Article

Not peer-reviewed version

# The Various Radii Circle Packing Problem in a Triangle

Ching-Shoei Chiang * and Yi-Ting Chiang

Posted Date: 12 July 2024

doi: 10.20944/preprints202407.0931.v1

Keywords: Malfatti's problem; Geometric Constraint Solver; Computer Aided Geometric Design; Circle Packing line

*Article*

# The Various Radii Circle Packing Problem in a Triangle

**Ching-Shoei Chiang * and Yi-Ting Chiang**

Computer Science and Information Management, Soochow University, Taipei, Taiwan; lhsxb22@gmail.com
* Correspondence: chiang@gm.scu.edu.tw

**Abstract:** The Malfatti's Problem solves the problem of fitting 3 circles into a triangle such that these 3 circles are tangent to each other, and each circle is also tangent to a pair of the triangle's sides. Furthermore, the problem has been extended to have $T_n$=1+2+…+n circles inside the triangle with special tangency properties among circles and triangle sides; it is called the extended Malfatti's problem or Tri($T_n$) problem. In the extended malfatti problem, the numbers of circles in the triangle are triangle numbers, because the tangency properties among the internal circles and 3 sides of the triangle have special type of structure. That is, the corner circle tangent to two sides of the triangle and two boundary circles, the boundary circles tangent to one side of the triangle and 4 other circles, and the inner circles always tangent to 6 other circles. The circles we find in extended general Malfatti's problem has the property that the smallest radius and the largest radius for the circle has big difference. In this paper, we proposed algorithms to solve the problem that the tangency properties among circles and sides of triangle is not fixed, so that the number of circles in the triangle is not necessary a triangle numbers. The purpose of this change is try to find the radii of the circles in the triangle is within a small range.

**Keywords:** Malfatti's problem; geometric constraint solver; computer aided geometric design; circle packing line

## 1. Introduction

Before we explain the problem, we introduce the related material such as Malfatti's problem and extended Malfatti's problem.

### *1.1. Malfatti's Problem and Extended General Malfatti's Problem*

In 1803, Gian Francesco Malfatti [1] proposed the problem: how to find the maximum total volume of three cylindrical columns out of a triangular prism of marble.

Identically, the problem can be recognized as to find three circles to fit in specific triangle and make sure the total area is maximum. Malfatti thought the solution is to make three circles tangent to each other and each circle tangent to two sides of the triangle. A case is shown as Figure 1 [2].

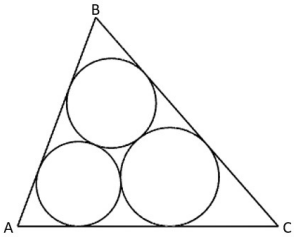



**Figure 1.** Malfatti's problem.

In fact, Malfatti's thought is not correct. In 1930, Lob and Richmond [3] found that a way to get a larger area of some triangles by using greedy algorithm that inscribes a single circle of maximal

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



radius within the triangle, inscribes a second circle within one of the three remaining corners of the triangle, the one with the smallest angle, and inscribes a third circle within the largest of the five remaining pieces. The difference in area for an equilateral triangle is just about over 1% (as shown in Figure 2), [4] but as Howard Eves (1946) mentioned, for an isosceles triangle with a very sharp apex, the optimal solution has nearly twice the area of the Malfatti circles. (as shown in Figure 3) [5] In 1960s, later proven firmly, Lob-Richmond procedure always gives the largest area and solved the problem. In the literature, the Malfatti's thought has become understood as the Malfatti's problem.

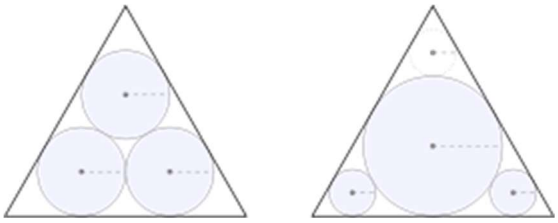


**Figure 2.** In an equilateral triangle.

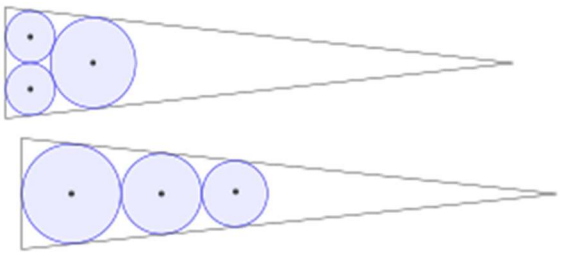


**Figure 3.** Malfatti's problem and max-area problem.

There are several known solutions to Malfatti's problem. Fukagawa and Pedoe [6] mentioned that Chokuen Ajima actually formulated and solved the Malfatti's problem. For a detailed history of the problem and explanations of various solutions and generalizations, interested readers are referred to [7–9].

The Extended Malfatti problem [10] aims to determine $T_n$ circles that can be inscribed inside a triangle while maintaining tangency among the circles and the triangle's sides. When considering the problem with $T_n=\sum_1^n i$ circles, it is referred to as the Tri($T_n$) n problem (as shown in Figure 4). Notably, when n=1, the problem seeks to find the inscribed circle of a triangle, and when n=2, it is precisely the Malfatti's problem.

