Research Article

A Heuristic Algorithm for Solving Triangle Packing Problem

Ruimin Wang,1 Yuqiang Luo,2 Jianqiang Dong,3 Shuai Liu,4 and Xiaozhuo Qi5

1 School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China
2 Informatization Office, University of Shanghai for Science and Technology, Shanghai 200093, China
3 Zhengzhou Xinda Jian Information Technology Co., Ltd., Zhengzhou 450002, China
4 College of Science, University of Shanghai for Science and Technology, Shanghai 200093, China
5 95107 Troops, Guangzhou 510500, China

Correspondence should be addressed to Ruimin Wang; iermwang@zzu.edu.cn and Yuqiang Luo; abel.luo.me@gmail.com

Received 20 September 2013; Accepted 24 October 2013

Copyright © 2013 Ruimin Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The research on the triangle packing problem has important theoretic significance, which has broad application prospects in material processing, network resource optimization, and so forth. Generally speaking, the orientation of the triangle should be limited in advance, since the triangle packing problem is NP-hard and has continuous properties. For example, the polygon is not allowed to rotate; then, the approximate solution can be obtained by optimization method. This paper studies the triangle packing problem by a new kind of method. Such concepts as angle region, corner-occupying action, corner-occupying strategy, and edge-conjoining strategy are presented in this paper. In addition, an edge-conjoining and corner-occupying algorithm is designed, which is to obtain an approximate solution. It is demonstrated that the proposed algorithm is highly efficient, and by the time complexity analysis and the analogue experiment result is found.

1. Introduction

Solving NP-hard problems is always the bottleneck task for computer science and techniques. However, in recent years, many research results have indicated that there is no a complete, accurate, and fast solving algorithm for this kind of problem at all. People tend to focus on searching for a quick and practical approximation algorithm; see, for example, [1].

Triangle packing problem is that given a known length and width of rectangle empty container and N triangles of definite size and shape; namely, the three sides of a triangle are known, and the question is whether the N triangles can be placed into the rectangle container. If we can, then give position and orientation of each triangle in the rectangle. If no, give a negative answer to this problem. Since triangle in the plane can be continuously translated and rotated, there are infinite numbers of placements which are detrimental to solve the problem. And so the placement should be limited always. The strategy in [2] is only to allow a polygon to be rotated, but not to be translated, which translates the polygon packing problem to NP-complete problem.

Triangular packing problem is a special case of a polygon packing problem but is still NP-hard. For now, there is no a quick and complete solving algorithm, and the heuristic method is still an ordinary method; see, for example, [3–7] and references therein. In this paper, a edge-conjoining and corner-occupying algorithm that is a quick and approximate method is designed based on the angle region-filling strategy. This paper has special significance for polygon packing problem, network resource optimization, and so forth, since it is a special case of a polygon packing problem; see, for example, [8–11].

In Section 2, the basic concepts such as angle region are given. The classification of angle region and the corner-occupying action are discussed, and the physical meaning has been analyzed in Section 3. In Section 4, two heuristics strategies and an approximate algorithm for solving triangle packing problem are proposed. Finally, the complexity of
algorithm is also discussed and some results of the simulation example are given.

2. Formation of Angle Region

Definition 1. Triangles in a same plane are said to be intersectant, if the overlapping area of two triangles is larger than zero. Or else the two triangles are said to be non-intersectant.

Definition 2. For two non-intersectant triangles in the same plane, if there exists one intersection at least between two edges which are from each triangle, then the two triangles are said to be tangent, as shown in Figure 1.

Definition 3. For two edges from two tangent triangles, respectively, if the following conditions are satisfied:

(1) There exists only one intersection (O) for the two triangles;
(2) An angle formed by the two edges is a positive angle and less than \( \pi \);
(3) In the angle of (2), there are no other edges of the two triangles except (1);

then the angle in (2) is said to be the angle region of the two tangent triangles. Its size is called the angle of the angle region. The overlapping part of the initial edge of the angle in (2) and one of the two triangles is said to be the initial edge of the angle region; similarly, the overlapping part of the terminal edge of the angle in (2) and one of the two triangles is said to be the terminal edge of the angle region; the intersection in (1) is said to be a vertex of the angle region. As shown in Figure 2, \( \triangle ABC \) is tangent to \( \triangle DEF \) at O(A), and \( \angle BOD \) is an angle region of the \( \triangle ABC \) and \( \triangle DEF \); the angle size of the region angle is denoted by \( \angle BOD \); OB and OD are the initial edge and the terminal edge of the angle region, respectively; O is said to be the vertex of the angle region.

