Improvements on circle packing algorithms in two-dimensional cross-sectional areas.

In-Ting Ho
Department of Applied Mathematics, University of Waterloo

August 13, 2015

Contents

Chapter 1

Introduction

Packing is a problem constantly needing to be solved in our daily lives. Before a long flight, travellers face the problem of fitting the greatest amount of clothes and necessities into the smallest number of suitcases in order to minimize extra baggage costs. During rush hour, subway passengers devise ways to fit themselves into an already-crowded train, whether by contorting themselves between other passengers or squeezing into a small space inside the car. Companies load cargo with hopes of packing as many units of product as possible into a container before shipment. Circle packing, in particular, has a number of real-world applications as well, such as cutting discs out rectangular sheets of metal on a factory floor; in this specific example, the challenge for the manufacturer would be to devise a method of minimizing the amount of leftover metal after cutting out discs.

Early mathematical approaches to packing the largest number of circles in a space included that of Thue and Lagrange.[3] Thue's theorem states that, if the centres of circles with identical radii are placed in a hexagonal lattice, the resulting arrangement of circles produces the highest possible packing density in a rectangular region; Lagrange calculated this density to be $\frac{\pi}{\sqrt{12}} \approx 0.9069$, assuming that circles are allowed to be cut off at the boundaries.[3] Furthermore, the problem of circle packing has connections, as conjectured by William Thurston in 1985, to the Riemann mapping theorem.[19] Thurston noted that, after repeated refinement, a circle packing would converge to a classical conformal map.[19]

Finding an optimal packing of circles is an NP-hard problem.[7] Although optimal solutions for packing small numbers of identical circles have been found and proven to be unique (up to symmetry), solutions for packing larger numbers of circles or packing circles of different sizes are much more difficult to optimize. When a computer simulation finds a good initial solution, different packing approaches are subsequently implemented in hopes of finding a marginally better packing.

Packings are determined primarily by two parameters: combinatorial and geometrical flexibility.[20] Combinatorial flexibility concerns the number of possible neighbours for an already-packed circle, subject to its location on the packing surface - that is, whether it lies on the interior or on the boundary.[20] On a plane, the "penny packing" in which every circle of a constant radius has exactly six neighbours, is the packing that best fits circles of constant radii; on a sphere, packing five circles around an already-packed circle on the surface is the best course of action.[20] Regarding geometrical flexibility, Stephenson introduces the notion of circle packings that may overlap with one another in the same space, but in this report, packings will be exclusively univalent, that is, all circles in the packing will be non-overlapping. The way in which these two parameters relate is that the geometry of the packing depends largely on the combinatorics resulting from the way circles, whether identical or non-identical, fit together inside a region. [20]

1.1 Definitions

Definition 1.1. Let i and j be circles in a packing with radii r_i and r_j, respectively. i and j are *tangential* or *tangentially adjacent* to each other if the distance between their centres equals $r_i + r_j$.

Definition 1.2. The *degree* of a circle in a packing is the number of circles that are tangentially adjacent to it.

Definition 1.3. The *fill percentage* of a packing is the total area of packed circles divided by the area of the region in which these circles are packed. The fill percentage is denoted by the symbol ϕ.

Chapter 2

Review of Previous Approaches to the Circle Packing Problem

The majority of the literature on circle packing focuses either on the problem of fitting the largest number of circles with identical radii into a given region, or the similar problem of finding the smallest circle that will contain a given number of smaller circles with identical radii. An alternative formulation of the problem is to maximize the radius of identical circles given a number of identical circles within a circular region.[5] Looking only at centres of circles, it is possible to describe the circle packing problem as a minimization problem, where the goal is to find the "minimum total pairwise distance between all points."[5] By modifying existing constraints or imposing additional ones, researchers have also found solutions for packing non-circular regions and for problems where circles have different radii.

Most approaches to solving the packing problem in circular regions involve algorithms with extensive amounts of non-linear programming. Given two arbitrarily chosen circles already packed into a circular region, Graham *et al.*[5] initially proposed taking the minimum pairwise distance between their respective centres and finding the maximum such distance in the entire packing in an attempt to improve the packing. Starting with an original packing, the algorithm was intended to converge quadratically to a locally optimal solution. However, the non-smooth nature of the first derivatives of the objective function led Graham *et al.* to use the following approximation of the optimization problem for packing n circles instead:[5]

$$\begin{aligned} \max \quad & d(S) \\ \text{where} \quad & d(S) = \min \left\{ \sum_{1 \le i < j \le n} \left(\frac{\lambda}{\|s_i - s_j\|} \right)^m \right\} \end{aligned}$$

where $S = \{s_i, \dots, s_j\}$ is the set of circle centre locations, in $\mathbb{R}^2$.

Graham *et al.* compare this objective function to "a potential energy function when there are repulsion forces between the points" where m is a parameter that accounts for the "strength" of these repulsions, and where λ is an arbitrary constant.[5] Solving the optimization problem uses a "simple steepest descent search with a Goldstein-Armijo backtracking line search." [5] After finding an appropriate search direction, the coordinates of the centres were transformed to create an unconstrained optimization problem. Tightening the packing in order to remove rattlers involves solving an overdetermined system of nonlinear equations. These equations are determined by two types of contact points: ones between circles and the boundary of the region, and ones between the circles themselves. Graham *et al.* proposed using either a modified Newton-Raphson method or a nonlinear least squares method to force a loose packing to converge quadratically to a much tighter one.[5]

Another approach used to pack identical circles in a circular region is the Maximal Hole Degree method proposed by Huang *et al.* The MHD method places the i^{th} circle, with centre c_i and radius

r_i, in a packing if it solves the following optimization problem for λ, the maximal hole degree:[9]

$$\begin{array}{lll} \text{max} & \lambda = 1 - \dfrac{d_{min}}{r_i} & \\ \text{where} & d_{min} = \min \ \left\| c_i - c_j \right\| & i \neq j \\ \text{subject to} & d_{min} > r_i + r_j & i \neq j \end{array}$$

where $c_i, \ldots c_j$ are the locations of the centres of the circles, in $\mathbb{R}^2$.

Several approaches have been used to maximize the number of circles in non-circular regions. The formulation search strategy proposed by Lopez and Beasley involves developing an initial solution before re-perturbing the solution to prevent circles from overlapping. To prevent the packing from violating the constraints imposed by the boundaries of the region, an upper bound for $R_{overlap}$, the largest radius that each circle in a set of identical circles can have, is determined entirely by the geometry of the region in which the circles are being packed.[15]

Region shape	$R_{overlap}$	Symbol definitions
Rectangle	$\frac{LW}{2n\pi}$	L=length, W=width, n = number of circles with an identical radius to be optimized
Triangle	$\frac{L}{\sqrt{2n\pi}}$	L=length
Semi-circle	$\frac{1}{\sqrt{2n}}$	

Lopez and Beasley used perturbations to determine if the initial solution can be improved. If the solution can be improved, a new perturbation size is selected until the solution can no longer be improved. One strategy used by Lopez and Beasley[15] involved representing the coordinates of some subset of circle centres in Cartesian coordinates, and the remainder in polar coordinates. Three approaches to separating centres into either of the two coordinate systems were explored. For each approach, there were nine substrategies that accounted for different values of m, which represents the cardinality of the subset of available circles whose centres had their coordinates represented in a different coordinate system.[15] For all 27 substrategies, the segregating of centres occurred either before or after a packing iteration, but no significant differences in fill percentage were found. The overall concept of grouping already-packed circles by the coordinate system in which their centres are expressed, however, was an idea that produced packings with a "smaller average deviation from best known results". [15]

Several strategies have been used to optimize the packing of circles in non-circular regions, including the simplex method, formalizations of the stochastic Langevin equation, and Cabri-Géomètre software developed by Mollard and Payan.[17] Quasi-Newton methods such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm and "variants of the Fletcher-Powell-Davidson method" have also been used; the BFGS method, however, was unable to optimize packings for 14, 15, and 17 identical circles in a larger circular region.[17]

Birgin and Martinez[2] formulated the packing problem to determine the number of circles in a set of k circles that will fit inside a rectangular region $[0, L]x[0, W]$. Once the distances between pairs of circles i and j, whose centres p_i and p_j are located at coordinates (x_i, y_i) and $(x_j, y_j$ respectively, are calculated, then solutions to the following optimization problem would represent locally, but not necessarily globally, optimal packings. [2]

$$\begin{array}{lll} \text{min} & \displaystyle\sum_{i \neq j} \left(\max \left\{ 0, (2r)^2 - \left\| p_i - p_j \right\|^2 \right\} \right)^2 & \\ \text{where} & p_i := (x_i, y_i) \in \mathbb{R}^2 & i = 1, \ldots, k \\ \text{subject to} & r \leq x_i \leq L - r & \\ & r \leq y_i \leq W - r & \end{array}$$

In order for packings to converge quadratically to some locally optimal solution, the objective function should at least be twice-differentiable. Unfortunately, when $(2r)^2 = \left\| p_i - p_j \right\|^2$ for some pair of circles i and j, the second derivatives of the objective function are discontinuous.[2] To correct this, the objective function is perturbed slightly by introducing the free variable z.

If we assume there are $(m+1)$ circles in the packing, then there exist $\binom{m+1}{2}$ unique pairs of circles, and we would require $m' = \binom{m+1}{2}$ copies of z, i.e. $z = \begin{bmatrix} z_1 & \dots & z_{m'} \end{bmatrix}^T$. If circles i and j are labelled as the q^{th} pair of circles in a packing, then the modified optimization problem, with a twice-differentiable approximation of the objective function, is as follows:[2]

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{q=1}^{m'} \left[g_q(\vec{x}_q) + z_q^2\right]^2 \\ \text{where} \quad & g_q(\vec{x}_q) = (2r)^2 - \left\|p_i - p_j\right\|^2 \quad ; \vec{x}_q \in \mathbb{R}^2 \end{aligned}$$

In two dimensions, the vector $\vec{x}_q$ is an ordered pair that denotes the distance between the q^{th} pair of circles along the x- and y-axes. The ordered pair $(\vec{x}, \vec{z})$, then, is the global minimizer of the approximated optimization problem and satisfies the equation $z_q^2 = \max \left\{0, -g_q(\vec{x}_q)\right\}$, where z_q is the q^{th} component of $\vec{z}$. The existence of such an ordered pair, which Birgin and Gentil[2] called a "good pair", implies that every circle in a given set can be packed into the given region.

