mm×60 mm). The PPD was set to 1 mm. Each polyhedron was allowed to have eight orientations (RN=8, each polyhedron is allowed to have eight rotation angles: 0°, 45°, …, 315°) around the x, y, and z coordinate axes at each packing point. The polyhedrons were sorted in order of decreasing volume before being allocated into the container.

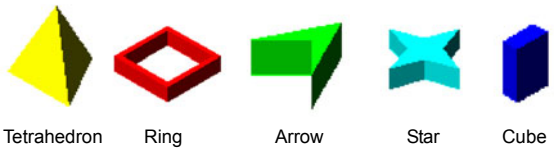

Fig. 11 Five types of polyhedrons

It can be seen from Fig. 12a that HAPE3D is capable of dealing with all kinds of pieces with irregular shapes including hole-filling and concave. To achieve a better packing result, SA was used as a search mechanism to derive new and improved input orderings for piece placement. The entire searching procedure was run for totally 500 iterations; i.e., 500 HAPE3Ds were executed during the entire searching procedure. The resulting packing height h obtained by HAPE3D+SA (Fig. 12b) was 31.2 mm, which was 20.0% lower than that of HAPE3D (Fig. 12a).

Table 1 Piece quantity and volume

Piece name	Quantity	Volume (mm^3)
Tetrahedron	8	144
Ring	4	132
Arrow	8	125
Star	8	108
Cube	8	48

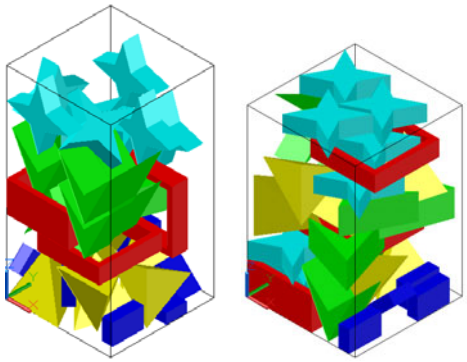

Fig. 12 Layout generated using HAPE3D (RN=8)
(a) HAPE3D (h=39.0 mm, t=16.1 s); (b) HAPE3D+SA (500 iterations, h=31.2 mm, t=9637.5 s)

4.2 Experiment 2

Stoyan *et al.* (2004) conducted four experiments using a PC with 900 MHz CPU which allocates 10 types of concave polyhedrons into a box-shaped

container with a rectangular bottom (35 mm×30 mm). In the first experiment 20 polyhedrons were packed, with two same polyhedrons of each type. The next three experiments were similar to the first experiment, but had 3, 4, and 5 copies for each type respectively. The packing height and execution time for these four experiments are listed in Table 2.

To evaluate the performance of HAPE3D, we conducted the same four experiments (Table 2 and Fig. 13) as in Stoyan *et al.* (2004). Table 2 shows that if a rotation is forbidden, h obtained by HAPE3D is similar to that of Stoyan *et al.* (2004). However, HAPE3D is much more time-consuming than the method proposed by Stoyan *et al.* (2004). The significant difference largely results from the fact that Stoyan *et al*. (2004) disassembled the polyhedron into many cuboids (AABB), whose overlap test algorithm was simple and efficient. However, if the polyhedron was rotated, the edge of the cuboids would not be parallel with the coordinate axis anymore. That means AABB will be transformed into OBB after rotation. As has been stated in Section 2, overlap test of OBB is much more complicated than that of AABB. That is to say, if rotation is allowed, the execution speed of Stoyan *et al.* (2004)'s method will also be slowed down.

Table 2 Comparison of HAPE3D and the method of Stoyan *et al.* (2004)

n	Packing height (mm)		Calculation time (s)	
	HAPE3D	Stoyan *et al.* (2004)	HAPE3D	Stoyan *et al.* (2004)
20	44.7	43.7	10.0	0.6
30	62.0	59.0	21.1	1.2
40	79.9	81.4	24.5	2.9
50	92.0	94.6	42.8	4.3

