Circle Packing in Euclidean and Hyperbolic Geometries

Mary E. Wilkerson

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mathematics

William J. Floyd, Chair
James E. Thomson
Peter E. Haskell

April 29, 2008
Blacksburg, Virginia

Keywords: Circle Packings, Uniform Neighbor Model

Circle Packing in Euclidean and Hyperbolic Geometries

Mary E. Wilkerson

(ABSTRACT)

Given a graph that defines a triangulation of a simply connected surface, it is possible to associate a radius with each vertex so that the vertices represent centers of circles, and the edges denote patterns of tangency. Such a configuration of circles is called a *circle packing*. We shall give evidence for the existence and uniqueness of circle packings generated by such graphs, as well as an explanation of the algorithms used to find and output a circle packing on the complex plane $\mathbb{C}$ and hyperbolic disc $\mathbb{D}$.

Cassie, Mom, and Dad–this is for you.

Acknowledgments

I would like to thank my adviser, Dr. William Floyd, for his guidance, support, and seemingly endless patience over this past year. I would also like to express gratitude to the rest of my thesis committee, Dr. Peter Haskell and Dr. James Thomson, for their time and effort in reviewing my work.

Last but not least, I owe many thanks to my family, friends, and Mike for giving their love and encouragement throughout this past year.

Without all of these wonderful people, this document may never have come to fruition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is often said that a picture is worth a thousand words. Rarely however, is this as true as it is for the field of circle packings!

Circle packings are configurations of circles with a specified pattern of tangencies. Studied by E. M. Andreev and Paul Koebe, circle packings went long unnoticed until William Thurston reintroduced them in a talk a little over 20 years ago. With the aid of many researchers, Thurston's conjecture that circle packings could potentially be used to approximate conformal maps has since grown into discrete analytic function theory–having a multitude of parallels to topics in complex analysis. Circle packings have since also found applications in the study of tilings, graph embedding, and in creating mappings of brain structures.

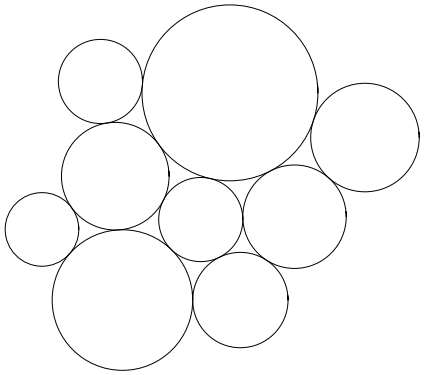

Figure 1.1: A sample packing

Circle packings no doubt are a fantastic tool for displaying mathematics as it happens. From subtle nuances of conformal maps to the networks contained within the cortical surface, these structures often serve as a useful visual aid. In such applications however, the ability to produce reliable output is critical. The basic ideas involved in creating a packing appear simple: fix boundary conditions, solve for the radii of the remaining circles in the packing,

and implement a placement procedure based on the geometry that the packing lives in. This method of solution may be perfectly fine for small systems. In modeling networks containing several hundred thousand nodes however, the system of equations involved in attempting a direct solution is computationally intensive. For this reason, circle packings are generally sought via iterative means.

In this paper, we examine "Acceleration" and the "Uniform Neighbor Model," strategies proposed by Collins and Stephenson in their paper *A Circle Packing Algorithm* for determining the radii of circles in a given packing. We build up to finding a circle packing with these methods by answering the following questions:

- What exactly is a Circle Packing?
- What criteria will allow the existence of these structures?
- What are the steps involved in implementing Uniform Neighbor model and Acceleration?
- Why do these methods work?
- Once a solution for radii is found, how is a circle packing actually laid out?

Upon developing the ability to create circle packings, we test the algorithms on several graphs determined by finite subdivision rules, noting effectiveness and possible alterations and improvements.

Chapter 2

Geometries

Before discussion of the structures that we are to create, it is necessary to address relevant properties of the surfaces that they will be constructed on. Although the notion of a circle (thus a circle packing) can be presented on a general manifold of constant curvature, we shall restrict to surfaces with constant curvature, focusing on Euclidean and Hyperbolic spaces and their respective geometries.