In order to solve for $z = \begin{bmatrix} z_1 & \dots & z_q \end{bmatrix}^T$ in the equation $z_q^2 = \max \quad \{0, -g_q(\vec{x}_q)\}$, Newton's method is used to force the convergence of an initial guess for z to a value that produces a global or local minimum for the objective function in the approximated optimization problem. Birgin and Gentil[2] assert that "a [Newton's method] iteration for minimizing [the objective function of the approximated optimization]" and updating z_q^2 by $\max \quad \{0, g_q(\vec{x}_q)\}$ is the same as "a [Newton's method] iteration for minimizing the objective function [of the unapproximated optimization problem]." Furthermore, for some perturbation of a good pair that minimizes the objective function of the approximated optimization problem, the positive semi-definite nature of perturbation makes the perturbed Hessian positive semi-definite as well.[2]

Other packing strategies involved genetic algorithms. Instead of using geometric representations, genetic algorithms use position strings, which determine a sequence of placing circles in the packing. The motivation for George, George, and Lamar to attempt a genetic algorithm was the ability to move circles at will from the top to the bottom of the packing region. [4] To initialize the genetic packing algorithm, two "parent" strings are first created. In George, George, and Lamar, as well as in Hifi and M'Hallah, position strings were created by shuffling randomly the indices of all packable circles. Hifi and M'Hallah went further by using either of three selection procedures to find the best "parent" candidates: "tournament selection, proportionate selection, and fitness scaling."[8] The "parents" then "mate" with each other so that strings exchange subsequences and produce "child" strings. Further "mutations" are then introduced to either create new position numbers in the string,[4] split the string, or change the order of the subsequences.[8] The packings that these child strings represent are not only "decoded" for visual representation, but also tested for feasibility. For example, if a child string is identical to either or both of its parents, it is rejected for redunancy.[8]

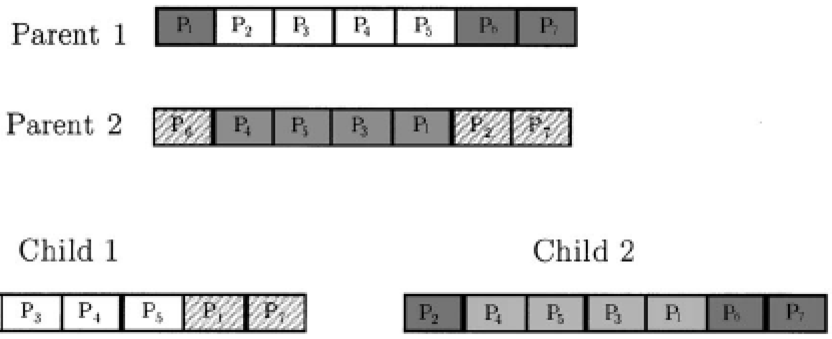

Parent 1
P_1 P_2 P_3 P_4 P_5 P_6 P_7
Parent 2
P_6 P_4 P_5 P_3 P_1 P_2 P_7
Child 1
P_6 P_2 P_3 P_4 P_5 P_1 P_7
Child 2
P_2 P_4 P_5 P_3 P_1 P_6 P_7

Figure 2.1: "Parent" strings 1 and 2 are "mated" to produce "children" strings 1 and 2. Source: [8]

Solutions produced by genetic algorithms are more likely to resemble known optimal packings and require less computational time than other packing heuristics such as placing large circles in corner positions. [4] One limitation of the genetic algorithm is the difficulty of improving the quality of packings obtained. George, George, and Lamar found that the best solution obtained from ten generations of "mating" was just as good as an optimal solution obtained after only three generations. [4]

Stochastic search methods have also been used to develop optimal circle packings. Szabó *et al.* suggested two stochastic algorithms: the Threshold Accepting Method – Single Agent Stochastic Search, Packing Equal Circles in a Square (TAMSASS-PECS), and the Modified Single Agent Stochastic Search (MSASS).[21] Both algorithms use an objective function that acts as an energy function, and treats the coordinates of the centres as electrical charges. The resulting optimization problem aims to minimize the potential energy of the system by increasing the minimum distance between adjacent charges.[21]

In TAMSASS-PECS, initial packings are created by dividing the region into rectangular "tiles" and packing circles in each "tile". The TAM component of the algorithm checks the initial solution if the fill percentage is sufficiently close to that of another candidate packing. If the fill percentage meets the pre-determined closeness threshold, then the new solution becomes the "reference packing".[21] To generate new packings, the SASS component of the algorithm applies perturbations to a candidate solution. Usually, the perturbations involve scaling the size of circles in the packing. The size of a perturbation is chosen randomly with a pre-determined mean size and follows a pre-determined probability distribution.[21] Rao and Karnop used a uniform distribution; Matyas used a normal distribution.[21] If a perturbation is ultimately too small to generate a better packing, the interval of perturbation is expanded, and another perturbation size is chosen at random. Once the SASS component of the TAMSASS-PECS algorithm generates another candidate packing, the closeness threshold is reduced, and the new packing is compared to the "reference packing" once again. The ultimate goal is for packings to converge to the best possible solution.[21]

In MSASS, only the centres of the circles in a packing are considered. Similar to the TAM component of the TAMSASS-PECS algorithm, a pre-determined distance threshold is established to ensure that circles are close enough to each other to maximize fill percentages.[21] Suppose there are two circles in a packing with centres s_i and s_j. The distance between them is $d(i,j)$, and the circle with centre s_j remains stationary. If moving the circle with centre s_i to a different location in the packing, all overlap and boundary constraints considered, implies that the value of $d(i,j)$ is still under the distance threshold, then move the circle with centre s_i to that new position.[21] The idea behind MSASS is to determine whether jiggling around a circle in a packing can maintain the fill percentage of a packing or improve it by creating space for packing additional circles.

Other circle packing methods include the Quasi-Human/Quasi-Physical algorithm, the PERM algorithm, and the dynamic search algorithm. For the Quasi-Human/Quasi-Physical method, the Quasi-Physical component of the algorithm treats the packing problem as the "squeezing and collision of cylinders of unit height along a tray with a hole in the centre of radius r_0."[26] Wang *et al.* assumed that the circles were elastic and all had a spring constant of 1.[26]

Now suppose we have a circular packing region, c_0. Suppose also that there is an arbitrary packed circle c_i in the interior of c_0, has its centre at (x_i, y_i), and is tangential to exactly two other circles – c_j and c_k. Then the force of c_j on c_i is $\vec{f_{j,i}}$ and the total force acting on c_i is $\vec{F_i}$, where:[26]

$$\begin{aligned} \vec{F_i} &= \vec{f_{0,i}} + \vec{f_{j,i}} + \vec{f_{k,i}} \\ \text{where} \quad \vec{f_{j,i}} &= \langle (x_i - x_{j,i}), (y_i - y_{j,i}) \rangle \end{aligned}$$

$\vec{F_i}$, the total force on c_i, is then used to determine how much the position of c_i is perturbed. Suppose $(x_i^{(t)}, y_i^{(t)})$ is the location of the centre of c_i after t perturbations. Then the $(t+1)^{th}$ perturbation of c_i results in the following:[26]

$$\begin{aligned} & (x_i^{(t)}, y_i^{(t)}) + \epsilon \vec{F}_i = (x_i^{(t+1)}, y_i^{(t+1)}) \\ \text{where} \quad & \epsilon > 0 \\ \text{and} \quad & (x_i^{(0)}, y_i^{(0)}) = (x_i, y_i) \end{aligned}$$

A limitation of the Quasi-Physical approach is that, even when packings converge toward a locally optimal packing, taking a circle out of a packing and placing it in a different location, which Wang *et al.* called a "jump", might still improve the packing further. Unfortunately, the only way to create these jumps with a Quasi-Physical approach is to start over from scratch.[26] To create these jumps more easily, Wang *et al.* used a Quasi-Human approach.

The Quasi-Human component of the algorithm treats circles as "members" of a "society" who keep trying to move upwards in the society's "hierarchy". Circles are continually moved towards the top of a packing region until they can no longer move up, i.e. until these "members" of "society" are satisfied.[26] According to Wang *et al.*, using bigger circles to "squish down" smaller circles is akin to poorer members of society being pressured downward by their richer counterparts.[26] When a smaller circle is moved to a new location in the packing, the action is deemed "pain relief" for a "poor" circle; when a bigger circle is moved, it becomes "resource surrender" for a "rich" circle.[26] A locally optimal packing produced by the Quasi-Human/Quasi-Physical algorithm is the solution to a nonlinear optimization problem with "$(2n+1)$ continuous deterministic variables".[26]

PERM, which stands for the Pruned-Enriched-Rosenbluth Method, is a tree search method that finds the best packing for a given set of circles. The method "prunes and enriches branches" while "searching the solution space."[16] The pruning and enriching of branches depends on the weight W_m of a packing of m circles, which is governed by the following recursion:[16]

$$\begin{aligned} & W_m = W_{m-1} \exp\left(\frac{\lambda_m}{T}\right) \\ \text{where} \quad & W_1 = W_2 = 1 \\ \text{and} \quad & \lambda_m = \max \left\{ 1 - \frac{d_{m,min}}{r_m}, 0 \right\} \\ \text{and} \quad & T \in [0.2, 0.5] \end{aligned}$$

In this recursion, λ_m is the hole degree, a metric that evaluates the merits of packing the m^{th} circle.[9] $d_{m,min}$ is the minimum Euclidean distance between the centre of the m^{th} circle and that of the circle closest to it; T is a temperature parameter whose value is randomly chosen in the interval $[0.2, 0.5]$ so that it is both non-zero and small enough in magnitude.[16]

At the very top of any PERM search tree is the rather trivial packing of zero circles.

To create branches of the search tree, Lu *et al.* extrapolated the quality of future packings of m circles by computing the predicted weight, W_m^{pred}, where the k largest hole degrees in the packing are selected and where $i = 1, ..., k$:[16]

$$W_m^{pred} = W_{m-1} \sum_{i=1}^{k} \frac{\exp \frac{\lambda_i}{T}}{k}$$

This predicted weight is compared to a threshold interval $(W_m^<, W_m^>)$, where:[16]

$$\begin{aligned} & W_m^> = C \cdot Z_m \cdot (C_m)^2 \\ & W_m^< = 0.02 \cdot W_m^> \\ \text{where} \quad & Z_m = \frac{1}{C_m} \sum_{j=1}^{C_m} W_m^{(j)} \\ \text{and} \quad & C \in [10^{-9}, 10^{-6}] \end{aligned}$$

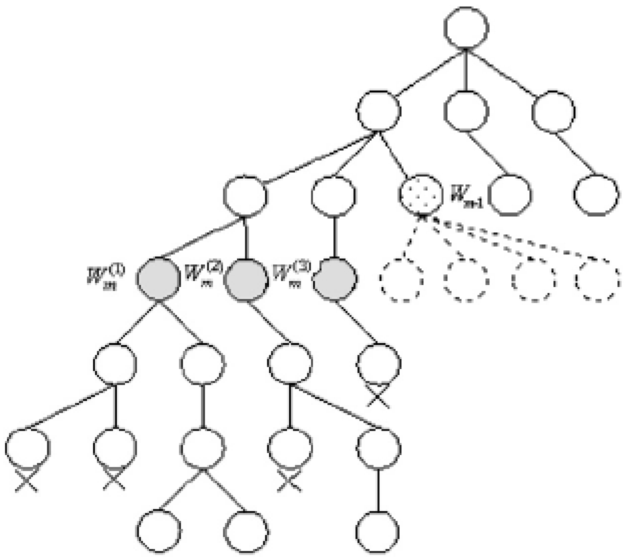


Figure 2.2: A partial PERM search tree. The top "node" in this tree is an arbitrary packing of $(m-3)$ circles. The tree can, of course, be traced back to the trivial packing of 0 circles. Source: [16]