Note: n is the number of polyhedrons

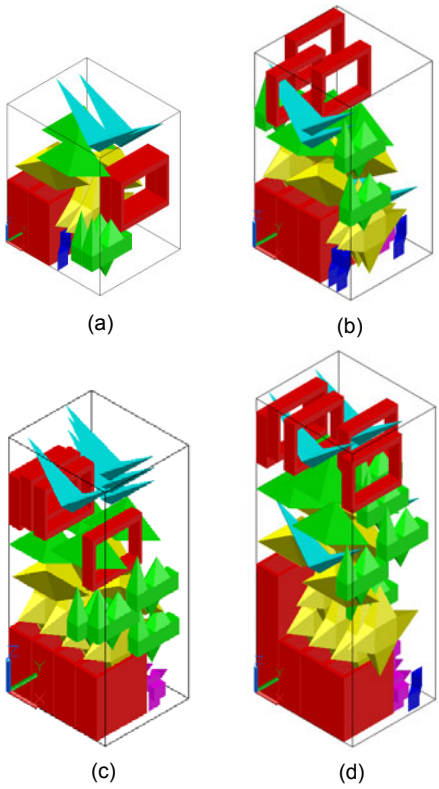

Fig. 13 Layout generated using HAPE3D when rotation is forbidden (RN=1)
(a) n=20, h=44.7 mm; (b) n=30, h=62 mm; (c) n=40, h=79.9 mm; (d) n=50, h=92.0 mm

From the previous discussion, it can be deduced that the execution time will be significantly increased under two conditions: (1) Polyhedron is allowed to be rotated; (2) A heuristic, e.g., SA, is used as a search mechanism. To further examine the performance of HAPE3D under these conditions, more experiments were conducted. HAPE3D was hybridized with SA (500 iterations) and all polyhedrons were allowed to rotate around the coordinate axis at RN different angles. The corresponding results are listed in Table 3 and Fig. 14. Due to space limitations, detailed layout results of only RN=4 are shown in Fig. 15.

As shown in Table 3 and Fig. 14, the packing height goes down gradually with the increase of RN, whereas the execution time rises up rapidly. Another conclusion can be drawn, by comparing Figs. 13 and 15, that HAPE3D can be well hybridized with SA, and further decrease the packing height. Taking n=50 as an example, h decreases from 92.0 mm (Fig. 13d) to 73.6 mm (Fig. 15d), which means the packing efficiency is significantly improved by 25%.

Table 3 Packing height and execution time with different RNs (HAPE3D+SA)

n	Packing height (mm)			Calculation time (s)		
	RN=1	RN=2	RN=4	RN=1	RN=2	RN=4
20	36.9	34.0	31.0	5921.5	14100.1	26202.1
30	55.6	48.0	46.0	10398.6	27745.6	53741.5
40	71.8	65.0	59.0	18383.9	48950.9	99952.0
50	92.0	80.1	73.6	23113.1	64463.0	125210.6

Note: n is the number of polyhedrons and RN=4 means each polyhedron is allowed to rotate around each coordinate axis at four angles (0°, 90°, 180°, and 270°)

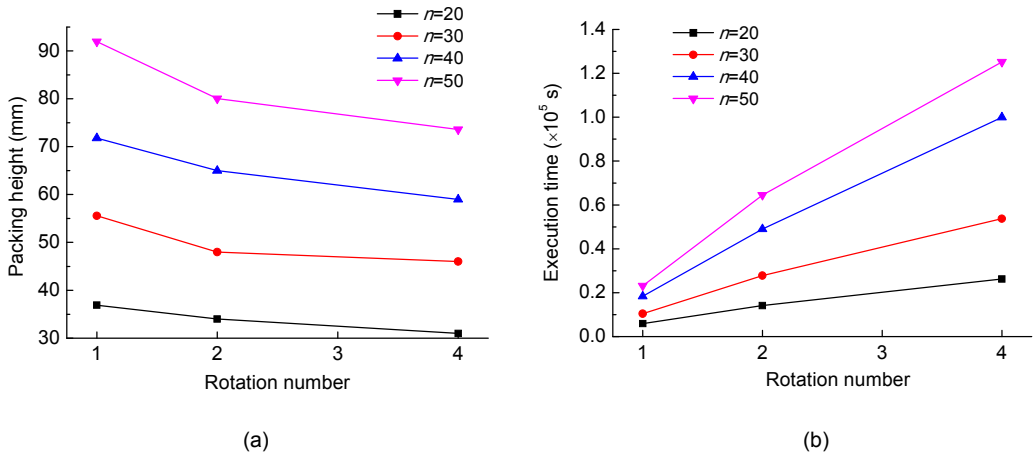

Fig. 14 Packing height (a) and execution time (b) with different rotation numbers

