Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

A heuristic method for solving triangle packing problem

CHEN Chuan-bo (陈传波), HE Da-hua (何大华)

(*College of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074, China*)

E-mail: chuanboc@163.com; hedahua@xinhuanet.com

Received June 11, 2004; revision accepted Oct. 20, 2004

Abstract: Given a set of triangles and a rectangle container, the triangle packing problem is to determine if these triangles can be placed into the container without overlapping. Triangle packing problem is a special case of polygon packing problem and also NP-hard, so it is unlikely that an efficient and exact algorithm can be developed to solve this problem. In this paper, a new concept of rigid placement is proposed, based on which a discrete solution space called rigid solution space is constructed. Each solution in the rigid solution space can be built by continuously applying legal rigid placements one by one until all the triangles are placed into the rectangle container without overlapping. The proposed Least-Destruction-First (LDF) strategy determines which rigid placement has the privilege to go into the rectangle container. Based on this, a heuristic algorithm is proposed to solve the problem. Combining Least-Destruction-First strategy with backtracking, the corresponding backtracking algorithm is proposed. Computational results show that our proposed algorithms are efficient and robust. With slight modification, these techniques can be conveniently used for solving polygon packing problem.

Key words: Triangle packing problem, Rigid placement, Flexibility, Destruction, Least-Destruction-First (LDF) strategy, Backtracking

doi:10.1631/jzus.2005.A0565 **Document code:** A **CLC number:** TP301.6

INTRODUCTION

Most packing problems (Dowsland and Dowsland, 1992) are NP-hard (Garey and Johnson, 1979); among which are bin-packing, floorplan, rectangle packing, packing a set of circles into a large circle or square, non-rectangular packing problems and so on (Li and Milenkovic, 1995; Liang *et al*., 2002; Lipnitskii, 2002; Milenkovic and Daniels, 1996; Milenkovic *et al*., 1991; Osogami and Okano, 2003; Wang, 2002). Some of these such as bin-packing problem and rectangle packing problem have been well studied, some, such as the non-rectangular packing problem, are being seriously considered. For non-rectangular packing problems, e.g. polygon packing problem, bounds and worst-case analysis and average case analysis of the algorithms are more difficult than that of rectangular packing problems.

For polygon packing problem, researchers tend to apply some constraints to the polygons, for example, no rotation is allowed (Li and Milenkovic, 1995; Milenkovic and Daniels, 1996; Milenkovic *et al*., 1991), then the textile layout problem is transformed to translational polygon containment problem which is NP-complete. This is useful in practice because material can sometimes be placed only in a small number of possible orientations so that the fabric pattern lines up correctly. But this is not the case all the time because these constraints may cause the optimal solution to be missed. In order to get better solutions, we should give the polygons more freedom; both translation and rotation should be considered.

Given a set of triangles and a rectangle container, the triangle packing problem is to determine if these triangles can be placed into the container without overlapping. Triangle packing problem is a special case of polygon packing problem (Li and Milenkovic, 1995; Milenkovic and Daniels, 1996) and also NP-hard, so it is unlikely that an efficient and exact algorithm can be developed to solve this problem. Furthermore, a triangle can continuously translate and rotate in a plane, this means the placements of a tri-

angle may be infinite and this makes the solution space of the problem is also infinite. As a consequence, it is necessary to apply some constraints to the triangles so that we can search a solution in a finite solution space. On the other hand, too many constraints may also adversely affect the quality of the solutions. To establish a trade-off between these, a new concept, rigid placement, is proposed. In this paper, a triangle has only a finite number of positions and orientations under this constraint. Based on this, the rigid solution space is constructed, each solution in which can be built by continuously applying legal rigid placements one by one until all the triangles are placed into the container. Thus the continuous problem is transformed into a combinatorial problem. To choose a better legal rigid placement under a partial solution, the Least-Destruction-First (LDF) strategy is proposed. Based on it, a heuristic algorithm and the corresponding backtracking algorithm are proposed for solving the problem; the computational results show that our proposed algorithms are efficient and robust.