Lu *et al.* defined Z_m as the "numerical average weight of m-circle patterns" and defined C_m as "the number of m-circle patterns" in all. As the search tree grows, Z_m and C_m are "continuously updated."[16]

Computing the predicted weight plays a significant role in determining which branch of the search tree to follow, or equivalently, determining an optimal sequence of packing cirlces. If the predicted weight is under the lower threshold, a branch has a 50% chance of being pruned. If the predicted weight is above the upper threshold, then it implies the existence of many possible placements for the $(m+1)^{th}$ circle. If the predicted weight is inside the threshold interval, then one arrangement of m circles is randomly chosen to continue packing with.[16] Packing continues until the search tree reaches a dead end. Lu *et al.* sought to reduce computational time by imposing an extremely small value for the constant C, which results in fewer possible patterns remaining and implies that a particular branch of the search tree is more likely to be enriched.[16]

The dynamic search algorithm takes an initial packing, and generates a number of available positions for the next circle to be placed.[6] Suppose a square region is being packed with n identical circles, where $n = k^2$ for some integer k. If the first circle is placed in a corner, then the best place for the next circle would be horizontally or vertically tangential to the first. Ultimately, the circles would be arranged into k rows of k circles each, as expected. However, for packing non-identical circles in non-square regions, Hifi and M'Hallah categorized available packing positions into one of three types: (1) a new circle of the largest available size placed next to an already packed circle of the largest available size, (2) a new circle placed tangential to already packed ones, and (3) a circle placed in a unpacked area of the region.[6] By focusing on Type 3 positions and creating as many of them as possible, Hifi and M'Hallah improved initial packings in almost all of their test cases.[6]

2.1 Motivation for Implementing the George-George-Lamar (GGL) Algorithm

George, George and Lamar proposed a packing algorithm for packing pipes into a rectangular shipping container. The idea was to place two pipes in each of the bottom two corners of the container, orient the remaining pipes, and optimize the packing by using gravity to "shake" the pipes down.[4] George, George, and Lamar produced a comprehensive list of eight packing heuristics that could improve packings: (1) place larger circles before smaller circles, (2) keep large circles close to corners, (3) start packing on the outside and work inward, (4) place similar-sized circles together, (5) place 1 circle and place other circles in a way that obeys gravity, (6) place every circle randomly, (7) pack circles and try to squeeze more circles in empty spaces, (8) shake a "box" of circles so they go to the bottom and open up space to pack more.[4] Each candidate packing method developed by George, George, and Lamar revolved around a subset of these heuristics.

While many algorithms focus exclusively on circles of identical radius or a circular packing region, the method proposed by George, George, and Lamar (the "GGL method") focuses on packing circles that may have different radii in regions that are not necessarily circular. Furthermore, many previously mentioned algorithms require at least one "good enough" packing solution before optimization can begin, but the GGL method allows packings to be created from scratch. In order to systematically place subsequent circles into a packing, George, George, and Lamar devised a numbering method, the "position number" system, that labels newly placed circles relative to the location of already-packed circles.

In the next chapter, we will creating an extension of the original GGL method to pack circles in rectangular, non-rectangular quadrilateral, and non-convex regions. Although the side number and position number system will be retained, the definition of a corner will change as the region becomes less rectangular, which then changes the numbering sequence of positions accordingly. A lookup table will also be produced that contains the set of remaining circles available to be packed into the region. It will also serve to more effectively track which circles have already been pack and to avoid packing individual circles more than once.

Chapter 3

Using the GGL Algorithm for Non-Circular Regions

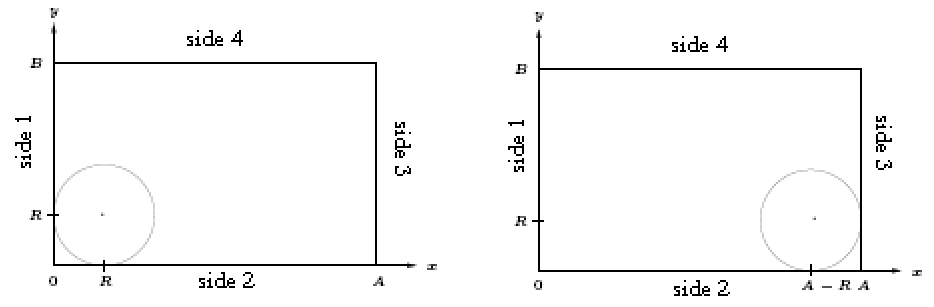
3.1 Definitions

Definition 3.1. An *already-packed circle* in a packing is any of $(n-1)$ circles already packed inside a a region before the n^{th} circle is packed. An arbitrary such circle is defined to have its centre at (x_a, y_a) and radius R_a.

Definition 3.2. A *new circle* is any circle not already packed into a region. An arbitrary such circle is defined to have its centre at (x^*, y^*) and radius R^*.

3.2 Position numbers

Central to the GGL algorithm is the idea of position number strings. In the problem originally introduced by George, George, and Lamar, the first two positions, Positions 1 and 2, refer to the bottom-left and bottom-right corners of a rectangular container, respectively.[4] Once the first circle is placed in a bottom corner, the second circle can be placed tangentially adjacent to the first. In order to keep circles as close to the bottom of the container as possible, George, George, and Lamar ignored the top side of the container, and forced circles to be oriented towards either the bottom, left, or right sides instead.[4]



(a) Position 1

(b) Position 2

Figure 3.1: Visual depiction of Positions 1 and 2 in a rectangular region. Source: [22]

Once the first two circles are placed, a third circle can be placed in either of three ways: (1) tangentially adjacent to only the 1st circle, (2) tangentially adjacent to only the 2nd circle, or (3) tangentially adjacent to both circles. Each of (1) and (2) produce three additional position numbers, one for each non-top side of the rectangular container, whereas (3) generates only one new position number. As a result, having a third circle involves adding 7 new positions in addition to the 2 original corner placements for a total of 9 available positions.

For every subsequent circle packed into the region, 3 additional position numbers will be generated for each previously packed circle — one for each non-top side of the rectangle — and 1 additional position number will be generated for every pair of previously packed circles. Given $(k-1)$ previously-packed circles, then, the number of possible position numbers for placing the k^{th} circle is equal to f_k, where: [4]

$$f_k = \begin{cases} k & \text{if } k = 1 \\ \frac{1}{2}(k^2 + 3k) & \text{if } k > 1 \end{cases}$$

The study conducted by George, George and Lamar established parameters for the circles to obey the laws of gravity because packings were meant to solve the problem of maximizing the number of pipes in a shipping container. In our problem, however, the arrangement of circles need not be constrained to the laws of gravity. While the initial GGL approach, as described in the remainder of this chapter, largely abides by the laws of gravity, subsequent approaches focus efforts towards the centroid of the region.

A real-world application of this scenario involves shipping a box of cylindrical paint cans, where the paint cans are one of three different sizes. If the paint cans are placed vertically in a box and a force of considerable magnitude were to be applied to one of the sides of this box, due to either a vehicle accident or a pothole in the road, then paint would very likely leak from at least some of the paint cans due to impact. However, by using an anti-GGL scheme, paint cans of the largest size would be placed as far away from the sides of the box as possible. In doing so, the paint cans most likely to be damaged would be those of smaller sizes, thereby minimizing the total loss of paint.

3.3 Extension of the GGL packing algorithm

1. Produce a lookup table containing all possible position numbers based on the circles available for packing. From this lookup table, an *availability list* can be generated, which is simply the list of circles not yet packed.
2. Place circles at the intersection of two line segments (”corners”). Placements in corners have designated position numbers. Once a circle is packed in each corner, subsequent position numbers are defined by the circle to which a placement would be tangent.
3. Define the interior of a region by the geometry of the boundary line segments. If the proposed positioning does not result in a circle being packed in the interior of a region, thereby violating geometric constraints, do not pack the circle.
4. Ensure that circles must be packed tangent to each other. For circles along the boundary, ensure that the centres of the circles must be at least one radius away from each other. For circles in the interior, use the cosine law for triangles, with the centres of the circles acting as vertices.
5. Establish a threshold limit for fill percentage. Any packing whose fill percentage falls below this threshold limit will be discarded.

3.4 Constraints for rectangular regions

Since George, George and Lamar had also focused on packing rectangular containers, the first region in which they began packing circles was the rectangle. George, George, and Lamar followed a packing heuristic that focused on placing the greatest number of circles of the largest available size along the sides before packing the interior of the region.

Suppose that there is exactly one already-packed circle in a rectangular region. If we were to pack a new circle tangent to side 1, side 2, or side 3, Vrscay determined the geometric constraints to be:[22]

$$\begin{array}{lrl}
\text{Side 1:} & & x^* = R \\
& & y^* = y_a \pm \sqrt{(R_a + x_a)(2R + R_a - x_a)} \\
& \text{where} & x_a \leq 2R + R_a \\
\text{Side 2:} & & x^* = x_a \pm \sqrt{(R_a + y_a)(2R + R_a - y_a)} \\
& & y^* = R \\
& \text{where} & y_a \leq 2R + R_a \\
\text{Side 3:} & & x^* = A - R \\
& & y^* = y_a \pm \sqrt{(R_a + R)^2 - (x_a - (A - R))^2}
\end{array}$$

Determining the quality of a packing algorithm for rectangles was only a matter of comparing the fill percentages of packings produced with the results obtained by George, George and Lamar. Initially, Qiao [18] proposed using a threshold limit of 0.7, considering most good packings filled at least 70% of the region. However, when the set of circles were simply too small to pack 70% of the region, the threshold limit would lead to the algorithm incorrectly stating that no feasible solution existed. As a correctional measure, a ranking algorithm is proposed to produce the top ten packings of given circles in a given region after a large number of iterations.

Initial experiments packed circles into the rectangular region between 500 and 1000 times, with each packing iteration taking approximately 1-2 minutes of computational time. A major limitation of this ranking algorithm lies in the fact that, to rank all possible packings of n circles, 2^n packing iterations would be required, which is incidentally the power set of the set of all circles available to be packed in the region. 1000 packing iterations would be sufficient to account for all possible packings of n circles if n=10; for n>10, an extremely massive amount of computational time would be required to calculate the fill percentage of all possible packings.

3.4.1 Results

Vrscay conducted four major variations in rectangular packing based on the ordering of available circles and on the radii of the circles. Using one large radius size and one smaller radius size, the GGL procedure produced the packing below.

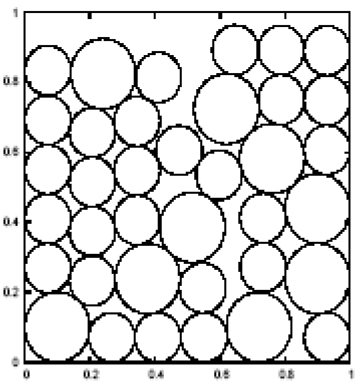


Figure 3.2: Fill percentage of the above packing is 0.7226. Source: [23]

When circles are sorted in the availability list by descending order of radii, the resulting packing is shown in Figure 3.3.