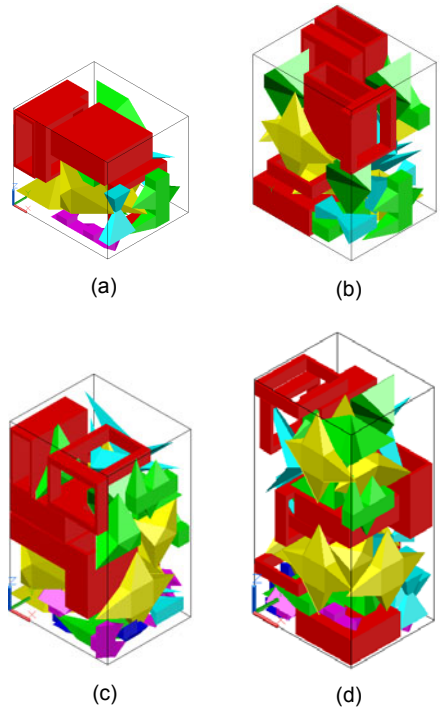

Fig. 15 Layout by HAPE3D+SA (500 iterations, RN=4)
(a) n=20, h=31.0 mm; (b) n=30, h=46.0 mm; (c) n=40, h=59.0 mm; (d) n=50, h=73.6 mm

5 Conclusions

From what has been discussed above, we can draw the conclusion that HAPE3D is a very promising constructive algorithm for the 3D packing problem. The most obvious distinction from the traditional packing algorithm is that HAPE3D does not need to calculate NFP anymore. Another merit for HAPE3D is that the 3D piece can retain its original shape. People do not need extra execution time on splitting an irregularly shaped piece into numerous tiny regular objects. The third advantage is that HAPE3D does not forbid the rotation freedom of the piece. Lastly, HAPE3D can be hybridized very well with other meta-heuristic algorithms to further improve the packing efficiency.

However, there is still much room for improvement. For example, the speed of HAPE3D does not fully satisfy the real industrial requirement. Because the smaller the PPD is, the more computing time it will cost. Actually, it is not suitable to assign only one PPD for different pieces as we do in this study. It is more reasonable to adjust PPD size dynamically according to the size of the piece.

Besides PPD, RN is another important impact factor on the performance of HAPE3D. Although larger RN tends to produce higher packing efficiency in this study, it is not suggested to assign a large value for RN without limitation, because the execution time is proportional to RN. Besides, Liu and Ye (2011) indicated that a larger RN does not always generate a better result. How to choose a reasonable value for PPD or RN could be a meaningful question for future research.

Acknowledgements

The authors would like to thank Guangzhou Wenchong Shipyard Co., Ltd. and Guangzhou Shipyard International Co., Ltd. for supporting this research.

References

Allen, S.D., Burke, E.K., Kendall, G., 2011. A hybrid placement strategy for the three-dimensional strip packing problem. *Eur. J. Oper. Res.*, **209**(3):219-227. [doi:10.1016/j.ejor.2010.09.023]

Al-Raoush, R., Alsaleh, M., 2007. Simulation of random packing of polydisperse particles. *Powder Technol.*, **176**(1):47-55. [doi:10.1016/j.powtec.2007.02.007]

Art, J.R.C., 1966. An Approach to the Two Dimensional Irregular Cutting Stock Problem. IBM Cambridge Scientific Center Report, Massachusetts.

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

Bortfeldt, A., Wäscher, G., 2013. Constraints in container loading—a state-of-the-art review. *Eur. J. Oper. Res.*, **229**(1):1-20. [doi:10.1016/j.ejor.2012.12.006]

Burke, E.K., Hellier, R.S.R., Kendall, G., *et al*., 2007. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *Eur. J. Oper. Res.*, **179**(1):27-49. [doi:10.1016/j.ejor.2006.03.011]

Cagan, J., Degentesh, D., Yin, S., 1998. A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout. *Comput.-Aid. Des.*, **30**(10):781-790. [doi:10.1016/S0010-4485(98)00036-0]

Cui, S., Zhang, S., Chen, X., *et al.*, 2011. Point-in-polyhedra test with direct handling of degeneracies. *Geo-spatial Inform. Sci.*, **14**(2):91-97. [doi:10.1007/s11806-011-0453-8]

Ericson, C., 2004. Real-Time Collision Detection. The Morgan Kaufmann Series in Interactive 3-D Technology. CRC Press, USA.