The rest of the paper is arranged as follows: a detailed description of rigid placement and rigid solution space is given in Section 2, the LDF strategy is proposed in Section 3, and in Section 4, the basic algorithm and the backtracking algorithm are described and computational complexity are also analyzed. At last in Section 5, we present some computational results and concluding remarks.

RIGID PLACEMENT AND RIGID SOLUTION SPACE

As we know, a triangle M in a plane has three degrees of freedom: one rotation and two translations. If triangle M interacts (just collide, no overlapping) with another triangle S, the contact configuration can only be one of the following three types or their combinations: (1) One vertex of triangle M and one vertex of triangle S have the same coordinates; (2) One side of triangle M and one side of triangle S lie in the same line; (3) One vertex of triangle M lies on one side of triangle S or one vertex of triangle S lies on one side of triangle M.

The above three cases are called vertex-vertex constraint, side-side constraint and vertex-side constraint respectively; the first two cases eliminate two degrees of freedom of triangle M and the third eliminates one. Obviously, if a triangle has one degree of freedom or more, it is not possible to determine its position and orientation uniquely, only when the three degrees of freedom of triangle M are eliminated can we determine its position and orientation definitely.

For convenience, we assume the rectangle container to be enclosed by four triangles, as shown in Fig.1. If k triangles out of the total n have been placed into the container without overlapping, we call the k+4 triangles including the four enclosing triangles and the k triangles in the container packed triangles, the n–k triangles outside of the container are called active triangles. Now we take an active triangle M and let it interact with those k+4 packed triangles, then every packed triangle will probably give constraints to triangle M and eliminate some degrees of freedom of triangle M. If the three degrees of freedom of triangle M are all eliminated, we call the position and orientation of triangle M a rigid placement. If a rigid placement of an active triangle is in the container and does not overlap with any of the packed triangles, then the rigid placement is called legal rigid placement.

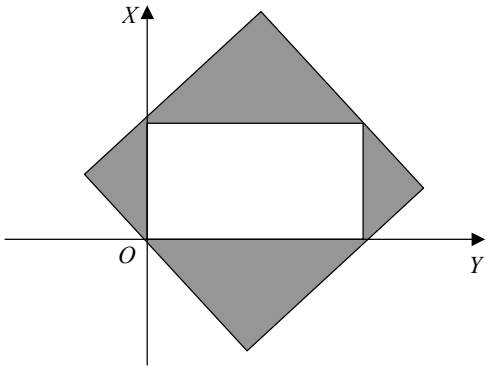
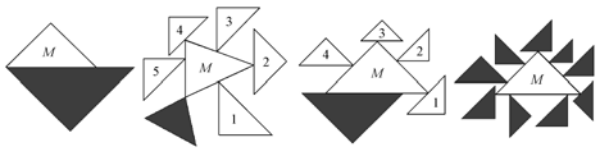


Fig.1 The enclosing triangles

Rigid placements include the following four types, as shown in Fig.2.

Here triangle M is an active triangle, the gray triangles and the numbered triangles are packed triangles. Fig.2a shows the type 0 rigid placement of triangle M; in Fig.2b, the gray triangle and one of the numbered triangle can determine a type 1 rigid placement of triangle M; in Fig.2c, the gray triangle and one of the numbered triangle can determine a type

2 rigid placement of triangle *M*; and in Fig.2d, any three gray triangles of the nine can determine a type 3 rigid placement of triangle *M*.



(a) (b) (c) (d)

Fig.2 Classification of rigid placements
(a) Type 0; (b) Type 1; (c) Type 2; (d) Type 3

Determining the legal rigid placements of an active triangle is trivial.

If a solution of the problem can be constructed by successfully repeatedly applying legal rigid placements *n* times, then it is called a rigid solution. The set of rigid solutions is called rigid solution space. As we will show in the following, the number of legal rigid placements under a partial solution is finite, then at each step, we have only a finite number of choices to place a triangle into the container. Since we can only repeat this *n* times at most, the rigid solution space is finite. Unfortunately, enumerating all the rigid solution will cost exponential time. In this paper, a heuristic method is used to solve the problem other than brute-force enumeration.