When the radii of circles were forced to abide by a formula where R_i, the radius of the i^{th} circle, is defined by:

$$R_i = 0.15 - 0.003(i - 1)$$

then the resulting packing is shown in Figure 3.4, with a fill percentage of 0.8194.[23]

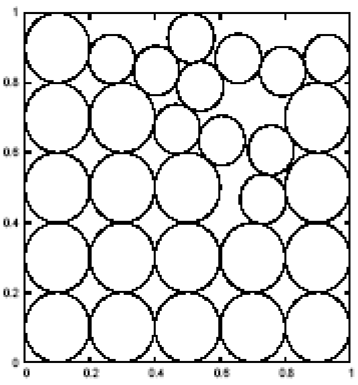


Figure 3.3: Fill percentage of the above packing is 0.7383. Source: [23]

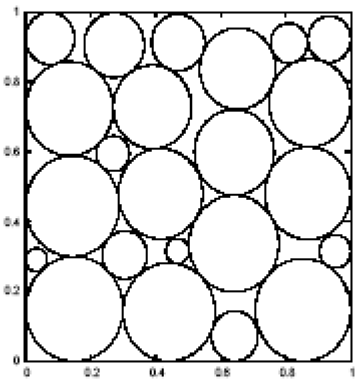


Figure 3.4: After 46 iterations. Fill percentage of the above packing is 0.8194. Source: [23]

3.5 Constraints for trapezoidal regions

For a trapezoid, side numbers are oriented the same way as those of rectangular regions. However, the coordinates of the centre of the newly packed circle in a trapezoid depend entirely on the slope of the side closest to the circle. Whereas the positions designated Position 1 and Position 2 in a rectangular region pertained to the bottom two corners, Position 1 and 2 in a trapezoidal region are the positions where the circle lies tangent to both the x-axis and to either of the two sloping sides.

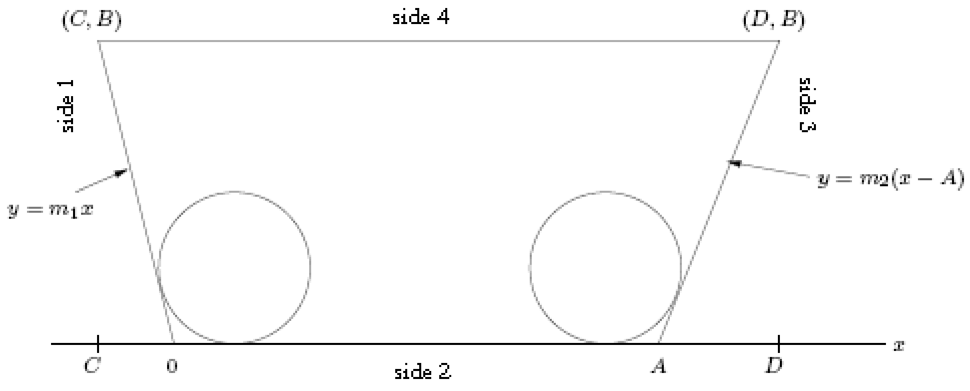


Figure 3.5: Positions 1 and 2 for a trapezoid whose vertices are at (0,0), (A,0), (D,B) and (C,B). Source: [23]

Given an arbitrarily circle that is already-packed, the placement of a new circle will be constrained by the four sides of the trapezoid. In order to prevent a newly-packed circle from intersecting the boundary of the trapezoid at two points, the circle must be kept at a certain distance away from the boundary. Given that a newly-packed circle must be kept a certain distance away from a non-parallel side of the trapezoid, whose slope is $m_i > 0$ where $i = 1,2$ and has a y-intercept of $b > 0$, the minimum distance between the centre and the boundary is:[23]

$$\begin{aligned} d &= \frac{1}{|m_i|} \cdot \frac{\left|y_a - m_i x_a - b\right|}{\sqrt{1 + \frac{1}{m_i^2}}} \\ &= \frac{\left|y_a - m_i x_a - b\right|}{\sqrt{1 + \frac{1}{m_i^2}}} \end{aligned}$$

Placing the centre of a circle along a normal vector only ensures that the distance between the circle and its closest boundary segment is minimal. It does not, however, ensure that the circle is placed in the interior of the region. For a circle of radius R with its centre at (x, y) and placed non-tangential to the boundary, the following condition guarantees that the circle will be in the interior of the region:[23]

$$d = \frac{\left|y - m_i x - b\right|}{\sqrt{1 + m_i^2}} \geq R$$

If a newly-packed circle with radius R is being placed tangential to both a boundary segment and an already-packed circle, whose centre is at (x_a, y_a) and has radius R_a, then the following condition guarantees that the newly-packed circle will remain in the interior of the region:[23]

$$d = \frac{\left|y_a - m_i x_a - b\right|}{\sqrt{1 + m_i^2}} \leq R_a + 2R$$

For each side of the trapezoid, the constraints for any circle translate to:

Side 1 (left non-parallel side): Slope $= m_1$

$$y \geq m_1 x \quad \text{if} \quad m_1 > 0$$
$$y \leq m_1 x \quad \text{if} \quad m_1 < 0$$
$$x \geq 0 \quad \text{as} \quad m_1 \to \pm\infty$$

Side 2 (parallel side lying on x-axis):

$$x^* = x_a \pm \sqrt{(R_a + y_a)(2R + R_a - y_a)}$$
$$y^* = R$$
$$\text{where} \quad y_a \leq 2R + R_a$$

Side 3 (right non-parallel side):

$$x^* = A - R$$
$$y^* = y_a \pm \sqrt{(R_a + R)^2 - (x_a - (A - R))^2}$$

Side 4 (other parallel side):

$$x^* = A - R$$
$$y^* = y_a \pm \sqrt{(R_a + R)^2 - (x_a - (A - R))^2}$$

Using these inequalities, it is possible to also compute the x and y coordinates of the newly-packed circle. Vrscay determined that, given the location of an already-packed circle centred at (x_a, y_a) the x coordinate of the newly-packed circle satisfies the equation:[23]

$$(1 + m_i^2)x^2 + 2(m_i c - m_i y_a + x_a)x + [{x_a}^2 + (c - y_a)^2 - (R + R_a)^2] = 0$$
$$\text{where} \quad c = b \pm R\sqrt{1 + m_i^2} \quad , \quad i = 1, 2$$

For the GGL scheme, it is important to place Position 1 and Position 2 prior to the placement of other circles. Similar to a rectangular region, Position 1 in a trapezoidal region is also located in the bottom-left corner; however, the coordinates of the centre of a circle in Positions 1 and 2 are not as intuitive as those of a rectangle. In fact, the coordinates will depend on the slope of the non-parallel side.

Every trapezoid has a pair of parallel sides connected by two non-parallel sides to form a four-sided figure. Suppose the non-parallel side on the left has slope m_1 and the non-parallel side on the right has slope m_2. The coordinates for the centre of a circle placed at Position 1 would be:

$$(x_1, y_1) = \left(\frac{R}{m_1} + R\sqrt{1 + \frac{1}{m_1}^2}, R \right)$$

As the slope of the non-parallel line approaches infinity, the angle between the non-parallel side and its immediately adjacent parallel side approaches $\frac{\pi}{2}$. Taking the limit of the x_1 coordinate, Vrscay re-obtained the x coordinate of the centre of a circle placed in Position 1 for a rectangular region.[23]

$$\begin{aligned}\lim_{m_1 \to \infty} x_1 &= \lim_{m_1 \to \infty} \frac{R}{m_1} + R\sqrt{1 + \frac{1}{m_1}^2} \\ &= 0 + R\sqrt{1 + 0^2} \\ &= R\end{aligned}$$

A circle placed at Position 2 would have its centre at:

$$(x_2, y_2) = \left(A + \frac{R}{m_2} - R\sqrt{1 + \frac{1}{m_2}^2}, R \right)$$

On the right side of the trapezoid, taking the limit of x_2 as m_2 goes to infinity gives the x_2 coordinate of the centre of a circle as the angle of intersection between the parallel and non-parallel sides approaches $\frac{\pi}{2}$. The limit is, as expected:

$$\begin{aligned}\lim_{m_1 \to \infty} x_2 &= \lim_{m_2 \to \infty} A + \frac{R}{m_2} - R\sqrt{1 + \frac{1}{m_1}^2} \\ &= A + 0 - R\sqrt{1 + 0^2} \\ &= A - R\end{aligned}$$

$$\begin{array}{lrl}
\text{Side 1:} & & x^* = R \\
& & y^* = y_a \pm \sqrt{(R_a + x_a)(2R + R_a - x_a)} \\
& \text{where} & x_a \leq 2R + R_a \\
\text{Side 2:} & & x^* = x_a \pm \sqrt{(R_a + y_a)(2R + R_a - y_a)} \\
& & y^* = R \\
& \text{where} & y_a \leq 2R + R_a \\
\text{Side 3:} & & x^* = A - R \\
& & y^* = y_a \pm \sqrt{(R_a + R)^2 - (x_a - (A - R))^2}
\end{array}$$

3.5.1 Results

In the figure below, two different sets of circles were packed into identical regions. The packing on the left resulted from packing 25 circles of the same size, each with a radius of 0.1 units. In the packing on the right, the radii were selected in such a way that the i^{th} circle in a set of 40 circles available to be packed, where $i = 1, \ldots, 40$, has a radius R_i such that:[23]

$$R_i = 0.15 - (i - 1) \cdot 0.003$$

The idea behind this scheme was to start with a small enough radius, namely 0.15 units, and decrease the size of the radius of subsequent circles in the availability list gradually enough to avoid having circles too small to significantly increase the fill percentage when packed. The fill percentage, using this set of 40 circles, was 0.8144; of the 40 circles available, only the circles with radii R_1,R_2,R_3,R_4,R_5,R_6,R_7,R_8,R_9,R_{10},R_{11},R_{12},R_{13},R_{14},R_{20},R_{23},R_{25},R_{28},R_{29},R_{30},R_{31},R_{35},R_{38},R_{39} and R_{40} were packed — 25 circles in all.[23]

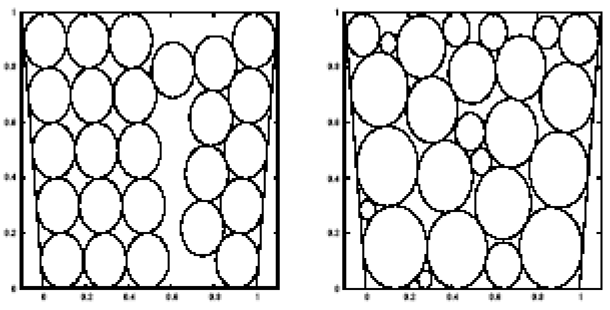


Figure 3.6: Packings of trapezoidal regions using circles of constant radii (left) and of radii $\frac{1}{2^{n/2}}$. Fill percentages are 0.7140 and 0.8144, respectively. Source:[23]

3.6 Constraints for L-shaped regions

The regions investigated thus far have all been convex. When a region has a locally non-convex subregion, then packing corners are not nearly as intuitive as they were for convex regions.