Huo, J., Li, G., Teng, H., 2006. Layout optimization of a satellite module using a parallel genetic-Powell-ant colony hybrid algorithm. *J. Dalian Univ. Technol.*, **46**(5):679-684.

Liu, H., He, Y., 2006. Algorithm for 2D irregular-shaped nesting problem based on the NFP algorithm and lowest-gravity-center principle. *J. Zhejiang Univ.-Sci. A*, **7**(4):570-576. [doi:10.1631/jzus.2006.A0570]

Liu, X., Ye, J., 2011. Heuristic algorithm based on the principle of minimum total potential energy (HAPE): a new algorithm for nesting problems. *J. Zhejiang Univ.-Sci. A (Appl. Phys. & Eng.)*, **12**(11):860-872. [doi:10.1631/jzus.A1100038]

Song, Y., Turton, R., Kayihan, F., 2006. Contact detection algorithms for DEM simulations of tablet-shaped particles. *Powder Technol.*, **161**(1):32-40. [doi:10.1016/j.powtec.2005.07.004]

Stoyan, Y., Chugay, A., 2009. Packing cylinders and rectangular parallelepipeds with distances between them into a given region. *Eur. J. Oper. Res.*, **197**(2):446-455. [doi:10.1016/j.ejor.2008.07.003]

Stoyan, Y.G., Gil, N.I., Pankratov, A., *et al.*, 2004. Packing Non-convex Polytopes into a Parallelepiped. Technische Universität Dresden, Dresden.

Szykman, S., Cagan, J., 1995. A simulated annealing-based approach to three-dimensional component packing. *J. Mech. Des.*, **117**(2A):308-314. [doi:10.1115/1.2826140]

Wu, Y., Li, W., Goh, M., *et al.*, 2010. Three-dimensional bin packing problem with variable bin height. *Eur. J. Oper. Res.*, **202**(2):347-355. [doi:10.1016/j.ejor.2009.05.040]

Zhang, W., Gao, Y., Fang, L., *et al.*, 2008. Three-dimensional component layout modeling and optimization design. *Acta Aeronaut. Astronaut. Sin.*, **29**(6):1554-1562 (in Chinese).

Appendix: Data set of Experiment 1

Data set of Tetrahedron

Name
Tetrahedron
Volume
144.3376
Mass center
5, 2.8868, 2.5
Reference/origin point
0, 0, 0

VertexList
0: 0, 0, 0 2: 5, 8.6603, 0
1: 10, 0, 0 3: 5, 2.8868, 10
EndVertexList

EdgeList
0: 0, 1 3: 0, 3
1: 1, 2 4: 1, 3
2: 2, 0 5: 2, 3
EndEdgeList

FaceList
0(3): 0, 2, 1 2(3): 1, 2, 3
1(3): 0, 1, 3 3(3): 3, 2, 0
EndFaceList

Data set of Ring

Name
Ring
Volume
132
Mass center
0, 0, 1.5
Reference/origin point
0, 0, 1.5

VertexList
0: −6, −6, 0
1: 6, −6, 0
2: 6, 6, 0
3: −6, 6, 0
4: −5, −5, 0
5: 5, −5, 0
6: 5, 5, 0
7: −5, 5, 0
8: −6, −6, 3
9: 6, −6, 3
10: 6, 6, 3
11: −6, 6, 3
12: −5, −5, 3
13: 5, −5, 3
14: 5, 5, 3
15: −5, 5, 3
EndVertexList

EdgeList
0: 0, 1
1: 1, 2
2: 2, 3
3: 3, 0
4: 4, 5
5: 5, 6
6: 6, 7
7: 7, 4
8: 8, 9
9: 9, 10
10: 10, 11
11: 11, 8
12: 12, 13
13: 13, 14
14: 14, 15
15: 15, 12
16: 4, 12
17: 5, 13
18: 6, 14
19: 7, 15
20: 0, 8
21: 1, 9
22: 2, 10
23: 3, 11
EndEdgeList