2.1 Euclidean Geometry

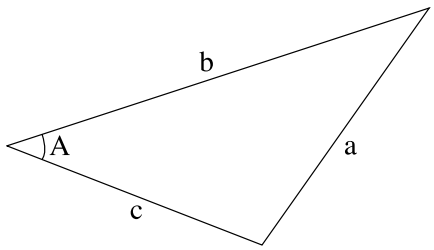

Figure 2.1: A triangle in Euclidean space

Euclidean space is of constant curvature 0. This is the first geometry that most students of mathematics encounter, and much of its workings are familiar if not routine: geodesics are "straight lines," and the distance metric and trigonometry of this space follow as expected. As it is referenced heavily in the Euclidean computations in this paper, we recall from elementary trigonometry the law of cosines: If we have a, b and c the side lengths of some triangle, and A the angle opposite the side of length a then equation 2.1 holds.

$$A = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) \tag{2.1}$$

This is tremendously useful in calculating the angles in any triangle whose vertices are the centers of three mutually tangent circles. Consider the geodesics that pass through these circle centers–they pass right through the point of tangency. Thus, the side lengths of the triangle are determined by the radii of the circles–as in figure 2.2. A few simple substitutions in the law of cosines yields formula 2.2, and with a bit of manipulation we may obtain from this equation 2.3. As the connection is not readily visible, but the mathematics involved is nothing more than routine algebra and trigonometry, we provide justification for equation 2.3 in Appendix A.

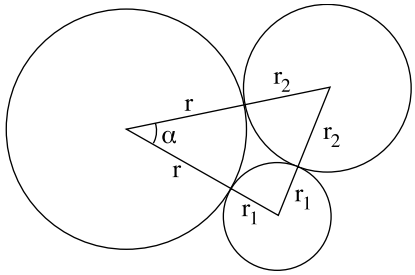

Figure 2.2: Reference for law of cosines with radii

$$\alpha = \arccos\left[\frac{(r+r_1)^2+(r+r_2)^2-(r_1+r_2)^2}{2(r+r_1)(r+r_2)}\right] = \arccos\left[1-\frac{2r_1r_2}{(r+r_1)(r+r_2)}\right] \tag{2.2}$$

$$\alpha = 2\arcsin\sqrt{\frac{r_1r_2}{(r+r_1)(r+r_2)}} \tag{2.3}$$

Finally, we have Herron's formula for the area of a triangle as based on side length. If s is the semi-perimeter of a triangle T with sides a, b, and c, then the area of T is given by:

$$Area(T) = \sqrt{s(s-a)(s-b)(s-c)}, \tag{2.4}$$

which in our mutually tangent trio of circles setting becomes:

$$Area(\triangle vv_1v_2) = \sqrt{rr_1r_2(r+r_1+r_2)} \tag{2.5}$$

These angle and area calculations yield a simple yet important result for circle packing, as given by Theorem 2.1.

Theorem 2.1 (Monotonicity I)**.** *Let v, v_1, and v_2 be the centers of three mutually tangent circles, with r, r_1, and r_3 their respective radii. The triangle formed by connecting these centers has an angle at v which is monotonically decreasing in r and angles at v_1 and v_2 which are monotonically increasing in r. Further, the area of the triangle is monotonically increasing in r.*

Proof. We give a sketch of the proof suggested by [6] and [7].

Taking partial derivatives of equation 2.2 with respect to r yields a negative result. Thus, the angle at v is strictly monotonically decreasing in r. Relabeling the indices to reflect the proper angle and taking partials with respect to r_1 or r_2 yields a positive result. Thus, the angles at v_1 and v_2 are strictly monotonically increasing in r.

Finally, taking the partial with respect to r as in equation 2.5 yields a positive result, which implies the area of the triangle is strictly monotonically increasing with respect to r. □

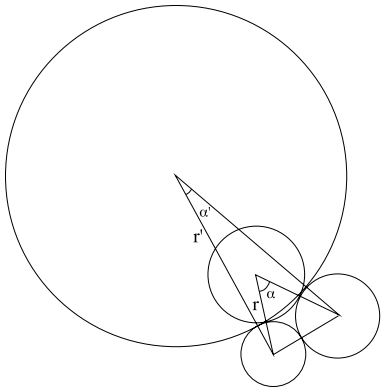

Figure 2.3: Monotonicity I in action.

As in the above figure, the effects of Monotonicity I are fairly easy to envision. We may extend the consequences of Monotonicity I to yield the following corollary.