Definition 3.3. Let S be a rectangle whose corners are at (0,0), (a,0), (b,0), and (a,b). Let Q be a rectangle with corners whose coordinates are (c,0), (d,0), (d,b) and (c,d). Then (c,d) is coordinate location of the *internal corner* of an L-shaped region when Q is removed from S.

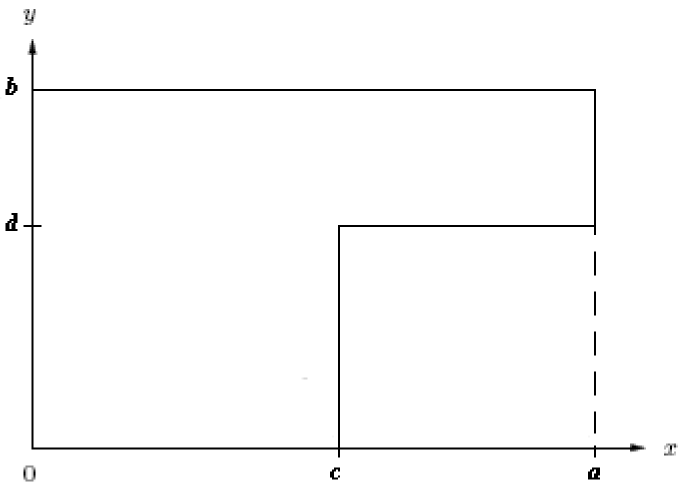


Figure 3.7: The internal corner of this L-shape is at (c,d).

Unlike convex quadrilateral regions such as the rectangle or trapezoid, which have two fundamental positions in each lower corner, the L-shape contains three.

Whereas convex regions have corners that circles can placed “lying down” in accordance with gravity, the biggest complication of the presence of an internal corner is that a circle can be packed while straddling the corner in an infinite number of ways. Furthermore, in terms of geometric constraints, it becomes much more difficult to ensure that the entire circle lies in the interior of the L-shape in a neighbourhood of the internal corner.

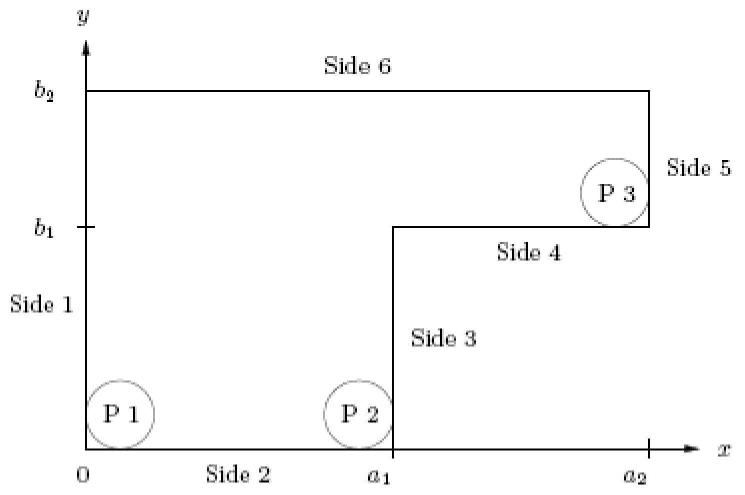


Figure 3.8: Positions 1,2 and 3 for an L-shaped region with an internal corner at (a_1,b_1), (A,0), (D,B) and (C,B). Source: [23]

The ideal position for the circle closest to the internal corner is to place it as close to the corner as possible, so as to minimize any space that cannot be otherwise occupied by another available circle. If the coordinates of the internal corner is defined to be (a_1, b_1) and a circle of radius R_a and centre at (x_a, y_a) is the circle closest to the corner, then the constraints on a newly packed circle of radius R and centre (x, y) are:

$$\begin{aligned} &(x - x_a)^2 + (y - y_a)^2 &&= (R + R_a)^2 \\ \text{and}\quad &(x - a_1)^2 + (y - b_1)^2 &&= R^2 \\ \text{where}\quad &\sqrt{(x_a - a_1)^2 + (y_a - b_1)^2} &&\leq R_a + 2R \end{aligned}$$

Regarding position numbers, given $(k-1)$ already-packed circles, Vrscay determined the number of positions available to the k^{th} circle to be f_k, where:[23]

$$f_k = \begin{cases} 1 & \text{if } k = 1 \\ \frac{1}{2}(k^2 + 7k - 2) & \text{if } k > 1 \end{cases}$$

3.6.1 Results

The radii of the circles were selected using an area-preserving bifurcation scheme. That is, given three sizes of circle radii r_1, r_2 and r_3, where

$$r_1 > r_2 > r_3$$

The ratio between r_i and r_{i+1}, where $i = 1, 2$, is such that the combined area of two circles of radius r_i have the same area as one circle of radius r_{i+1}. In the packing below, 140 circles were created for packing: 20 circles with a radius of 0.5 units, 40 circles with a radius of $\frac{1}{\sqrt{2}} \cdot 0.5 \approx 0.3536$ units, and 80 circles with a radius of $\frac{1}{2} \cdot 0.5 = 0.25$ units. A threshold limit was set at 0.7000 due to the fact that Vrscay's packing results for each of the previously mentioned quadrilateral regions had a fill percentage of at least 0.7000. [22] [23] Placing the internal corner at (2,2) and the external corner at (4,4), the packing below was achieved with a fill percentage of 0.7306. Of the original 140 circles available for packing, 3 circles with a radius of 0.5 units, 5 circles with a radius of approximately 0.3536 units, and 21 circles with a radius of 0.25 units were ultimately packed.

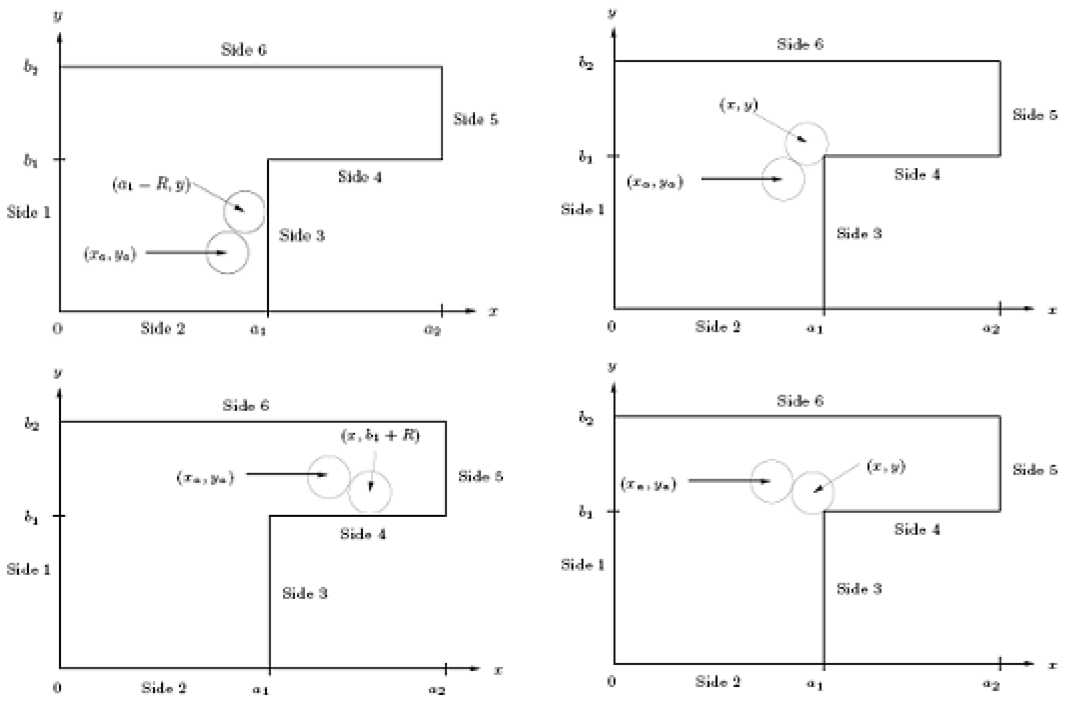


Figure 3.9: The four possible configurations of packing a circle along the internal corner: touching only Side 3 (top left), touching only Side 4 (bottom left), straddling the internal corner with another circle packed closer to Side 3 (top right), and straddling while packing another circle closer to Side 4. Source: [23]

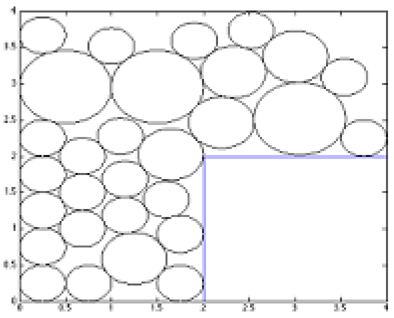


Figure 3.10: Circles with radii selected by an area-preserving bifurcation scheme packed into an L-shape. Fill percentage is 0.7036.

Chapter 4

The Anti-GGL Algorithm

4.1 Definitions

Definition 4.1. An *boundary vector* a vector that forms part of the boundary between the interior and exterior of a packing region. Circles are packed solely in the interior of a packing region and intersect a boundary vector at no more than one point.

Definition 4.2. A *packing corner* is the intersection of exactly two boundary vectors.

Definition 4.3. A packing corner is *valid* if it obeys constraints determined by the geometry of the packing region, and is *invalid* if it does not.

4.2 Motivation

The algorithm adapted from George, George, and Lamar was implemented originally for packing pipes in a rectangular container. For this reason, gravity would be a factor, and ensuring the lower two corners were occupied was an optimal strategy for maximizing amount of cross-sectional area filled by pipes. However, when circle packing has no real-world dependency on gravity, then a packing strategy posed by George, George, and Lamar was to place larger circles in the centre, and smaller circles around the perimeter.[4]

The anti-GGL algorithm takes this exact principle of packing larger circles around the centroid of the region, and placing smaller circles along the periphery. One slight difference between the GGL and anti-GGL algorithms is that the availability list for the anti-GGL algorithm sorts the radii of available circles in descending order in order to ensure that the greatest possible number of circles with the largest radius are packed first. Another difference between the two packing algorithms is that a greater emphasis is placed on using the two-circle algorithm than on position numbers.

4.3 The "two-circle" algorithm

The two-circle algorithm was first implemented in the GGL method introduced in the last chapter. Most circles in a packing will not be tangent to the boundary, so an algorithm was developed to take two circles that were closest to each other – tangential or not – and pack a new circle that would be tangential to both selected circles. The centre of this new circle would be computed by first requiring that the new circle be place tangentially to the other two. Once the circle is placed tangentially, the location of the centre would be computed using the cosine law for triangles.