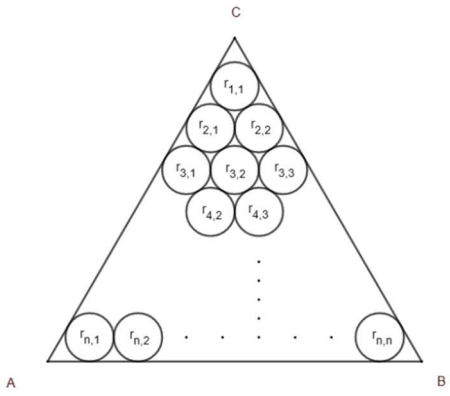


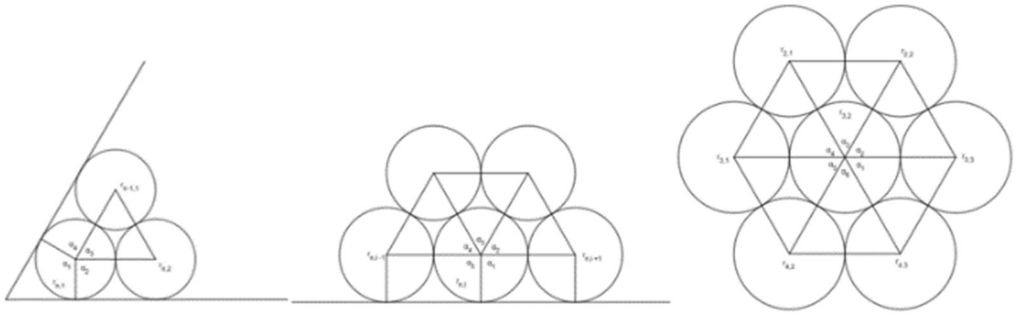

**Figure 4.** The tangency property for Tri($T_n$) problem.

There are closed forms for finding inscribed circle and malfatti's circles. So, when n=1 or n=2, it is easy to propose a constant time algorithm to solve these problems. Chiang [10] proposed algorithms to solve the Tri($T_n$) problem when n>2.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



The main idea of the algorithm [10] is to calculate total surrounding angles for each circle and change the circle's size depends on the sum of angles. (shown in Figure 5). There are three kinds of circles, which is corner circle, edge circle, inner circle. (from left to right in Figure 5). Circles are surround by other circles and sides of the triangle. Every circle is surround by a series of object (other circles and sides of the triangle), and correspond angle between center of this circle to two objects, center of other circle of the nearest points on the sides of the triangle) can computed. If the total surrounding angles is greater than $2\pi$, the radius of the circle should be enlarged. Otherwise, the radius of the circle should be redused. There are different ways to compute the surrounding angles for corner circles, edge circles [10], and inner circles [12]. With these constraints is not enough to find one solution. Consider Tri($T_2$) problem, there are only corner circles in this problem (See Figure 5a, assume these three circles has radii $r_1$, $r_2$, and $r_3$. When we multiply these radii by a constant k, the angle computation (in Figure 5a) for $\alpha_1$, $\alpha_2$, $\alpha_3$, $\alpha_4$ have the same result. It means that there are infinitely many solutions in this case. It need one more constraint to find one solution only. The constant k can be computed by the comparison of the length in one side of the triangle with correspond side of triangle bounded these three circles, as shown in Figure 6. This constant k is called the change ratio for the current solution and the real solution.



(**a**) Corner Circle (**b**) Edge Circle (**c**) Inner Circle

**Figure 5.** Calculating total surrounding angles for 3 kinds of circles.

So, the algorithm Chiang [10] proposed is:

1. Input the coordinate of 3 vertices A, B, C; Calculate the angle A, B, C, and the edge length c.
2. Set a tolerance variable eps = 0.01
3. Initial radius list r= [$r_{i,j}$ |where $1 \le i \le n$, $1 \le j \le i$.]
4. Calculate the surrounding angle list for all circles $C_{i,j}$ (whose associated radius is $r_{i,j}$), call them $\theta_{i,j}$.
5. While one of $|\theta_{i,j}-2\pi| > eps$:
   - 5.1 Modify the radius list, all radii value is computed by current radii, and assign to new radius list in the meantime.
   - 5.2 Calculate the surrounding angle list for all circles $C_{i,j}$(using new radii), call them $\theta_{i,j}$.
   - 5.3 From the radius list, calculate the change ratio, and modify the radius list from the change ratio.
6. Display the result.

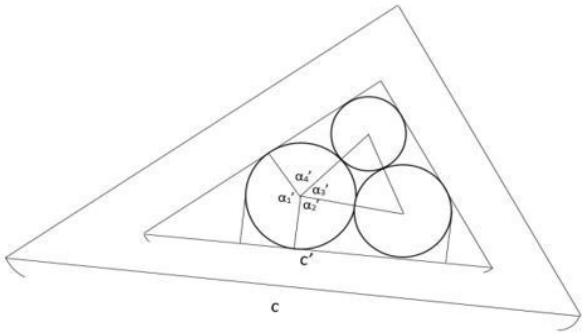



**Figure 6.** Tri($T_2$) problem, change ratio = c′/c.

The result for Tri($T_n$) problem, n=2,3,4,8,31, using Chiang's algorithm [10] is shown in Figure 7.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



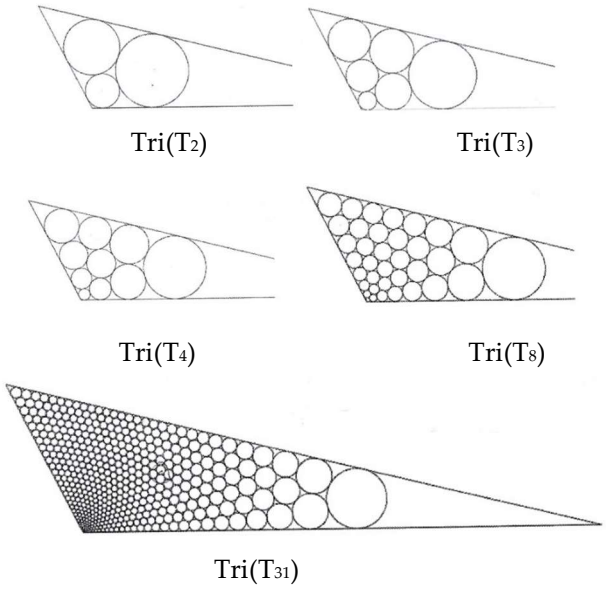



**Figure 7.** Tri(Tn) Problem, where n=2, 3, 4, 8, 31.

*1.2. Problem Statement*

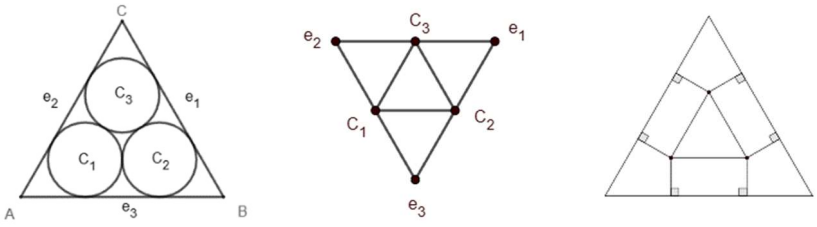
We would like to modify Chiang's Algorithm, so that the radii of these circles have small range. Consider the result in Figure 7, the smallest angle of triangle is corresponding to the biggest circle. On the other hand, the largest angle is corresponding to the smallest circle, especially for the acute angle. In the circle packing area, some problem fixed the radius of the circles, or at least similar size of circles inside the specified region. So, we propose new algorithm to modify Chiang's algorithm, so that the circles radii have small range of the radii.