To judge whether there is an angle region formed by two tangent triangles, it is just to take out two edges with common vertex from each of triangles and judge that whether the two triangles form a positive angle less than \( \pi \) and verify that there is no other edges of the two triangles on the angle. The computational procedure can be realized according to vertex coordinate of the triangle.

By the definition of angle region, whether the two edges of the tangent triangles can form angle region is unrelated to the size and shape of the triangle that will be put into the angle region.

3. The Classification of an Angle Region and the Corner-Ocupying Action

Depending on the two triangles' different positions, the two tangent triangles may form zero, one or two angle region, which is shown in Figures 3 and 4(a)–4(c) and 5(a)–5(d), respectively.

These angle regions look like the scraps generated during the material machining processes. If we make use of these scraps, then the ratio of utilization of material can be increased on basis of practical production experiences. It is illustrated through practice to sufficiently increase the ratio of material, and then the position of the filter within the vessel should be stable. Whether a plane figure is precarious depends on shiftable degree of freedom and the rotatable degree.
of freedom. For a triangle, the rotatable degree of freedom refers to a clockwise or counterclockwise rotation of one triangle with the origin at one of its vertexes, and here the triangle is still in the rectangle container after be rotated, and the triangle cannot conflict with other triangles around it; namely, there is no overlapping part between the rotated triangle and other triangles, which means you can rotate the triangle freely; similarly, the translatable degree of freedom refers to a translation of one triangle along two opposite directions of one of the straight lines on which one edge of the triangle lies in the barycentric coordinate of the triangle as the initial point, when the triangle is still in the rectangle container after the manipulation of translating, and the triangle cannot conflict with other triangles around it; namely, there is no overlapping part between the translated triangle and other triangles, which means you can translate the triangle freely.

If a triangle with the barycentric coordinate as the initial point can translate freely along any straight line where a edge of the triangle lies, then the triangle is deemed to be impeding. Apparently, the position of this kind of triangle is unstable, such as the triangle 1 shown in Figure 6.

If a triangle can translate freely along one straight line only or rotate freely around a certain vertex, this kind of position is pushable and swingable, and also it is unstable, such as the triangles 2 and 3 shown in Figure 6.

Generally speaking, the position where a triangle cannot translate freely along one straight line only or rotate freely around a certain vertex is stable, such as the triangles 4, 5, 6, and 7 shown in Figure 6.

If one edge of the triangle 1 is conjoined with a edge of another triangle 2, namely, the length of the overlapping part is more than zero and the triangle 1 can only translate along
the edge of the triangle 2, then the position of the triangle 1 is pushable. Pushing the triangle 1 along the edge of the triangle 2, if the triangle 1 is just tangent to a third triangle and the edge of the triangle 1 is still conjoined with the edge of the triangle 2, then the position of the triangle 1 is stable.

In general, compared with the positions of triangles 4 and 7, the positions of triangles of 5 and 6 can easily divide the space, which is bad for making use of space. It is reasonable to assume that the positions of triangles 4 and 7 are better than those of triangles 5 and 6. For this reason, we have the following definitions.

Definition 4. Corner-occupying action: If one edge of a triangle is conjoined with a initial or terminal edge of a angle region, namely, the length of the overlapping part is more than zero and another edge of the triangle is conjoined with a terminal or initial edge of the region, then the process of putting the triangle into the angle region is called a corner-occupying action. In the case, the angle opposite to the third edge is called the cut-in angle.

It can find a corner-occupying action in this way: pushing a edge which is conjoined with a initial or terminal edge of the triangle region of a triangle outside a rectangle container into the rectangle container, if when the triangle cannot be put away any more (when the triangle is tangent to the terminal or initial edge), the edge of the triangle is still conjoined with the initial or terminal edge of the angle region, then the process of putting the triangle into the position is called corner-occupying action. Therefore, the problem of putting a triangle into an angle region formed by two tangent triangles can be transformed to a issue of position relation between the edge of triangle and the line segment. Thereby, the search space can be confined in a limited number of points from a continuous Euclidean space.