Corollary 2.2 (Monotonicity II)**.** *Let v, v_1, and v_2 be the centers of three mutually tangent circles, with r the radius at v. The angle at $v, \theta(v)$ is a continuous function of r. We have that $\lim_{r\to\infty} \theta(v) = 0$ and $\lim_{r\to 0} \theta(v) = \pi$. Thus, given a value α in $(0, \pi)$, there exists some unique value of r which makes $\theta(v) = \alpha$.*

Proof. Again, we sketch proofs given by [6] and [7].

The formula for $\theta(v)$ is given by equation 2.3, a continuous function. The limit calculations are very straightforward. It is the last statement in the corollary which requires a small amount of work.

Given α in $(0, \pi)$, since $\lim_{r\to 0} \theta(v) = \pi$, there exists some value r_1 that makes $\theta(v) > \alpha$. Similarly, since $\lim_{r\to\infty} \theta(v) = 0$, there exists a value r_2 which makes $\theta(v) < \alpha$. That there exists some r which allows $\theta(v) = \alpha$ may be shown by applying the Intermediate Value Theorem to $[r_1, r_2]$ in $(0, \infty)$. As Monotonicity I shows that $\theta(v)$ is strictly monotonically decreasing in r, the value this determines for r must be unique. □

Finally, we note that isometries in Euclidean space are generally given as compositions of translations, rotations, reflections, and glide reflections. However, for the purposes of circle packings, the only isometries are compositions of translations and rotations as we wish to preserve the orientation of our packings. Isometries of circle packings on the complex plane are given by

$$f(z) = e^{i\theta}(z - z_0), \tag{2.6}$$

where z_0 is sent to the origin, and θ identifies an angle of (counterclockwise) rotation. Rewriting $e^{i\theta}$ using Euler's formula and expanding via complex multiplication yields a result identical to utilizing the traditional rotation matrix on a translation in $\mathbb{R}^2$. We may identify Euclidean space with either $\mathbb{R}^2$ or $\mathbb{C}$, but shall refer mostly to the latter.

2.2 Hyperbolic Geometry

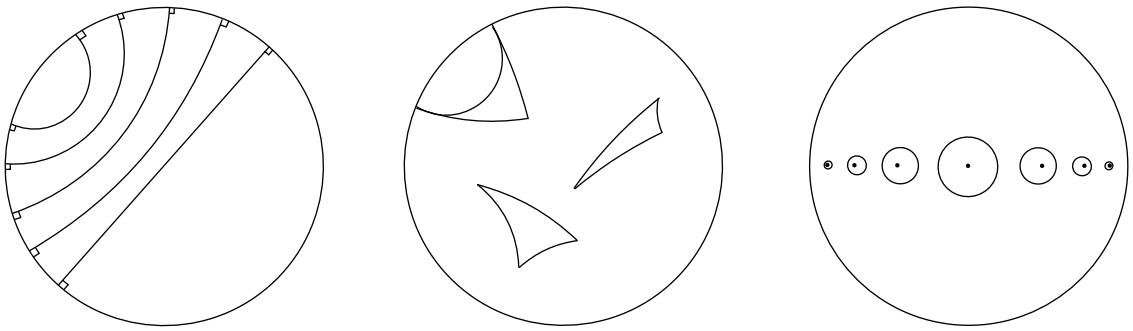

Figure 2.4: Geodesics, triangles, and circles on the hyperbolic disk $\mathbb{D}$

Hyperbolic space is of constant curvature -1. We shall work on the unit disc, $\mathbb{D}$. Here, geodesics are arcs of circles which are perpendicular to the boundary of the unit disc and Euclidean segments which go through the origin. (We may think of these geodesics passing through the origin as arcs of circles with infinite Euclidean radii.) We use the same notation for points in the disc as we do for points on the complex plane. All formulas that follow are as given in [1].

The metric on the hyperbolic disc as given by Poincaré is:

$$\rho(z, w) = \log \frac{|1 - z\bar{w}| + |z - w|}{|1 - z\bar{w}| - |z - w|} \tag{2.7}$$

Note that this gives the distance between a point w and the origin as $\log \frac{1+|w|}{1-|w|}$. Thus, any point on the boundary of the unit disc may be considered infinitely far away from the origin.

By the triangle inequality, we may extend this to say that points on the boundary are infinitely far away from any other point in $\mathbb{D}$.