There are four sections in this paper, the first section illustrates the Malfatti's problem and extended Malfatti problem. The second section define some terms, derived theories and proposed algorithms for the new problem. The third section describe the implementing details. The conclusion is at the final section.

## 2. Term, Theories and Algorithms

We want to find circles inside a triangle, and there are specified tangency constraints among these circles and 3 sides of the triangle. We use the tangency graph to show the tangency constraints. Consider a graph G (V, E), V is the vertices which represents the circles or sides of triangles, and E represents the edge connecting vertices. When an edge connects two vertices, it means that these two geometric object tangent to each other. That is, circle externally tangent to circle if these two geometric objects are circles, or circle tangent to side of triangle. When a circle is tangent to two, one, or zero sides of a triangle, we call it a corner circles, edge circle or inner circle respectively.

We use Tri($T_2$) as an example (see Figure 8a), its tangency graph G(V, E) where V={$c_1$, $c_2$, $c_3$, $e_1$, $e_2$, $e_3$} and E={($c_1$, $c_2$),($c_1$, $c_3$),($c_2$,$c_3$),($c_1$,$e_2$),($c_1$,$e_3$),($c_2$,$e_1$),($c_2$,$e_3$),($c_3$,$e_1$),($c_3$,$e_2$)}, as shown in Figure 8b. To see more clearly for the edge circle or corner circle tangent to side of triangle, we draw the tangency graph as in Figure 8c.




(a) Tri($T_2$) problem (b) Tangency graph (c) our tangency graph

**Figure 8.** Tangency graph.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



The main idea of the new proposed algorithm is to generate the tangency graph, find solution radii for all circles satisfied the constraints specify by the tangency graph, and then display. The purpose of this algorithm is to find circles with small range of radii. In the very beginning, we need to set a value r, so that all circle we generate to pack inside the triangle has radius near r.

So, the main idea of the proposed algorithm is:

1. Input the coordinate of 3 vertices A, B, C; Calculate the angle A, B, C, and the edge length c.
2. Set a tolerance variable eps = 0.0001, epsr=0.3. Set a value r. set i=1.
3. Generate the r-offset of the triangle. Call it $T_i$.

3.1 Generate the vertices on $T_i$.

3.2 Generate the edges for vertices on $T_i$, including the edges connecting 2 neighbors of vertices A, B, C. (See Figure 13a).

3.3 Generate edges for vertices on Ti to the side of the triangle $\overline{AB}$, $\overline{BC}$, $\overline{CA}$. (See Figure 13a).

4. While Ti did not satisfy some conditions:

4.1 Generate the $\sqrt{3}$r-offset of the triangle $T_i$, call it $T_{i+1}$.

4.2 Generate the vertices and edges on $T_{i+1}$, as we do on step 3.1 and 3.2.

4.3 Generate the edge connect vertices on Ti to vertices on $T_{i+1}$. (See Figure 13b–f)

4.4 i=i+1

5. Decide whether the last triangle contains only one vertex(circle) or 3 vertices(circles) (See Figure 13e,f. Call the last triangle $T_{last}$.
6. Establish the edge of graph to connect $T_{i-1}$ and $T_{last}$. (See Figure 13e,f)
7. Using Chiang's algorithm with tolerance eps to find the solution.
8. Display the result.

We will describe the above algorithm more detail, including the answer of the following 4 questions:

1. How to find the r-offset of a triangle in step 3
2. How to construct the tangency graph (vertices and edges) for the same layer?
3. How to construct the tangency (vertices and edges) for two consecutive layers?
4. How the while loop stops in step 4.

The above 4 questions will be described in section 2.1 to section 2.4.

### *2.1. Offset a Triangle*

Consider three different terms, line, ray and line segment. There is no end point for line, and one end points for ray, and two end points for line segment. Consider the parametric form for a line, ray, and line segment passing through two points $P_1$ and $P_2$, they have the same parametric form $(1-t)P_1+tP_2$ with different range of the parameter. It is a line, ray, line segment when $t\in R$, $t\geq 0$ , $0\leq t\leq 1$ respectively. Notice that the ray has $P_1$ as its end point, denote Ray($P_1$, $P_2$) and with the orientation of Ray($P_1$, $P_2$), we may define the left side and right side of Ray($P_1$, $P_2$). Let L= Ray ($P_1$, $P_2$), we denotes the point $P_1'$ is the point moving $P_1$ to the left side of Ray($P_1$, $P_2$) r distance, denotes $P_1'$ =Move($P_1$, L, r). The Ray($P_1'$, $P_2'$), also denote $Ray_r(P_1, P_2)$, is called the r-offset of Ray($P_1$,$P_2$).