Due to the nature of the two-circle algorithm, however, there may exist two possibilities for a new circle to be placed. When two possibilities exist, particularly after the initial placement of two circles at the centroid, the algorithm simply arbitrarily picks one of the two placement possibilities for the new circle and continues packing. Since the GGL algorithm focuses largely on packing upward from the lowest corners and the anti-GGL algorithm allows for packing in both upward and downward directions, the number of positions available to the k^{th} circle is twice the number of available positions in a GGL scheme, that is:[25]

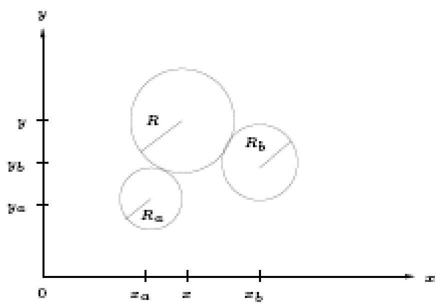


Figure 4.1: Example of the two-circle problem: packing a new circle with radius R given two smaller circles with radii R_a and R_b. Source: [22]

$$f(k) = k(k-1)$$

4.4 Packing algorithm

1. Take the two largest circles available to be packed, and place them such that they touch at exactly one point, and that point is the centroid of the region.
2. Use the two-circle algorithm to pack subsequent circles, subject to boundary constraints.

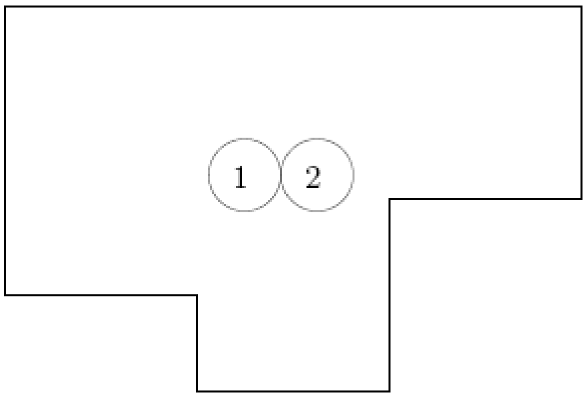


Figure 4.2: Placing the first two circles at the centroid using the anti-GGL scheme. Source: [25]

Jiang also suggested applying constraints pertaining to the distance, in radii, of a circle's centre to the boundary. If a packing were using two sizes of circles, with radii R and αR ($|\alpha| < 1$), then the distance d between a circle and the boundary must obey the following constraints:[11]

$$\begin{aligned} d \geq & \quad (2\alpha+1)R && \text{for circles of radius R} \\ d \geq & \quad (2\alpha+1)\alpha R && \text{for circles of radius } \alpha R \end{aligned}$$

4.5 Redesigning the packing region

The simplest method of coding a complex region is to find the coordinates of all valid packing corners, and construct straight lines between valid packing corners. However, when a region has a very large number of sides and may even include numerous locally non-convex subregions, then a region may be broken down piecewise as a collection of a very large finite number of boundary vectors.

In order to ensure that a circle, when packed, would remain in the interior of the region, Vrscay determined that, given two adjacent boundary vectors, a valid packing corner must obey the following constraint:[24]

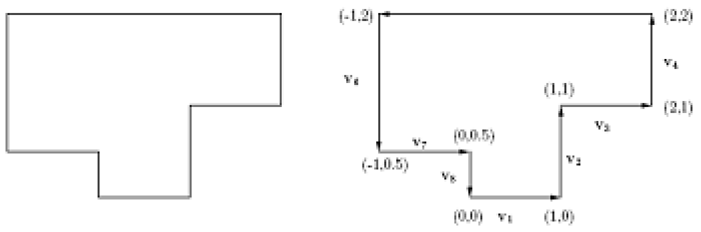


Figure 4.3: Example of the two-circle problem: packing a new circle with radius R given two smaller circles with radii R_a and R_b. Source: [22]

$$(v_{k-1,1}, v_{k-1,2}, 0) \times (v_{k,1}, v_{k-1,2}, 0) = (0, 0, A) \quad A > 0$$

where $v_{k,1}$ and $v_{k,2}$ are the x and y components of $\vec{v_k}$, respectively. (The same definitions apply to $v_{k-1,1}$ and $v_{k-1,2}$.) This constraint assumes that the two-dimensional region, when viewed in three dimensions, lies completely flat on the xy plane. That is, the value of z is always 0. Boundary vectors are also assumed to have at most one nonzero component. An additional assumption is that when either $v_{k,1}$, $v_{k,2}$,$v_{k-1,1}$, or $v_{k-1,2}$ are negative in value, it is because the boundary vector is pointing downward or towards the left, i.e. pointing towards negative values of x and y. When the cross product is calculated from the aforementioned two boundary vectors, a positive value for A results in a vector in the positive z direction, indicating that it is a valid packing corner.[24]

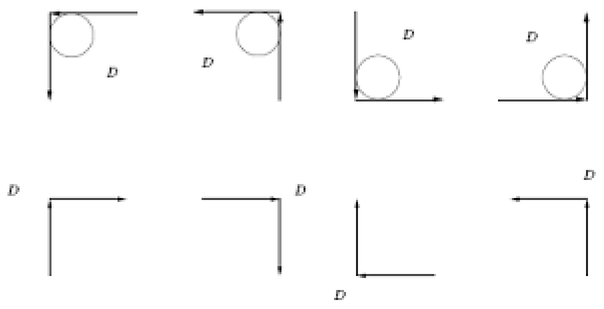


Figure 4.4: Examples of valid packing corners (top four) and invalid packing corners (bottom four), where D is the interior of the packing region. Source: [24]

Once valid packing corners are determined, then the centre of a newly packed circle with radius R will located at (x_i, y_i) where:[24]

$$x_i = p_k + s_k \cdot R, \quad y_i = p_k + s_k \cdot R \quad k = 1, \ldots, n_{pc}$$

In the equation above, the region is assumed to have exactly n_{pc} valid packing corners, and $p_k = (p_{k,1}, p_{k,2})$ is the coordinate of an arbitrarily chosen valid packing corner. $s_k = (s_{k,1}, s_{k,2})$ is called an "auxiliary index" whose components can only take the values 1 or -1.[24] The geometric significance of these auxiliary indices is that their numerical values determine the direction in which a circle must be packed relative to a packing corner. A component value of 1 indicates that the circle must be packed above or to the right of the packing corner; a component value of -1 indicates that the circle must be packed under or to the left of the packing corner. Depending on the values of $v_{k,1}$ and $v_{k,2}$, which are the components of boundary vector v_k where $k = 1, \ldots, n_{pc}$, the values of $s_k = (s_{k,1}, s_{k,2})$, the auxiliary indices, will be as follows:[24]

$v_{k,1}$	$v_{k,2}$	s_k	Location of packing circle (relative to packing corner)	Location of packing region
> 0	0	$(1,1)$	Right, up	Above v_k
< 0	0	$(-1,-1)$	Left, down	Below v_k
0	> 0	$(-1,1)$	Left, up	To the left of v_k
0	< 0	$(1,-1)$	Right, down	To the right of v_k

4.6 Results

Due to the placement of largest circles closest to the centroid, anti-GGL packings tend to bias larger circles towards the centre and smaller circles closer to the boundary, as predicted. Moreover, when "narrow strip" subregions are present in a packing region, filling the middle of the region with the entire collection of larger circles results in more circles with smaller radii being available to pack the aforementioned narrow strip sections. In the figure below, the availability lists do not end up being the same, since the purpose of the anti-GGL scheme on the right was to maximize the number of largest circles packed. As a result, the availability list for the anti-GGL packing will be mostly populated by smaller circles.

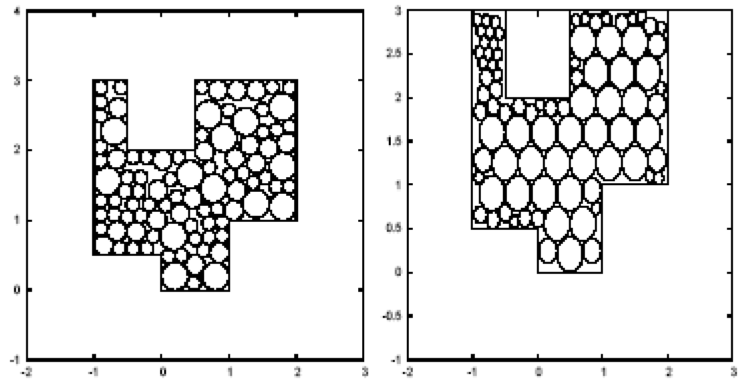


Figure 4.5: A comparison of packing a region with locally non-convex subregions using the GGL scheme (left) and the anti-GGL scheme (right). Fill percentages are 0.7685 and 0.7878, respectively. 10 iterations and 100 iterations were used, respectively. Source: [24],[25]

Chapter 5

Improving the Anti-GGL Algorithm

One of the problems with both the GGL and anti-GGL algorithms is that it does not allow for solutions to be optimized by rolling out or jiggling circles to fill previously unpacked spaces in the region. There a number of real-life situations that confirm the success of this approach. Given a box of balls, the best way to fit more balls in the box is to shake the box around and continue packing in the newly-created space. Similarly, if clothes are being packed in a suitcase, the best way to pack more clothes would be to move them around in the suitcase.

5.1 Particle dynamics

Treating the packing problem as a set of particles colliding with each other is one way to model the jiggling process. Graham *et al.* suggested using a billiards model that treated packing circles as collisions of billiards inside a container. [5] The idea was to let "disks experience elastic collsions at impacts beginning with a randomly chosen initial configuration."[5] Suppose S is a set of n points floating around freely inside a circular region of radius 1. If each point becomes the centre of a circle with diameter d, then the amount of space remaining in the region decreases as d increases. The common diameter of the circles is increased until a local maximum, i.e. a "steady state", is reached.[5]

Szabó *et al.* proposed a similar algorithm involving pulsating disks. The Pulsating Disk Shaking (PDS) algorithm resembles the billiards method in that it involves shaking and rattling identical circles inside the region. The common radius of the circles is increased and decreased repeatedly until a non-overlapping packing is achieved. To save computational time, distances are classified as "near" (contacting) or "far" (non-contacting).[21]

Kettlewell constructed the following optimization problem as a variant of the billiards model and the PDS algorithm:

$$\begin{array}{lll}
\max & N & \\
\text{subject to} & \displaystyle\sum_{\substack{j=1\\ j\neq i}}^{N} F_{ji} + \sum_{\substack{k=1\\ k\neq i}}^{M} n_{ki} = m_i \frac{dv_i}{dt} & v_i(0) = v_0 \\
& \left\| c_j - c_i \right\| \geq r_i + r_j & \forall i \neq j \\
& F_{ji} = \dfrac{k m_i m_j}{\left\| c_i - c_j \right\|^2} & \\
& n_{ki} = \begin{cases} \frac{|n_{ki}|(c_i - cj)}{\|c_i - c_j\|} & \text{if } \|c_i - cj\| = r_i + r_j \\ 0 & \text{otherwise} \end{cases} &
\end{array}$$

In this problem, F_{ji} is defined as the net force of circle j on circle i, whereas n_{ki} is the normal force of circle k on circle i. The norm $\left\| c_j - c_i \right\|$ is the Euclidean distance between the centres of circle i and circle j.