LEAST-DESTRUCTION-FIRST STRATEGY

We will place the triangles into the container one by one. At each step, there may be a number of legal rigid placements. Suppose *k* triangles have been placed in the container without overlapping, then there are *n*–*k* active triangles left outside of the container. The strategy to choose a legal rigid placement to place the triangles will greatly affect the efficiency of the algorithm. In this paper, the LDF strategy is proposed. As we will explain below, LDF strategy is not only intuitively reasonable but also efficient and robust in practice.

When *k* (*k*=0, 1, …, *n*–1) triangles have been placed in the container, we call the status of the solution a partial solution. According to Fig.2, one packed triangle may determine a type 0 rigid placement; two packed triangles may determine a type 1 or type 2 rigid placement; and three packed triangles may determine a type 3 rigid placement. So under a partial solution, the number of type 0~type 4 rigid placements of an active triangle *M* will be $O(k+4)$, $O(C_{k+4}^2)$, $O(C_{k+4}^2)$ and $O(C_{k+4}^3)$ and these can be simplified as $O(k)$, $O(k^2)$, $O(k^2)$ and $O(k^3)$, and the total number of rigid placements of triangle *M* is $O(k^3)$. Since the number of the legal rigid placements will be no more than that of rigid placements, and will also be $O(k^3)$. For an active triangle *M*, we define the number of its legal rigid placements under a partial solution as its flexibility. We also define the sum of the flexibilities of the *n*–*k* active triangle as flexibility of the partial solution, and these will be $O(k^3)$ and $(n-k)O(k^3)=a_1nk^3+a_2k^4$ respectively, where a_1 and a_2 are constants.

For a legal rigid placement *R* of an active triangle *M*, the destruction D_R of *R* is defined as follows:

$$D_R = V_M + \sum_{T\in ACTIVE \ \&\ T\neq M} O_T \qquad (1)$$

where V_M stands for the flexibility of triangle *M*; *ACTIVE* stands for the set of active triangles under the partial solution; O_T stands for the number of legal rigid placements of triangle *T* that overlap with *R*.

Under a partial solution, if we apply the legal rigid placement *R* of triangle *M*, then in the next partial solution in which only *n*–*k*–1 active triangles are left, every legal rigid placement of *M* becomes illegal because *M* is no longer an active triangle, and also every legal rigid placement that overlaps with *R* becomes illegal because it overlaps with *R* which has been a packed triangle. Intuitively, the flexibility of a partial solution reveals somewhat its capacity of holding triangles, so the legal rigid placement with the least destruction will be a good choice to place triangles.

We should do little harm to or even promote the flexibility of a partial solution in order to place more triangles into the container, so the legal rigid placement with the least destruction has the privilege to be placed; we call this strategy LDF strategy.

BASIC ALGORITHM

Based on rigid solution space and LDF strategy,

we propose a heuristic algorithm for solving triangle packing problem, it is described as follows:

(a) Establish a Cartesian coordinate system as shown in Fig.1, let k=0;
(b) Compute the legal rigid placements of the n–k active triangles and their destructions;
(c) If there are no legal rigid placements, go to (e), otherwise apply the legal rigid placement with the least destruction, let k=k+1;
(d) If k=n, then return SUCCESS, otherwise go to (b);
(e) Return FAILURE.

As we know, the flexibility of an active triangle is $O(k^3)$ and the flexibility of the partial solution is $a_1nk^3+a_2k^4$. Here we assume the time needed to compute a rigid placement, or to judge if a triangle is in the container, or to judge if two triangles in a plane overlap with each other, or to compare two integers are constants. Since there are $a_1nk^3+a_2k^4$ legal rigid placements at most under the partial solution, the time needed to compute all the legal rigid placements of the n–k active triangles will be $a_1nk^3+a_2k^4$. According to Eq.(1), we should check if two triangles in a plane overlap with each other almost $a_1nk^3+a_2k^4$ times to get the destruction of a legal rigid placement, so the time needed to compute the destructions of all the legal rigid placements will be