Before the more detail description for proposed algorithm, we have the following theorem:

**Theorem 1.** *r-offset of a ray. Given Ray*($P_1$, $P_2$), *where* $P_1 = (x_1, y_1)$, $P_2=(x_2, y_2)$, *the parametric form for* $Ray_r(P_1, P_2)$ is $(1-t)P_1+tP_2+r\times n_L$, $0 \leq t$, *where* $n_L= \frac{1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}(y_1 - y_2, x_2 - x_1)$. *The implicit form for the line*

 doi:10.20944/preprints202407.0931.v1



*cover* $Ray_r(P_1, P_2)$ is $a'x+b'y+c'=0$ *where* $a' = (y_2-y_1)$, $b' = (x_1-x_2)$, *and* $c' = -\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} - (r \times -\sqrt{(x_2-x_1)^2+(y_2-y_1)^2})$

**Proof of Theorem 1.** The direction of the ray is $P_2$-$P_1$= $(x_2-x_1, y_2-y_1)$. The two vectors $n_L = \frac{1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}(y_1-y_2, x_2-x_1)$ and $n_R = \frac{1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}(y_2-y_1, x_1-x_2)$ are perpendicular to $P_2$-$P_1$. This two vectors points to its left ($n_L$) and its right($n_R$). We can move points on Ray($P_1$, $P_2$) along $n_L$ direction r distance, so the parametric form for $Ray_r$ ($P_1$, $P_2$) is $(1-t)P_1+tP_2+r\times n_L$. Consider two points $P_1'$, $P_2'$, which is the points $P_1$, $P_2$, move to its left. That is $P_i'=(x_i', y_i')=P_i+r\times n_L$, i=1,2. The implicit form for Ray($P_1$, $P_2$) is: ax + by + c = 0, where $a = y_2 - y_1$, $b = x_1 - x_2$, $c = -\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}$. Let the r-offset of Ray($P_1$, $P_2$) denote as $Ray_r$ = a'x + b'y + c' = 0, where $a'=y_2'-y_1'=a$, $b'=x_1'-x_2'=b$, $c'=-\begin{vmatrix} x_1' & y_1' \\ x_2' & y_2' \end{vmatrix} = c + \left(r * \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}\right)$. So, the implicit form of $Ray_r$ is $(y_2-y_1)x + (x_1-x_2)y - \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} + \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \times r = 0$.

**Example 1.** Given two points $P_1$=(1,1), $P_2$=(5,2). Let L= $Ray_1$ ($P_1$, $P_2$), the points $P_1'$= Move($P_1$, L, 1) =(0.757,1.970) and $P_2'$= Move($P_2$, L, 1) =(4.757,2.970). In Figure 9, the L is shown in blue with implicit form x+4y+3=0. Furthermore, the $Ray_1(P_1,P_2)$ = Ray($P_1'$,$P_2'$) is shown in orange with implicit form x+4y+7.123=0.

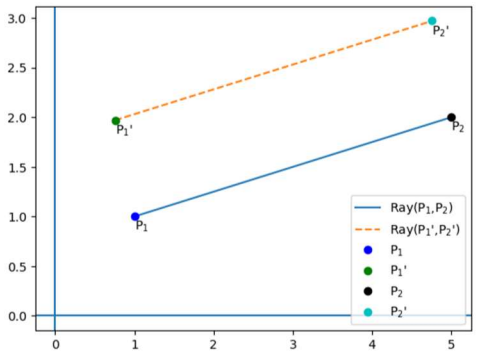



**Figure 9.** r-offset of a ray.

We call Ray($P_1'$,$P_2'$) is the r-offset of Ray($P_1$,$P_2$).
The following theorem is directly from [11].

**Theorem 2:** *Intersection of two lines from 4 points [11]. Given 4 points* $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, $P_4(x_4, y_4)$. *Two lines* $L_1$, $L_2$, *where* $L_1$ *passing through* $P_1$, $P_2$ *and* $L_2$ *passing through* $P_3$, $P_4$. *Then: If* $\begin{vmatrix} (x_1-x_2) & (y_1-y_2) \\ (x_3-x_4) & (y_3-y_4) \end{vmatrix} = 0$, $L_1$, $L_2$ *are parallel to each other. Otherwise, these two lines intersect each other at* (x,y) *where,* $x = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} \\ (x_1-x_2) & (x_3-x_4) \end{vmatrix}}{\begin{vmatrix} (x_1-x_2) & (y_1-y_2) \\ (x_3-x_4) & (y_3-y_4) \end{vmatrix}}$, $y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} \\ (y_1-y_2) & (y_3-y_4) \end{vmatrix}}{\begin{vmatrix} (x_1-x_2) & (y_1-y_2) \\ (x_3-x_4) & (y_3-y_4) \end{vmatrix}}$

**Theorem 3:** *Intersection of two r-offset rays from 3 points. Given* Ray($P_1$, $P_2$), Ray($P_2$, $P_3$), *where* Pi=($x_i$,$y_i$), i=1,2,3. *The intersection of* $Ray_r(P_1, P_2)$, $Ray_r(P_2, P_3)$ is (x, y) = $(\frac{\Delta_x}{\Delta}, \frac{\Delta_y}{\Delta})$ *if* $\Delta \neq 0$ *where*

$$\Delta = \begin{vmatrix} (x_1-x_2) & (y_1-y_2) \\ (x_2-x_3) & (y_2-y_3) \end{vmatrix} \quad (1)$$

$$\Delta_x = \begin{vmatrix} \begin{vmatrix} x_1 & y_1 & -r \\ x_2 & y_2 & -r \\ n_{12x} & n_{12y} & 1 \end{vmatrix} & \begin{vmatrix} x_2 & y_2 & -r \\ x_3 & y_3 & -r \\ n_{23x} & n_{23y} & 1 \end{vmatrix} \\ (x_1-x_2) & (x_2-x_3) \end{vmatrix} \quad (2)$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



$$\Delta_y = \begin{vmatrix} \begin{vmatrix} x_1 & y_1 & -r \\ x_2 & y_2 & -r \\ n_{12x} & n_{12y} & 1 \end{vmatrix} & \begin{vmatrix} x_2 & y_2 & -r \\ x_3 & y_3 & -r \\ n_{23x} & n_{23y} & 1 \end{vmatrix} \\ (y_1 - y_2) & (y_2 - y_3) \end{vmatrix} \quad (3)$$

$n_{12} = (n_{12x}, n_{12y}) = \frac{1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}(y_1 - y_2, x_2 - x_1)$ (4)

$n_{23} = (n_{23x}, n_{23y}) = \frac{1}{\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}(y_2 - y_3, x_3 - x_2)$ (5)