One of the major complications of this nonlinear program was the amount of computational time required to find a locally optimal solution. Since the constraints contain a system of differential equations with linear inequalities, Kettlewell estimated the computational time to be implausibly high to find even one packing solution.

Using Rhino, a variant of the Grasshopper program, Qiao created a similar model to simulate the jiggling of circles inside a region. At the same time, in keeping with the anti-GGL scheme, packings where larger circles were placed closest to the centroid were preferred. During each iteration, circles "jiggled" by colliding with each other and ultimately "gravitated" towards a single point in the middle that acted like a "black hole". This gravitation managed to reduce the amount of space between already-packed circles, and created space for packing additional circles. Unfortunately, due to the randomness of collisions, Rhino was unable to ensure that only the largest circles would end up closest to the "black hole". Furthermore, based solely on our experience with Rhino, there did not seem to be a built-in method for ensuring that circles remained in the interior of a pre-defined bounded region.

5.2 "One-circle" algorithm

Packing an arbitrary region starting from the centroid, the anti-GGL method biases large circles towards the middle and fills the central area quite nicely. However, the fact that the two-circle algorithm relies on the positions of two previously-packed circles makes it increasingly likely for corner regions and boundary strips to remain entirely unpacked.

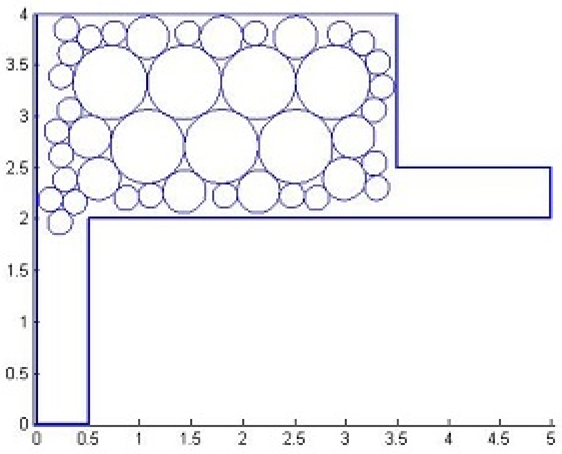


Figure 5.1: Example of an unpacked boundary strip. Source: [13]

To correct this, Jiang packed circles into the region using the anti-GGL method until the packing approached an ϵ distance of the boundary. Once this threshold was reached, the algorithm would switch from the anti-GGL method to the One-Circle Algorithm.[10]

Jiang's One-Circle Algorithm is intended to target the packing of corner regions.[10] The algorithm takes a circle located just beyond the ϵ threshold of the boundary and generates up to n additional circles, where g is the desired degree of the initial circle. For the problem of packing identical circles, $g = 6$ due to the optimal denseness of the hexagonal lattice packing. For unequal circles, however, the value of g is less straightforward. Suppose $C = \left\{c_{initial}, c_2, c_3, ..., c_g, ..., c_n\right\}$ is the set of the centres of all n circles in a packing. If a circle with centre c_i has radius r_i for all $i = 1, 2, ..., n$, then g is the solution to the following optimization problem:

$$\begin{array}{ll} \max & g \\ \text{subject to} & g \leq n \\ & \|c_i - c_{i+1}\| = \min\ \|c_i - c_j\| \qquad\qquad \forall c_i, c_j \in C \setminus \{c_{initial}\} \\ & \min \left\{ \sum_{i=2}^{g} \|c_{initial} - c_i\| \right\} = r_{initial} \cdot g + \sum_{i=2}^{g} r_i \\ & \sum_{i=2}^{g-1} \cos^{-1}\left(\frac{(c_i - c_{initial}) \cdot (c_{i+1} - c_{initial})}{\|c_i - c_{initial}\| \|c_{i+1} - c_{initial}\|} \right) \leq 2\pi \end{array}$$

The first constraint ensures that circles with consecutively numbered indices will be a minimum distance from each other. By doing so, the algorithm will be more likely to count the circles placed directly around a particular already-placed circle before applying the one-circle algorithm. Furthermore, this constraint ensures that the "last" circle, placed tangential to the initial circle, will be adjacent to the "first" circle placed. The second constraint ensures that the circles remain tangential to the initial circle, but may not necessarily end up being tangential to each other. The third constraint prevents overlap of circles by placing a restriction on the angle sum. $(c_i - c_{initial})$ is the vector created by subtracting the coordinates of the centre of the initial circle from those of another arbitrarily chosen circle. Taking the vectors for two circles with consecutively-labelled indices, the dot product of these two vectors can be exploited to reveal the cosine of the angle between them.

Once g is determined, g circles are generated tangential to the initial circle. In conjunction with the anti-GGL method, the onecircle algorithm tries to pack the largest circle available before moving on to smaller circles. If we assume that circle j has already been packed, the possible position numbers for circle i no longer depend on the number of sides in the region but rather the desired degree of tangency, g, for circle i.

Lemma 5.1. Suppose $f(i) = g + (i-1)(g+i)$, where g is the desired degree calculated for the onecircle algorithm. Then, for $i \geq 2$ the number of positions available to circle i is equal to $f(i-1)$.

Proof. By induction. When $i = 2$, it means that there is only one circle in the packing at the moment. The value for g produced by the onecircle algorithm determines the number of circles to be generated, which is essentially the same as generating the number of available positions for the next circle. Now suppose that the number of positions available to the n^{th} circle in a packing is $f(n-1) = g + (n-1)(g+n)$. The $(n+1)^{th}$ circle can be placed using either of two methods: (1) any of g positions tangential to a previously packed circle, or (2) tangential to any two previously packed circles.

For method (1), there would be $n \cdot g$ position numbers. For method (2), since being tangential to circle 1 and 2 is the same as being tangential to circle 2 and 1, there would be $\frac{n(n-1)}{2}$ ways to choose any two previously packed circles. Then, for each pair of previously packed circles, there would be two ways to place a new circle tangentially to both of them, leaving us with $n(n-1)$ total positions for method (2).

Adding the number of possible position numbers for methods (1) and (2), we get:

$$\begin{array}{rl} & ng + n(n-1) \\ = & g + (n-1)g + n(n-1) \\ = & g + (n-1)(g+n) \\ = & f(n) \end{array}$$

$\square$

The table below shows the position numbers for the fifth circle of a packing. [10]

	$s=1$	...	$s=g$	$j=1$	$j=1$	$j=2$	$j=2$	$j=3$	$j=3$	...
$i=1$	1	...	g							
$i=2$	g+1	...	2g	2g+1	2g+2					
$i=3$	2g+3	...	3g+2	...	...	...	3g+6			
$i=4$	...	...	...	...	...	...	...	...	4g+12	

5.2.1 Results

Jiang's results for a triangular region are as follows:

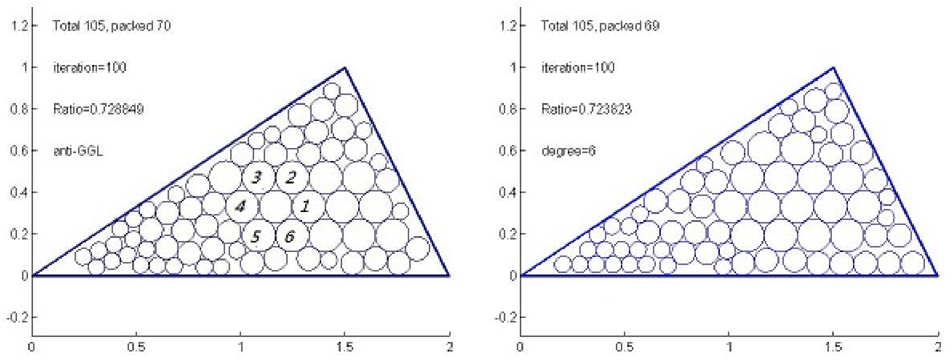


Figure 5.2: A comparison of packing circles solely using the anti-GGL scheme (left) and using the one-circle algorithm (right). Source: [10]

The degree was set by Jiang to 6 in order to reflect "penny packing", which is the optimal packing of circles of constant radii in any region. Using the one-circle algorithm, the fill percentage worsened from 0.7238 to 0.7288, but this result can be attributed to the convex nature of the triangle. For regions with locally non-convex subregions, the one-circle algorithm is effective in packing regions otherwise impossible to pack using the anti-GGL scheme.

5.3 Removal of circles on the outer layer

According to Qiao and Jiang, packings generated by the anti-GGL algorithm could be improved by removing circles within a certain distance of a region's boundary and repacking the newly empty space with smaller circles.[12][18]

In order to implement the anti-GGL algorithm at all, a region must first be defined as a union of several boundary vectors $\vec{v_i}$ for $i = 1, 2, ..., n$. That is, we can define a boundary D as:

$$D = \bigcup_{i=1}^{n} \vec{v_i}$$

The boundary vectors were then converted into a union of boundary points, with $p_{0,i}$ being a point on the i^{th} boundary vector acting as a point of reference. That is, for an arbitrary boundary point p on an arbitrary boundary vector $\vec{v_i}$,

$$p = p_{0,i} + t(\vec{v_i}) \quad \text{for some} \quad t \in \mathbb{R}$$

Kettlewell and Qiao exploited this vector property of the boundary and computed a normal vector $\vec{n_i}$ for each boundary vector. By travelling an arbitrarily small distance ϵ along the direction of each normal vector, and taking the inward normal vector to be arbitrarily positive, Qiao turned the boundary D into a new boundary $\tilde{D}$ by using the following transformation on an arbitrary boundary point p:

$$
\begin{aligned}
& & \tilde{D} = \bigcup \tilde{p} \\
&\text{where} & \tilde{p} = p_{0,i} + t(\vec{v}_i) + \epsilon(\vec{n}_i) \quad \forall i = 1, 2, ..., n \\
&\text{for some} & t, \epsilon \in \mathbb{R}
\end{aligned}
$$

Jiang also found a method to improve packings by removing the outermost layer of circles. To determine an outermost layer, Jiang used an adjacency matrix to attribute a layer number to each circle. When the outermost layer of circles was removed from a packing, a completely separate system of position numbers was needed, the table for which is shown below:[12]

	$s = 1$	...	$s = n_s$	$j = 1$	$j = 2$	$j = 3$	...
$i = 1$							
$i = 2$			$n_{pc} + 1$				
$i = 3$			$n_{pc} + 2$	$n_{pc} + 3$			
$i = 4$			$n_{pc} + 4$	$n_{pc} + 5$	$n_{pc} + 6$		
...	...	...	...	...	...	...	...
$i = n_{L2}$			$\frac{(n_{L2}-1)(n_{L2}-2)}{n_{pc}+3}$	...	...	...	...
$i = n_{L2} + 1$	$M_1 + 1$	...	$M_1 + n_s$	$M_1 + n_s + 1$	...	...	...