Definition 5. The conjoint degree of a corner-occupying action: If one edge (the length is \(a_1\)) of a triangle and one edge (the length is \(b_1\)) of an angle region are conjoint and let the length of the overlapping part be \(c_1\), then the conjoint degree of corner-occupying action is \(c_1/\max(a_1, b_1)\). Specially, if another edge (the length is \(a_2\)) of the triangle and another edge (the length is \(b_2\)) of the angle region are conjoint and the length of the overlapping part is \(c_2\), then the conjoint degree of corner-occupying action is \(c_1/\max(a_1, b_1) + c_2/\max(a_2, b_2)\).

Definition 6. For two edges of an angle region, if the angle between a edge which is conjoined with a triangle and another edge which is tangent to the triangle is \(\alpha\) and the size of the cutin angle is \(\beta\), then the conformity degree of the corner-occupying action is \(\min(\alpha, \beta)/\max(\alpha, \beta)\).

The conjoint degree and the conformity degree are used to measure the adjacent degree between the filling triangle and the position where the filling triangle will be placed. The bigger the conjoint degree and the conformity degree of a corner-occupying action are, the higher the adjacent degree is.

4. An Algorithm of Edge-Conjoining and Corner-Occupying

As shown in Figure 7, a rectangle container can be deemed as a plane figure circled by four triangles.

Definition 7. Pattern: A pattern refers to a kind of ordered pair \((P, Q, R)\), where \(P\) is the set of four triangles forming the rectangle container which some triangles have been put in (each element in the set is denoted by three vertices coordinates of the triangle), \(Q\) is the set of triangles out of the rectangle container (each element in the set is denoted by three edges of the triangle), and \(R\) is a set of angle region formed by triangles in \(P\).

At the first moment, \(P\) contains only four triangles forming the rectangle container, and \(Q\) is the set of all triangles to be put in the rectangle container, and \(R\) is a set of four angle regions formed by four triangles in \(P\), and then the pattern at this moment is called the initial pattern, which is denoted by \((P_0, Q_0, R_0)\). Starting with the initial pattern, the pattern formed by putting the kth triangle taken out from \(Q\) into the rectangle container is called kth pattern, which is denoted by \((P_k, Q_k, R_k)\). In the kth pattern, if \(N_k = 0\), then it is called the terminal pattern.

In the kth pattern before the terminal pattern, let a triangle perform corner-occupying action, if the triangle is still in the rectangle container and do not intersect with other triangle in \(P_k\), then the corner-occupying action is called a reasonable corner-occupying action.

Corner-Occupying Strategy: For two reasonable corner-occupying actions \(a_1\) and \(a_2\), let the conformity degree of both be...
\(p_1\) and \(p_2\), respectively. If \(p_1 > p_2\), then \(a_1\) is said to have a higher priority than \(a_2\).

**Edge-Conjoining Strategy.** For two reasonable corner-occupying actions \(a_1\) and \(a_2\), let the conjoint degree of both be \(q_1\) and \(q_2\), respectively. If \(q_1 > q_2\), then \(a_1\) is said to have a higher priority than \(a_2\).

Based on the edge-conjoining strategy and the corner-occupying strategy, a heuristic algorithm for solving triangle packing problem is provided below.

Given the initial pattern \(\langle P_0, Q_0, R_0 \rangle\), where \(P_0\) is the set of four triangles forming the rectangle container, \(Q_0\) is the set of all triangles to be put in the rectangle container, and \(R_0\) is a set of four angle regions formed by four triangles in \(P_0\). Arrange the angle region and the reasonable corner-occupying action in time order.

**Step 1.** Set \(t = 0\), and the current pattern \(S = S_t\).

**Step 2.** If \(Q_0 = \emptyset\), then it is reported that all triangles have been put into the rectangle container, and stop; if \(Q_0 \neq \emptyset\) and there is no reasonable corner-occupying action to perform in \(S_t\), then the failure of algorithm is reported, and go to Step 3; if \(Q_0 \neq \emptyset\) and in \(S_t\), there exists a reasonable corner-occupying action to perform, and then go to Step 3.

**Step 3.** Choose a reasonable corner-occupying action with the highest priority to perform according to corner-occupying strategy. If there exists only one reasonable corner-occupying action according to corner-occupying strategy, then fill the triangle according to this corner-occupying action, and go to Step 5; if not, choose a reasonable corner-occupying action with the highest priority to perform according to conjoining-occupying strategy, and go to Step 4.