$$T_k' =(a_1nk^3+a_2k^4)^2=b_1n^2k^6+b_2nk^7+b_3k^8 \quad (2)$$

where b_1, b_2 and b_3 are constants, and T_k' constitutes the main part of the computational time of the algorithm. The time needed to judge if each rigid placement is in the container is $a_1nk^3+a_2k^4$; we should check if two triangles in a plane overlap with each other k times in order to determine if a rigid placement is legal because there are k packed triangles under the partial solution, so the time to determine all the legal rigid placements will be $k(a_1nk^3+a_2k^4)=a_1nk^4+a_2k^5$; the time needed to find the legal rigid placement with the least destruction is $a_1nk^3+a_2k^4$, and these can all be omitted with respect to T_k'. Let T_k denote the time that needed to process the partial solution, then

$$T_k = b_1n^2k^6+b_2nk^7+b_3k^8 \quad (3)$$

Obviously, in the worst-case the total running time T of the basic algorithm is

$$T = \sum_{k=0}^{n-1} T_k \sim O(n^9) \quad (4)$$

Hence the computational complexity of the basic algorithm is $O(n^9)$.

BACKTRACKING ALGORITHM

Combining the basic algorithm with backtracking, we get the corresponding backtracking algorithm. Note that the backtracking algorithm will always outperform the basic algorithm. A new concept should be defined before describing the algorithm.

Under a partial solution, the potential of a legal rigid placement R of an active triangle M is defined as follows: R is placed into the container to form a new partial solution which has k+1 packed triangles in the container, then the basic algorithm runs starting with the new partial solution until all the triangles are placed into the container or the basic algorithm returns FAILURE, the number of packed triangles at this time in the container is the potential of R.

According to the definition, the legal rigid placement with the largest potential will probably lead to placing more triangles into the container, and it has the privilege to be placed. Now we can describe the backtracking algorithm as follows:

(a) Establish a Cartesian coordinate system as shown in Fig.1, let k=0;
(b) Compute the legal rigid placements of the n–k active triangles and their potentials;
(c) If there are no legal rigid placements, go to (e), otherwise apply the legal rigid placement with the largest potential, let k=k+1;
(d) If k=n, return SUCCESS, otherwise go to (b);
(e) Return FAILURE.

In the worst-case, the total running time of the backtracking algorithm is $Q = \sum_{k=0}^{n-1} Q_k$, where Q_k denotes the time needed to process the partial solution. The time needed to compute the potentials of legal

rigid placements will comprise the main part of Q_k, and the time needed to judge if a rigid placement is in the container, or to judge if each rigid placement is legal, or to find the legal rigid placement with the largest potential are very similar to those of the basic algorithm and can all be omitted. Since the computing of the potential of a legal rigid placement is exactly the execution of the basic algorithm starting with a partial solution which has k+1 packed triangles in the container, the time needed to compute a potential is $\sum_{i=k+1}^{n-1} T_i$, and there are $a_1nk^3+a_2k^4$ legal rigid placements under a partial solution, hence

$$Q_k=(a_1nk^3+a_2k^4)\sum_{i=k+1}^{n-1} T_i \tag{5}$$

In the worst-case the total running time Q of the backtracking algorithm is

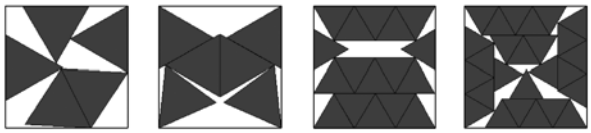
$$Q=\sum_{k=0}^{n-1} Q_k \sim O(n^{14}) \tag{6}$$

Hence the computational complexity of the backtracking algorithm is $O(n^{14})$.

COMPUTATIONAL RESULTS AND CONCLUDING REMARKS

We have implemented the program in Visual C++, since there are few benchmarks on triangle packing problem, the problem of packing unit equilateral triangles into the smallest square (http://www.stetson.edu/~efriedma/triinsqu) is used to test the algorithms, here we only provide the computational results of the backtracking algorithm because it always outperforms the basic algorithm. The computation environments are Pentium IV 2.4 GHz/512 M/80 G.