	$j = n_{L2} - 1$	$j = n_{L2}$	$j = n_{L2} + 1$	$j = n_{L2} + 2$
$i = n_{L2}$	M_1			
$i = n_{L2} + 1$	...	$M_1 + n_s + n_{L2}$		
$i = n_{L2} + 2$	...	...	$M_1 + 2n_s + 2n_{L2} + 1$	
$i = n_{L2} + 3$	...	...	...	$M_1 + 3n_s + 3n_{L2} + 3$

where n_s is the number of sides in the packing region, n_{pc} is the number of packing corners in the region, n_{L2} is the number of circles in the exposed layer once the outermost layer of packed circles was removed, and where

$$M_1 = \frac{(n_{L2} - 1)(n_{L2} - 2)}{n_{pc} + n_{L2} + 1}$$

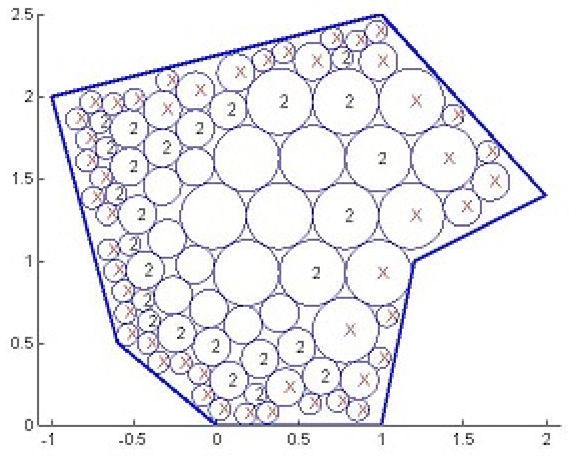


Figure 5.3: Visualization of using an adjacency matrix to label layers of packed circles. Source: [12]

After removing the outermost layer of circles, an additional hybrid packing procedure was also suggested by implementing the anti-GGL scheme and subsequently using either or both of the one-circle and two-circle algorithms to pack boundary strips or areas closer to the boundary. The net effect of using this hybrid scheme was to simulate the act of jiggling a container to create additional free space to pack more circles; removing outermost circles and re-packing them using the aforementioned hybrid scheme gave the illusion that the region was being “shaken” in order to create more packing space for placing more circles and thus improving the fill percentage.

5.3.1 Results

In the polygonal region below, Qiao decreased the total area of the region by 10% while maintaining the aspect ratio of the original polygon. In the figure below, the outer boundary represents the shape of the original region, and the inner boundary represents the shape of the newly shrunk region.

By making the polygonal region smaller, circles that could originally be packed along the newly-created inner boundary remained unpacked. The two-circle algorithm, which would have placed circles in the space now occupied by the inner boundary, has then become constrained by the new boundary and is unable to pack a circle in the interior of the polygon bounded by the inner boundary. This approach forces circles to be re-packed into a smaller space; once circles are re-packed successfully, the inner boundary can then be removed to pack additional circles in hopes of optimizing the fill percentage. If the fill percentage is not improved, a new "inner boundary" can be created once again to force the re-packing of circles.

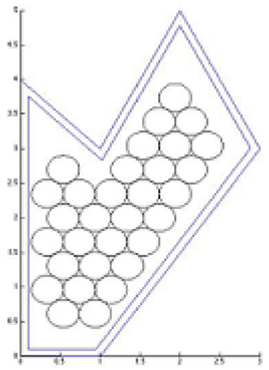


Figure 5.4: Result of applying Qiao and Kettlewell's boundary transformation to shrink a region. Source: [18]

In the triangular region below, Jiang removed all circles with the smallest radius. Using the one-circle algorithm, he re-packed the circles of the smallest radius; when the optimal degree of each non-smallest circle was calculated, that number happened to be 6. That is, given an already-packed non-smallest circle in the region, new circles would be packed around it until that particular already-packed non-smallest circle was tangent to exactly six circles.[12] The significance of being tangent to six circles lies in the fact that the "penny packing", or hexagonal packing, is the optimal method of packing circles of an identical radius.[20] Although this region is packed with circles of three different radii, the hexagonal packing principle, given its benefits, was still applied as a packing heuristic. However, if the ratio between the size of the radii of the circles being packed increases drastically, the hexagonal heuristic certainly cannot be guaranteed to function as well as it did in the example below. If the ratio between radii increases significantly, the optimal degree will be much larger than 6, as it will be possible to pack many more circles with much smaller radii tangentially around one that is much larger.

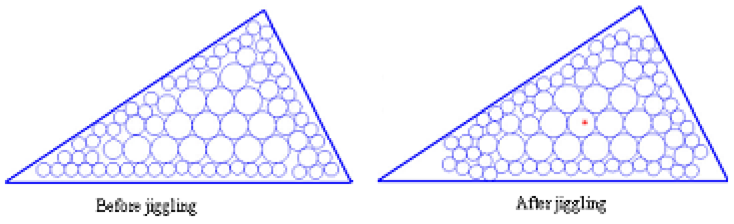


Figure 5.5: Effects of "jiggling" using a combination of layer removal, the one-circle algorithm, and the two-circle algorithm, as suggested by Jiang. Source: [14]

Chapter 6

Conclusion

Circle packing has many real world applications. Although hexagonal lattice arrangements are ideal, geometric constraints often demand for packings to take a different shape. Researchers have devised numerous packing algorithms with hopes of maximizing space either through jiggling, rotating, re-packing or random assortment, but the algorithm devised by George, George, and Lamar was ultimately chosen for this report due to its position and side numbering system, which allowed for circles to be easily tracked and labelled while being packed. For non-rectangular quadrilateral regions, numbering positions and sides became increasingly complex, but not impossible; once sides and positions were numbered and geometric constraints identified, circles were packed using the algorithm as before. When the region became that of a circle, the boundary of the region was approximated using smaller boundary vectors in order to maintain use of the side and position numbering scheme. Packing corners were no longer defined as the bottom two corners, as was the case with a quadrilateral, but by the intersection of two boundary vectors.

The GGL algorithm was particularly useful in working towards a solution for fitting as many pipes into a rectangular container as possible, a problem that relied heavily on gravity. A different direction was explored in which gravity did not play a factor, which led to the development of the "anti-GGL" packing algorithm. The purpose of the algorithm is to maximize fill percentage while packing as many larger circles towards the centroid of the region as possible. For regions with narrow strips or tight corners, the one-circle algorithm, which was created to supplement the anti-GGL algorithm, filled these tight sections by removing the outermost layer of circles and re-packing those removed circles into the packing region. The one-circle algorithm provided the illusion of jiggling circles in the packing in order to minimize the amount of unpacked space in the region. Additional layer-removing and jiggling approaches were explored using particle dynamics, region-shrinking techniques, and various software packages.

6.1 Final remarks

Jiggling remains the greatest obstacle to optimizing the fill percentage in a region. In the real world, jiggling is fairly intuitive as it only involves shaking around a container before additional packing space is finally created. However, in terms of creating a computerized algorithm that simulates this action, the best approach thus far has been to remove an outer layer of packing circles before re-packing them into the region. A path similar to that of Brownian motion could hypothetically be generated for a particular circle in a packing to simulate jiggling, but as seen by the system of differential equations and inequalities present in the particle dynamics approach to the problem, the complexity of the problem indicates that it may not be easily solved to produce a solution that both simulates jiggling and creates a maximum amount of additional space by means of this jiggling effect.

Bibliography

[1] Birgin, E.G., Gentil, J.M. New and improved results for packing identical unitary radius circles with triangles, rectangles, and strips. *Computers and Operations Research* 37 (2010): 1318-1327.

[2] Birgin, E.G., Martinez, J.M., Ronconi, D.P. Optimizing the packing of cylinders into a rectangular container: A nonlinear approach. *European Journal of Operational Research* 160 (2005): 19-33.

[3] Chang, H.C., Wang, L.C. A simple proof of Thue's theorem on circle packing (2010): 1-4.

[4] George, J.A., George, J.M., Lamar, B.W. Packing different-sized circles into a rectangular container. *European Journal of Operational Research* 84 (1995): 693-712.

[5] Graham, R.L., Lubachevsky, B.D., Nurmela, K.J., Ostergard, P.R.J. Dense packings of congruent circles in a circle. *Discrete Mathematics* 181 (1998): 139-154.

[6] Hifi, M., M'Hallah, R. A dynamic adaptive local search algorithm for the circular packing problem. *European Journal of Operational Research* 183 (2007): 1280-1294.

[7] Hifi, M., M'Hallah, R. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research* (2009): 1-22.

[8] Hifi, M., M'Hallah, R. Approximate algorithms for constrained circular cutting problems. *Computers and Operations Research* 31 (2004): 675-694.

[9] Huang, W.Q.; Li, Y.; Li, C.M.; Xu, R.C. New heuristics for packing unequal circles into a circular container. *Computers and Operations Research* 33 (2006): 2125-2142.

[10] Jiang, W.Z. Anti-GGL 2.0. June 17, 2014.

[11] Jiang, W.Z. Anti-GGL 2.1. June 26, 2014.

[12] Jiang, W.Z. Anti-GGL 3.0. June 26, 2014.

[13] Jiang, W.Z. Progress report. May 19, 2014.

[14] Jiang, W.Z. Progress report. April 21, 2015.

[15] Lopez, C.O., Beasley, J.E. A heuristic for the circle packing problem with a variety of containers. *European Journal of Operational Research* 214 (2011): 512-525.

[16] Lü, Z.P., Huang, W.Q. PERM for solving circle packing problem. *Computers and Operations Research* 35 (2008): 1742-1755.

[17] Nurmela, K.J., Ostergard, P.R.J. Packing up to 50 equal circles in a square. *Discrete Computational Geometry* 18 (1997): 111-120.

[18] Qiao, T. Report. August 20, 2014.

[19] Stephenson, K. Circle packing: a mathematical tale. *Notices of the AMS* 50 (2003): 1376-1388.

[20] Stephenson, K. *Introduction to Circle Packing: The Theory of Discrete Analytic Functions*. Cambridge University Press. 2005.

[21] Szabó, P.G., Markót, M.C., Csendes, T., Specht, E., Casado, L.G., García, I. *New Approaches to Circle Packing in a Square With Program Codes*. Springer: New York. 2007.

[22] Vrscay, E.R. Some notes on the George-George-Lamar (GGL) formalism to solve circle-packing problems with some preliminary explorations. January 28, 2014.

[23] Vrscay, E.R. Some notes on the George-George-Lamar (GGL) formalism to solve circle-packing problems with some preliminary explorations - Chapter 2. February 19, 2014.

[24] Vrscay, E.R. Some notes on the George-George-Lamar (GGL) formalism to solve circle-packing problems with some preliminary explorations - Chapter 3. March 28, 2014.

[25] Vrscay, E.R. Some notes on an "anti-George-George-Lamar" ("anti-GGL") formalism to solve circle-packing problems - Chapter 4. March 28, 2014.

[26] Wang, H.Q., Huang, W.Q., Zhang, Q., Xu, D.M. An improved algorithm for the packing of unequal circles within a larger circle. *European Journal of Operational Research* 141 (2002) 440-453.