**Step 4.** Choose a reasonable corner-occupying action with the highest priority to perform according to conjoining-occupying strategy. If there exists only one reasonable corner-occupying action according to conjoining-occupying strategy, then fill the triangle according to this corner-occupying action; if not, fill the triangle according to the first corner-occupying action.

**Step 5.** Add three vertical coordinates of the triangle performing the corner-occupying action to the \(P_t\), and remove three edges of the triangle performing the corner-occupying action from the \(Q_t\). Change \(P_t\) to \(P_{t+1}\), \(Q_t\) to \(Q_{t+1}\), and \(P_{t+1}\) to \(R_{t+1}\) which is the set of angle regions formed by triangles in \(P_{t+1}\). Switch to a new pattern \(\langle P_{t+1}, Q_{t+1}, R_{t+1} \rangle\), and go to Step 2.

### 5. Time Complexity Analysis of the Algorithm of Edge-Conjoining and Corner-Occupying

For the initial pattern \(\langle P, Q, R \rangle\) if there are \(m\) triangles in \(P\) and \(n\) triangles in \(Q\) without generality, find out the reasonable corner-occupying action with the highest priority and calculate out the \(f(m, n)\) time units (step number of calculating) spent on performing the action.

Firstly, calculate the time complexity of all the angle regions formed by \(m\) triangles in \(P\).

Given two triangles, take out one edge from each of them, if it will take \(a\) time units to judge whether the two edges can form an angle region. Then, it will take \(C(3, 1) \ast C(3, 1) \ast a = 9a\) time units at most to find out all the angle regions formed by two given triangles.

It will take \(C(m, 2) \ast 9 \ast a\) time units to find out all the angle regions formed by \(m\) triangles in \(P\). By the definition of angle region, two triangles can form two angle regions at most. Therefore, there are \(2 \ast C(m, 2)\) angle regions in \(R\) at most. In fact, there is little chance that a triangle is tangent to every triangle, where all the triangles are in the same rectangle container. Generally speaking, the total angle regions formed by \(m\) triangles are far less than \(2 \ast C(m, 2)\). This is the root cause of the high speed of the proposed algorithm.

Secondly, calculate the time complexity of all the corner-occupying actions of putting \(n\) triangles which are in the \(Q\) into the angle regions in \(R\).

If it will take \(b\) time units to put one triangle into the angle region along one of the edges of the triangle which is conjoined with a edge of the angle region, and \(22 \ast b\) time units at most to put one triangle into a given angle region with \(22\) different corner-occupying actions at most, then it will take \(2 \ast C(m, 2) \ast 22 \ast b\) time units at most to put one triangle in \(Q\) into all the angle regions in \(R\), and \(2 \ast C(m, 2) \ast 22 \ast b \ast n\) time units at most to complete the corner-occupying actions of putting \(n\) triangles in \(Q\) into all the angle regions in \(R\).

Thirdly, calculate the time complexity of judging whether each corner-occupying action is a reasonable corner-occupying action.

If it will take \(c_1\) time units to judge whether two triangles are intersecting and \(c_1\) time units to judge whether a triangle is in the rectangle container, then it will take \(2 \ast C(m, 2) \ast 22 \ast (m \ast c_1 + c_2)\) time units to find out all reasonable corner-occupying actions.

Finally, calculate the time complexity of finding out the corner-occupying action with the highest priority from all the reasonable corner-occupying actions.

If it will take \(d\) time units to compare the priorities of two corner-occupying action, then it will take \(2 \ast C(m, 2) \ast 22 \ast n \ast d\) time units at most to find out a reasonable corner-occupying action with the highest priority. From the analysis above, the time spent on taking out the triangle corresponding to the corner-occupying action with the highest priority from \(Q\) is at most

\[
C^2_m \ast 9 \ast a + 2 \ast C^2_m \ast 22 \ast b \ast n
+ 2 \ast C^2_m \ast 22 \ast (m \ast c_1 + c_2) + 2 \ast C^2_m \ast 22 \ast n \ast d
\leq 4 \ast 22 \ast m^2 \ast (m - 1) \ast n \ast k_0 \ast k_0 (k_0 = \max(a, b, c_1, c_2, d))
\leq k \ast m^2 \ast (m - 1) \ast n \quad (k = 88 \ast k_0).
\]
Table 1: Experimental data.