Since type 3 rigid placements are relatively complex to compute and obviously not a good way to make use of space, they are ignored in the algorithm. In Fig.3, n denotes the number of unit equilateral triangles; L denotes the side length of the smallest square that holds the n unit equilateral triangles; the corresponding computational time (including drawing the triangles and updating the views) for n=5, n=6, n=17 and n=26 are 1.76, 1.42, 4.96 and 1.7 s respectively. The side lengths of the smallest square obtained by the algorithm are very close to those of Friedman. Note that the topologies of the packing patterns for n=5 and n=6 are very different from those of Friedman.



(a) (b) (c) (d)

Fig.3 Packing unit equilateral triangles into a square
(a) n=5, L=1.8366; (b) n=6, L=1.937;
(c) n=17, L=3.017; (d) n=26, L=3.599

Furthermore, for randomly drawn rectangle container and triangles, our algorithms can also give encouraging packing patterns.

Though the computational complexity analysis shows that the complexity polynomial has a high index, our algorithms are efficient and robust in practice. For randomly given parameters, the average time needed by the basic algorithm for 20 triangles is almost 3 s if type 3 rigid placements are ignored.

As we can see from the computational results, our proposed algorithms can rapidly generate very compact packing patterns; this is due to the new concept of rigid placement and the LDF strategy. Rigid placements are computable and enumerable, and thus we transform the triangle packing problem from a continuous problem to a combinatorial problem. Experientially, rigid placement can be considered a very good way to nest an object so as to make good use of space, and the best solution in the rigid solution space should be very close to optimal. The LDF strategy is an efficient local optimization method because it takes into consideration all the information of the partial solution before taking an action. These tactics can also be easily generalized for solving polygon packing problem, but more detailed work should be undertaken.

Triangle packing problem is a decision problem and the basic and backtracking algorithms are both approximate; this means that they may report no solution in some instances that actually have solutions. But to what degree can our algorithms give right

answers to those instances that actually have solutions is very difficult to analyze, this is because the side lengths of the rectangle container and the triangles can be real numbers, and it is a challenge to give a set of instances with their parameters having reasonable statistical distribution. General benchmarks for this problem are not available, and statistical characterization of computational results of our proposed algorithms cannot be easily drawn.

A deficiency of the rigid solution space is that the optimal solution may not be included in it. In this case, our proposed algorithms are sure to fail. This is because optimal solutions of some instances should be constructed by simultaneous containment of several triangles while our proposed algorithms can only place the triangles one after one. For example, the packing patterns for n=5 and n=6 for the problem of packing unit equilateral triangles into the smallest square in Fig.3 are not as good as those of Friedman. To get the optimal solution, some triangles should be placed into the container simultaneously. Simultaneous containment is another challenge. More work should be undertaken to take a further step into the problem.

References

Dowsland, K.A., Dowsland, W.B., 1992. Packing problems. *European Journal of Operational Research*, **56**(1):2-14.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, New York.

Li, Z., Milenkovic, V.J., 1995. Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research*, **84**:539-561.

Liang, K.H., Yao, X., Newton, C., Hoffman, D., 2002. A new evolutionary approach to cutting stock problems with and without contiguity. *Computers and Operations Research*, **29**(12):1641-1659.

Lipnitskii, A.A., 2002. Use of genetic algorithms for solution of the rectangle packing problem. *Cybernetics and Systems Analysis*, **38**(6):943-946.

Milenkovic, V.J., Daniels, K.M., 1996. Translational Polygon Containment and Minimal Enclosure Using Mathematical Programming. Proceedings of the Annual ACM Symposium on Theory of Computing, p.109-118.

Milenkovic, V.J., Daniels, K.M., Li, Z., 1991. Automatic Marker Making. Proceedings of the Third Canadian Conference on Computational Geometry, p.243-246.

Osogami, T., Okano, H., 2003. Local search algorithms for the bin packing problem and their relationships to various construction heuristics. *Journal of Heuristics*, **9**(1):29-49.

Wang, H.Q., 2002. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, **141**(2):440-453.