**Proof of Theorem 3.** Let $L_1$=Ray($P_1$,$P_2$), $L_2$=Ray($P_2$, $P_3$), $L_3$=Ray$_r$($P_1$, $P_2$), $L_4$=Ray$_r$($P_2$, $P_3$). Let $P_1'$=Move($P_1$, $L_1$, r), $P_2'$=Move($P_2$,$L_1$,r), $P_3'$=Move($P_2$,$L_2$,r) and $P_4'$=Move($P_3$,$L_2$,r), then: $P_1' = P_1 + r \times n_{12}$, $P_2' = P_2 + r \times n_{12}$, $P_3' = P_2 + r \times n_{23}$, $P_4' = P_3 + r \times n_{23}$, where $n_{12} = \frac{1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}(y_1 - y_2, x_2 - x_1)$ and $n_{23} = \frac{1}{\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}(y_2 - y_3, x_3 - x_2)$. The proof can be easily derived by using theorem 2 with these 4 points $P_1'$, $P_2'$ ,$P_3'$, $P_4'$.

Denote $\Delta = \begin{vmatrix} x_1' - x_2' & y_1' - y_2' \\ x_3' - x_4' & y_3' - y_4' \end{vmatrix} = \begin{vmatrix} (x_1 - x_2) & (y_1 - y_2) \\ (x_2 - x_3) & (y_2 - y_3) \end{vmatrix}$ (6)

$$\Delta_x = \begin{vmatrix} \begin{vmatrix} x_1 & y_1 & -r \\ x_2 & y_2 & -r \\ n_{12x} & n_{12y} & 1 \end{vmatrix} & \begin{vmatrix} x_2 & y_2 & -r \\ x_3 & y_3 & -r \\ n_{23x} & n_{23y} & 1 \end{vmatrix} \\ (x_1 - x_2) & (x_2 - x_3) \end{vmatrix} \quad (7)$$

$$\Delta_y = \begin{vmatrix} \begin{vmatrix} x_1 & y_1 & -r \\ x_2 & y_2 & -r \\ n_{12x} & n_{12y} & 1 \end{vmatrix} & \begin{vmatrix} x_2 & y_2 & -r \\ x_3 & y_3 & -r \\ n_{23x} & n_{23y} & 1 \end{vmatrix} \\ (y_1 - y_2) & (y_2 - y_3) \end{vmatrix} \quad (8)$$

The intersection P (x, y) = $(\frac{\Delta_x}{\Delta}, \frac{\Delta_y}{\Delta})$, when $\Delta \neq 0$

**Example 2.** Given 3 points $P_1$=(1,1), $P_2$=(10,2), $P_3$=(3,5), then $P_1'$=(0.889,1.993), $P_2'$=(9.889, 2.993), $P_3'$=(9.606, 1.080) and $P_4'$=(2.606,4.080). The intersection points P for Ray$_1$($P_1$,$P_2$) and Ray$_1$($P_2$,$P_3$) is at (6.119, 2.575), As shown in Figure 10.

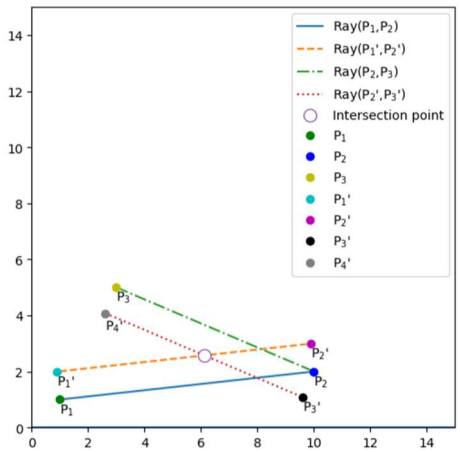



**Figure 10.** Intersection of two r-offset rays from 3 points.

We denote P=Ray$_1$($P_1$,$P_2$)∩Ray$_1$($P_2$,$P_3$).

From △ABC, we can find 3 points B′=Ray$_r$(A,B) ∩Ray$_r$(B,C), C′=Ray$_r$(B,C)∩Ray$_r$(C,A) and A′=Ray$_r$(C,A) ∩Ray$_r$(A,B). We call △A′B′C′ is the r-offset triangle for △ABC.

**Example 3.** Given A=(1,1), B=(10,2), C=(3,5), the 1-offset triangle for △ABC is a triangle with vertices A′=(2.716, 2.196), B′=(6.119,2.575) and C′=(3.472, 3.709), see Figure 11.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



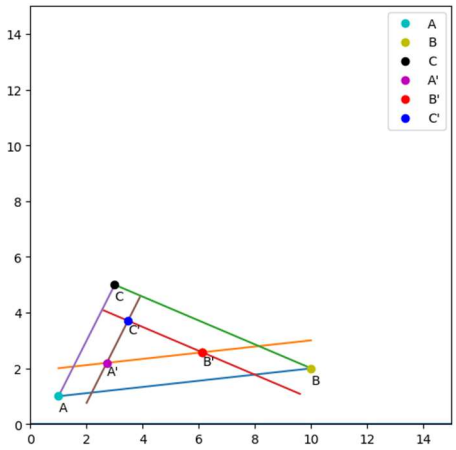



**Figure 11.** The result of r-offset triangle.

*2.2. How to Construct Vertices and Edges on the Same Layer*

The main task for the proposed algorithm is to generate the tangency graphic G (V, E) so that circles in the triangle has small range of radii. So, we need to generate vertices, and edges connect vertices. We generate vertices layer by layer, and generate edges for the same layer and edges connect two consecutive layers. Given a triangle △ABC and circles with radius r, where r is much less than the least length of 3 sides of the triangle. Given △A′B′C′ is the r-offset of △ABC. Then, the circles centered at the sides of △A′B′C′ with radius r tangent to △ABC. We want to find the vertices on sides of △A′B′C′ with the following approach (see Figure 12):

1. The vertices of △A′B′C′ belongs to V.
2. For 3 sides of △A′B′C′:

2.1 Find the a value r′ and integer n, so that |r-r′| is minimum, and 2(r+n*r′) is equal to the length of this side of the △A′B′C′.