<table>
<thead>
<tr>
<th>Example no.</th>
<th>Edge lengths of rectangle containers</th>
<th>Edge lengths of triangles</th>
<th>Graphic outputs</th>
<th>Algorithm results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80, 80</td>
<td>(80.0, 56.6, 56.6) × 2</td>
<td>Figure 8(a)</td>
<td>two triangles left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(40.0, 40.0, 56.7) × 4</td>
<td></td>
<td>Utilization: 75.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Computing time: 0.008 sec</td>
</tr>
<tr>
<td>2</td>
<td>700, 400</td>
<td>225.0, 275.5, 160.0</td>
<td>Figure 8(b)</td>
<td>one triangle left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>166.0, 174.9, 55.0</td>
<td></td>
<td>Utilization: 71.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>278.9, 134.0, 245.0</td>
<td></td>
<td>Computing time: 0.006 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>229.0, 190.0, 299.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>108.0, 260.0, 279.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>296.0, 278.2, 300.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>167.6, 293.2, 273.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>200, 200</td>
<td>(100.0, 100.0, 100.0) × 6</td>
<td>Figure 8(c)</td>
<td>Success!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Utilization: 65.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Computing time: 0.005 sec</td>
</tr>
<tr>
<td>4</td>
<td>400, 300</td>
<td>166.0, 174.9, 55.0</td>
<td>Figure 8(d)</td>
<td>Success!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>278.9, 134.0, 245.0</td>
<td></td>
<td>Utilization: 96.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>229.0, 190.0, 299.0</td>
<td></td>
<td>Computing time: 0.007 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>108.0, 260.0, 279.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>296.0, 278.2, 300.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>167.6, 293.2, 273.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At this moment, a triangle has been put in the rectangle container, and then switch to a new pattern. it will take \( k \times m^2 \times (m-1) \times n \) time units at most for each step. In other words,

\[
f(m, n) \leq k \times m^2 \times (m-1) \times n.
\]

(2)

Similarly,

\[
f(m + 1, n - 1) \leq k \times \left( m + 1 \right)^2 \times (m) \times (n - 1);
\]

\[
\vdots
\]

\[
f(m + i, n - i) \leq k \times \left( m + i \right)^2 \times (m + i - 1) \times (n - i)
\]

(3)

where \( f(m + n, 0) = 0 \). From the initial pattern, it will reach the end of the algorithm after \( n \) steps at most. By the analysis above, the time complexity analysis of the proposed algorithm is

\[
\sum_{i=0}^{n-1} f(m + i, n - i) = k \times \sum_{i=0}^{n-1} (m + i)^2 \times (m + i - 1) \times (n - i)
\]

\[
\leq k \times (m + n - 1)^2 \times (m + n - 2) \times n^2
\]

\[
\leq k \times (m + n)^3 \times n^2.
\]

(4)

In the initial pattern, \( m = 4 \), \( n \) is the number of the triangles to be put in the rectangle container. So

\[
\sum_{i=0}^{n-1} f(m + i, n - i) \leq k \times (n + 4)^3 \times n^2 = O \left( n^5 \right).
\]

(5)

That is to say, the time complexity of the proposed algorithm is \( O(n^5) \). If there is no reasonable corner-occupying action to perform, then stop. Thus, the order of magnitude of the total computing time is not more than \( O(n^5) \).

6. Some Illustrative Examples

In this section, we have developed a program to verify the effectiveness of the proposed algorithm, and the experimental data is included in Table 1. This program is developed by using Visual C++6.0, which is realized on a computer with Pentium 2.8 GHz processor and 512 MB of RAM.

Input is the following:

- the length and width of rectangle container (error is 0.1),
- the edge length of each triangle to be put into (error is 0.000001).

Output is the following:

- the utilization (error is 0.1%),
- computing time (error is 0.01 second),
- the sequence to put the triangles into the rectangle container,
- the vertex coordinates of triangles to be put,
- the calculating error is 0.000001; the display unit is pixels.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Acknowledgments

This work was supported in part by the Key Project of Technology Department of Henan Province of China under Grant 122102210042, the Scientific and Technological Brainstorm Project of Henan Province of China under Grant 12B520054 and the 863 Program of China under Grant 2011AA01A201.

References