FaceList
0(4): 0, 4, 5, 1
1(4): 5, 6, 2, 1
2(4): 3, 2, 6, 7
3(4): 3, 7, 4, 0
4(4): 8, 9, 13, 12
5(4): 9, 10, 14, 13
6(4): 15, 14, 10, 11
7(4): 8, 12, 15, 11
8(4): 12, 13, 5, 4
9(4): 13, 14, 6, 5
10(4): 14, 15, 7, 6
11(4): 15, 12, 4, 7
12(4): 9, 8, 0, 1
13(4): 10, 9, 1, 2
14(4): 11, 10, 2, 3
15(4): 8, 11, 3, 0
EndFaceList

Data set of Arrow

Name
Arrow
Volume
125
Mass center
5, 5, 2.5
Reference/origin point
5, 5, 2.5

VertexList
0: 0, 0, 0
1: 5, 5, 0
2: 10, 0, 0
3: 5, 10, 0
4: 0, 0, 5
5: 5, 5, 5
6: 10, 0, 5
7: 5, 10, 5
EndVertexList

EdgeList
0: 0, 1
1: 1, 2
2: 2, 3
3: 3, 0
4: 4, 5
5: 5, 6
6: 6, 7
7: 7, 4
8: 0, 4
9: 1, 5
10: 2, 6
11: 3, 7
EndEdgeList

FaceList
0(4): 0, 3, 2, 1
1(4): 4, 5, 6, 7
2(4): 5, 4, 0, 1
3(4): 6, 5, 1, 2
4(4): 7, 6, 2, 3
5(4): 4, 7, 3, 0
EndFaceList

Data set of Star

Name
Star
Volume
108
Mass center
0, 0, 1.5
Reference/origin point
0, 0, 1.5

VertexList
0: 0, −6, 0
1: 1.5, −1.5, 0
2: 6, 0, 0
3: 1.5, 1.5, 0
4: 0, 6, 0
5: −1.5, 1.5, 0
6: −6, 0, 0
7: −1.5, −1.5, 0
8: 0, −6, 3
9: 1.5, −1.5, 3
10: 6, 0, 3
11: 1.5, 1.5, 3
12: 0, 6, 3
13: −1.5, 1.5, 3
14: −6, 0, 3
15: −1.5, −1.5, 3
EndVertexList

EdgeList
0: 0, 1
1: 1, 2
12: 12, 13
13: 13, 14

2: 2, 3
3: 3, 4
4: 4, 5
5: 5, 6
6: 6, 7
7: 7, 0
8: 8, 9
9: 9, 10
10: 10, 11
11: 11, 12
14: 14, 15
15: 15, 8
16: 0, 8
17: 1, 9
18: 2, 10
19: 3, 11
20: 4, 12
21: 5, 13
22: 6, 14
23: 7, 15
EndEdgeList

FaceList
0(3): 0, 7, 1
1(3): 3, 2, 1
2(3): 5, 4, 3
3(3): 7, 6, 5
4(4): 7, 5, 3, 1
5(3): 8, 9, 15
6(3): 9, 10, 11
7(3): 11, 12, 13
8(3): 13, 14, 15
9(4): 9, 11, 13, 15
10(4): 0, 1, 9, 8
11(4): 1, 2, 10, 9
12(4): 2, 3, 11, 10
13(4): 3, 4, 12, 11
14(4): 4, 5, 13, 12
15(4): 5, 6, 14, 13
16(4): 6, 7, 15, 14
17(4): 0, 7, 15, 8
EndFaceList

Data set of Cube

Name
Cube
Volume
48
Mass center
1, 2, 3
Reference/origin point
1, 2, 3

VertexList
0: 0, 0, 0
1: 2, 0, 0
2: 2, 4, 0
3: 0, 4, 0
4: 0, 0, 6
5: 2, 0, 6
6: 2, 4, 6
7: 0, 4, 6
EndVertexList

EdgeList
0: 0, 1
1: 1, 2
2: 2, 3
3: 3, 0
4: 4, 5
5: 5, 6
6: 6, 7
7: 7, 4
8: 0, 4
9: 1, 5
10: 2, 6
11: 3, 7
EndEdgeList

FaceList
0(4): 0, 3, 2, 1
1(4): 4, 5, 6, 7
2(4): 0, 1, 5, 4
3(4): 1, 2, 6, 5
4(4): 3, 7, 6, 2
5(4): 0, 4, 7, 3
EndFaceList