2.2 From this r′, find associated vertices, which are center of a series of circles shown in Figure 12, in this side of triangle.

2.3 Connect two consecutive vertices to establish edge of the graph.

3. Connect two neighbors of the vertices of △A′B′C′.
4. Connect edges from $\overline{A'B'}$ to $\overline{AB}$, from $\overline{B'C'}$ to $\overline{BC}$, and from $\overline{C'A'}$ to $\overline{CA}$ (described in the section 2.3)

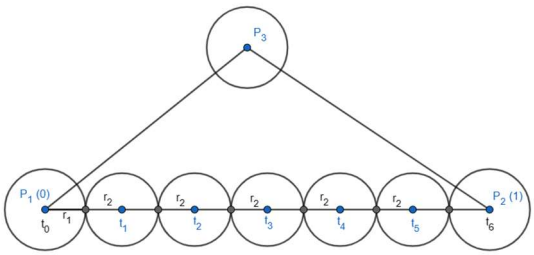


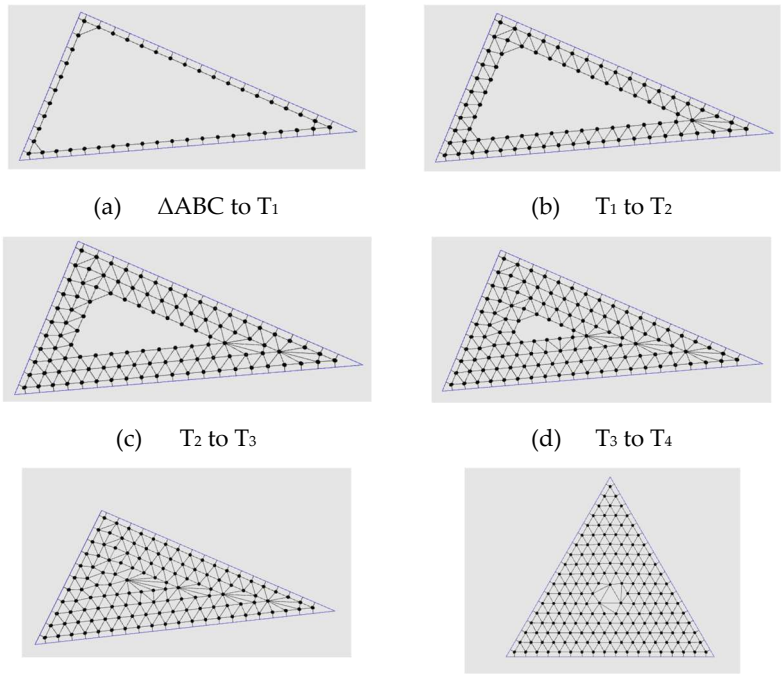

**Figure 12.** vertices setup for one side of triangle.

Using this approach, start from the △ABC, the r-offset of △ABC is $T_1$(△A′B′C′), the vertices and edges for $T_1$ is shown in Figure 13a. Notice that the 3 vertices of triangle $T_1$ has the properties that a circle center at these 3 vertices with radius r tangent to △ABC.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



Now, consider the triangle $T_2$ which is $\sqrt{3}$r-offset of $T_1$. The way to generate vertices and edges for $T_2$ is the same as the way to generate vertices and edges for $T_1$. After we generate edges for vertices in $T_1$ to vertices in $T_2$(will described in the next section), the tangency graph is shown in Figure 13b.



(a) ΔABC to $T_1$ (b) $T_1$ to $T_2$

(c) $T_2$ to $T_3$ (d) $T_3$ to $T_4$

(e) $T_{last}$ contains 1 vertex (f) $T_{last}$ contains 3 vertices

**Figure 13.** Generating the tangency tree for triangle.

*2.3. How to Construct Edges on Two Consecutive Layer*

Now is the time to generate edges between vertices on $T_1$ and vertices on $T_2$. Before we explain our strategy to connect vertices on these two triangles, we have the following theorem:

**Theorem 4.** *Find foot point of a point onto a line. Given a point* g ($x_3$, $y_3$) *and a line* L(g∉L) *passing through two points* ($x_1$, $y_1$), ($x_2$, $y_2$). *The nearest points, also called the footpoint, of* g *onto* L *is at* $(x, y) = (1-t)(x_1, y_1) + t(x_2, y_2)$. *where* $t = \frac{(x_3-x_1)(x_2-x_1)+(y_3-y_1)(y_2-y_1)}{(x_2-x_1)^2+(y_2-y_1)^2}$ (Shown as Figure 14)

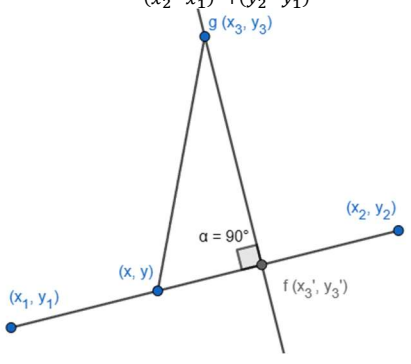



**Figure 14.** Find out the foot point f of point g.

**Proof of Theorem 4.** A point (x,y) on L, we can generate the square of the distance function from (x,y) to point g: f(t) = $(x-x_3)^2 + (y-y_3)^2$=$[(1-t)x_1 + tx_2 - x_3]^2 + [(1-t)y_1 + ty_2 - y_3]^2$. We want to find the value t, so that f(x) is minimum. That is, $\frac{d}{dt}f(t) = 0$, the parameter value t can be computed from $\frac{d}{dt}f(t) = 0$.

Consider the correspondence edge $\overline{A'B'}$ on $T_1$ and $\overline{A"B"}$ on $T_2$. Assume the parameter values for the vertices on $A'B'$ are $t_0$, $t_1$, …, $t_m$ and the parameter values for the footpoints of vertices on

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



$\overline{A"B"}$ onto $\overline{A'B'}$ are $t_0'$, $t_1'$, …, $t_n'$. We explain the details with m=7 and n=3, as shown in Figure 15. To simplify our description, we use the parameter values as its vertex name from now on. The vertex $t_0$ and $t_7$ are the vertices on $\overline{A'B'}$, it did not connect any vertices on $\overline{A"B"}$($T_2$) yet. When we construct edges for △A′B′C′, we construct one edge from $t_1$ to the last vertex on $\overline{C'A'}$., and $t_6$ to the first vertex on $\overline{B'C'}$. Now, we need to construct the edges for $t_i$, i=1...,6 and $t_j'$, j=0,3. We use the following approach to connect edges between vertices generate on $T_1$ and $T_2$. Notice that we try to connect vertices in $\overline{A"B"}$ to the vertices near it in $\overline{A'B'}$. We did the same process for vertices in $\overline{B"C"}$ to the vertices near it in $\overline{B'C'}$ and for vertices in $\overline{C"A"}$ to the vertices near it in $\overline{C'A'}$.

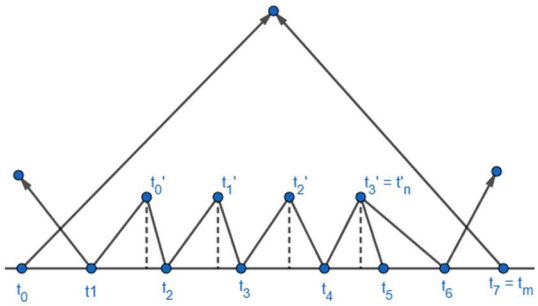



**Figure 15.** relation between vertices.

We define floor and ceiling for the vertices $t_i'$ as the following:

$\lfloor t_i' \rfloor = \max \{t_j \mid t_j < t_i'\}$; $\lceil t_j' \rceil = \min \{t_j \mid t_j > t_i'\}$, Where $\begin{cases} \lceil t_n' \rceil = t_{m-1} \\ \lceil t_{-1}' \rceil = t_1 \end{cases}$, With the above definition, we connect vertices $t_i'$ to all vertices from $\lceil t_{i-1}' \rceil$ to $\lceil t_i' \rceil$ $\forall_i$ = 0, .., n. So, in Figure 15, $t_0'$ connect vertices from $\lceil t_{-1}' \rceil$=$t_1$ to $\lceil t_0' \rceil$=$t_2$, and $t_3'$ connect vertices from $\lceil t_2' \rceil$=$t_4$ to $\lceil t_3' \rceil$=$t_6$. So, vertex with parameter $t_3'$ *in* $A"B"$ has 3 edges connect vertices on $\overline{A'B'}$.

After connecting edges between $T_1$ and $T_2$, the result is shown on Figure 13b. Continue to generate vertices on $T_3$=$\sqrt{3}$r-offset($T_2$), $T_4$=$\sqrt{3}$r-offset($T_3$), and generate vertices and edges for the tangency graph G(V,E), increase vertices on V, and edges connecting vertices on the same layer and previous layer, see Figure 13c,d, it is time to think how to stop the process.

### *2.4. How to Stop the While Loop*

We know any triangles has an inscribed circle and 3 malfatti's circles. The criterion to stop the algorithm depends on the radius of the inscribed circle and the minimum radius of the 3 malfatti's circles.

This process specified the tangency properties among circles, representing vertices on G(V,E). Let $r_1$ be the radius of the inscribed circle for $T_{last}$, and $r_2$ be the minimum radius of the 3 malfatti's circles for $T_{last}$. One of this two radii is less than (1+epsR)r, eqpR is a small value we set, we stop the process and go out of the loop. If $r_1$ is close to r, we use one vertex for $T_{last}$(See Figure 13e), otherwise we use 3 vertices for $T_{last}$(See Figure 13f). The last thing for constructing tangency graph is to connect edges for vertices of $T_{last}$ with the previous layer triangle (See Figure 13e,f).

## 3. Examples

We use Intel(R) Core(TM) i7-10750H CPU, 16G Ram, and Anaconda 23.5.2 with Python 3.11.3 to implement the algorithm. We give two examples as the following:

**Example 4.** Given a triangle with vertices at A=(10,10), B=(100,20) and C=(30,50), and initial r=2. Set epsR=0.3, eps=0.0001, the tangency graph it generates is shown in Figure 13a–e, and after the angle computation for every circles, including vertices circle, boundary circles and inner circles, the result figure is shown in Figure 16. It takes 144.9, 6.140, and 0.288 seconds to construct tangency graph, find the radii for all circles, and display the result respectively. All the circles generated has the radii range (0.829, 4.129), which is (0.415r, 2.065r).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2024 doi:10.20944/preprints202407.0931.v1



In this case, at the last layer, the radius of the inscribed circle is 2.22, which is less than (1+epsR)r=2.6, and the radii of the malfatti's circles are 1.432, 1.099, 1.766, the minimum of them is 1.099. The value 2.22 is closed to r=2, compare to the difference of 2 and 1.099. So we generate the last layer has one vertex only. The output is shown in Figure 16.

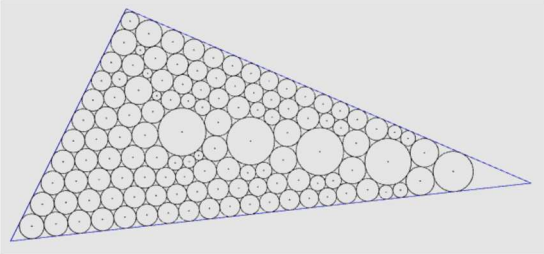


**Figure 16.** Output Display for Example 4.

Notice that there are 4 circles in Figure 16 has larger inner circles. We find that when a circle has large number of surround circle, it has large chance that its radius is large. So, if we add more circle near by, is it possible to reduce the range of the radii for the problem? So, after constructing the tangency graph, we check the degree of each vertex, if the degree is larger than 15 or between 12 and 15, we constructing two vertices/circles or one new vertex/circle with this vertex. With this approach, we have the following example.

**Example 5.** Given a triangle with vertices at A=(10,10), B=(100,20) and C=(30,30), and initial r=1.9. Set epsR=0.3, eps=0.0001. The tangency graph this algorithm construct is shown in Figure 17a, its associated tangency circles in triangle is shown in Figure 17b. Notice that two vertices have more than 15 vertices, so we add two more vertices for each of these two vertex, as shown in Figure 17c and its associated tangency circles in triangle is shown in Figure 17d. Notice that the range of the radii for Figure 17d is smaller than the range of the radii for Figure 17b.

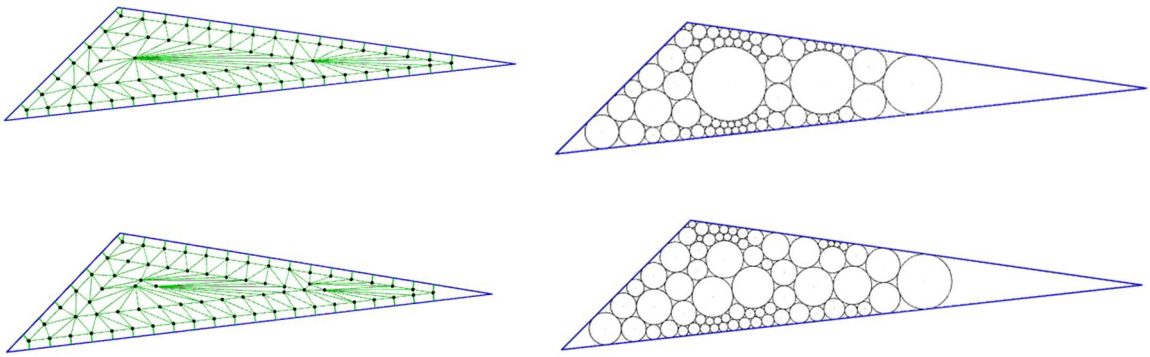


**Figure 17.** Add more vertices in Example 5.

**Example 6.** Given a triangle with vertices at A=(0,0), B=(37.5,64.952) and C=(75,0). Set r=2, epsR=0.4, eps=0.0001, the tangency graph it generates is similar to the last example, except the final step, shown in Figure 13f. After the angle computation for every circles, including vertices circle, boundary circles and inner circles, the result figure is shown in figure 17. It takes 127.106, 1.430, and 0.130 seconds to construct tangency graph, find the radii for all circles, and display the result respectively. All the circles generated has the radii range (1.506, 2.194), which is (0.753r, 1.097r).

In this case, at the last layer, the radius of the inscribed circle is 4.330, and the minimum radius of the malfatti's circles is 2.745. Because 4.330>2.8 and 2.745<2.8, we generate the last layer has three vertices, and connect these 3 vertices to the vertices in the previous layer triangle. The output is shown in Figure 17.

 doi:10.20944/preprints202407.0931.v1



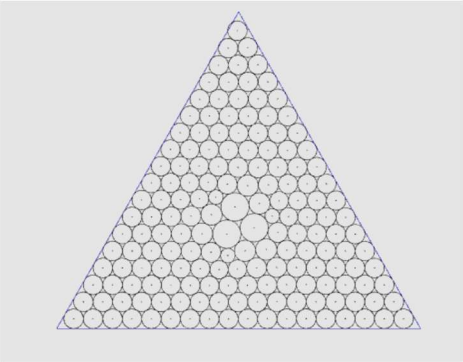


**Figure 18.** Output Display for Example 6.

## 4. Conclusions

We proposed an algorithm to generate the tangency graph to specify the tangency relationship among all circles and triangle boundary. The purpose of this proposed algorithm is to reduce the range of the radii among all circles. The process to generate tangency graph, takes more CPU time, compare to the time spend on construct circle and display circles.

**Acknowledgments:** This research is support NSTC-112-221-E-031-003. We extend our heartfelt gratitude to Guan Wen Wang and Hsi Wen Wang for their invaluable contributions to the programming aspect of this research.

## References

1. Malfatti, Gianfrancesco (1803), "Memoria sopra un problema stereotomico", Memorie di Matematica e di Fisica della Società Italiana delle Scienze, 10: 235–244.
2. Ching-Shoei Chiang, Christoph M. Hoffmann, Paul Rosen, “A gereralized Malfatti’s Problem”, Computational Geometry, Volume 45, Issue 8, October 2012, pages 425-435.
3. Lob, H. and H. W. Richmond (1930). "On the Solutions of Malfatti's Problem for a Triangle." Proceedings of the London Mathematical Society s2-30(1): 287-304.
4. Wells, David (1991), "Malfatti's problem", The Penguin Dictionary of Curious and Interesting Geometry, New York: Penguin Books, pp. 145–146, ISBN 978-0-14-011813-1.
5. Ogilvy, C. Stanley (1990), "Malfatti's problem", Excursions in Geometry, Dover, pp. 145–147, ISBN 978-0-486-26530-8.
6. Fukagawa, H. and Pedoe, D. "The Malfatti Problem." Japanese Temple Geometry Problems (San Gaku). Winnipeg: The Charles Babbage Research Centre, pp. 28 and 103-106, 1989.
7. Bottema, “The Malfatti Problem,” Forum Geometricorum 1, 43-50, 2000.
8. M. Stefanović. “Triangle centers associated with the Malfatti circles.” Forum Geometricorum 3,83-93, 2003.
9. Wolfram Math World. “Malfatti Circles,” Available online: http://mathworld.wolfram.com/MalfattiCircles.html.
10. Ching-Shoei Chiang, Hung Chieh Li, Min-Hsuan Hsiung, and Fan-Ming Chiu, “Extended General Malfatti’s Problem, The 19th International Conference on Scientific Computing, 2021/7/26-29.
11. Weisstein, Eric W. "Line-Line Intersection." From MathWorld--A Wolfram Web Resource. Available online:. https://mathworld.wolfram.com/Line-LineIntersection.html.
12. Collins, C. R., & Stephenson, K. (2003). A circle packing algorithm. Computational Geometry, 25(3), 233-256.