Due to the new metric, trigonometric operations and properties of triangles differ considerably from those of Euclidean space. Note in Figure 2.4 that angles do not necessarily sum to π in every triangle. In fact, the sum of angles of a triangle in hyperbolic space determine the area of that triangle! If T is some triangle with angles α, β, and γ, then we have that the area of T is given by equation 2.8.

$$Area(T) = \pi - \alpha - \beta - \gamma \tag{2.8}$$

Another consequence is that we work with a significantly different law of cosines: If we have a, b and c finite side lengths of some triangle, and A the angle opposite the side of length a as in Figure 2.1, then equation 2.9 holds.

$$A = \arccos \frac{\cosh b \cosh c - \cosh a}{\sinh b \sinh c} \tag{2.9}$$

Circles in this space appear as circles in Euclidean space, although hyperbolic circles generally have centers that do not match those of their Euclidean counterparts. As in Euclidean space, we have that geodesics connecting centers of tangent circles pass through the point of tangency–similar to Figure 2.2. With a few simple substitutions, this yields formula 2.10.

$$\alpha_v = \arccos \frac{\cosh(r + r_1)\cosh(r + r_2) - \cosh(r_1 + r_2)}{\sinh(r + r_1)\sinh(r + r_2)} \tag{2.10}$$

With a bit of manipulation and the assumption that $r^* = e^{-2r}, r_1^* = e^{-2r_1}$, and $r_2^* = e^{-2r_2}$, this yields alternate equation 2.11. We will refer to r^*, r_1^*, and r_2^* as the *transformed labels* of r, r_1, and r_2. As the result is not obvious, but requires only routine algebra and trigonometric calculations, we provide a justification for equation 2.11 in Appendix A.

$$\alpha_v = 2 \arcsin \sqrt{\frac{r^*(1 - r_1^*)(1 - r_2^*)}{(1 - r^* r_1^*)(1 - r^* r_2^*)}} \tag{2.11}$$

Note that these formulas work only for triangles with finite side lengths. If a triangle contains a vertex that lies on the boundary, the sides of the triangle adjacent to that vertex are taken to have infinite length. This special case arises in mutually tangent trios of circles whenever at least one circle is internally tangent to the unit disc. Such internally tangent circles are called *horocycles*, and are taken to have radii of infinite length. Utilizing that $\cosh x = \frac{e^x + e^{-x}}{2}$, $\sinh x = \frac{e^x + e^{-x}}{2}$, and taking limits as either r, r_1 or r_2 go to infinity, we can still obtain formulas for angles of triangles, even when side lengths are infinite. As this paper does not explicitly deal with circle packings containing horocycles, only two of these formulas will be applicable within the context of this paper: determining the angle at v when

v_1 and v_2 lie on the boundary, in which case equation 2.12 holds, and determining the angle at v when v lies on the boundary, in which case equation 2.13 holds.

$$\alpha_v = 1 - 2e^{-2r} \tag{2.12}$$

$$\alpha_v = 0 \tag{2.13}$$

An interesting result is that even though we have different formulas for angles and area in hyperbolic space, the conclusions of Theorem 2.1 and Corollary 2.2 still hold. Using equations 2.11 and 2.8, a partial derivative argument as in the initial proof shows that the results remain valid.

Isometries of circle packings in this space are also affected. Although similarly we may consider translation and rotation as isometries, the means of expressing an isometry changes. We use the Möbius transformation,

$$f(z) = e^{i\theta}\frac{z - z_0}{1 - z\bar{z_0}}, \tag{2.14}$$

where z_0 denotes a point within $\mathbb{D}$ to be translated to the origin, and θ denotes an angle of rotation. Möbius transformations preserve circles, and as they are conformal, they also preserve angles. This means they maintain patterns of tangency between circles, or that they preserve circle packings as the isometries in Euclidean space do.

2.3 Spherical Geometry

Much of the origins of circle packings lie within spherical geometry. Thus, although we do not perform many calculations within this geometry, many of our proofs rely on the existence of packings on the unit sphere $\mathbb{S}^2$. We are concerned primarily with the ability to move our packings from geometry to geometry: thus, that stereographic projection preserves circles is tremendously useful! Further, If we identify coordinates in $\mathbb{S}^2$ with their counterparts in the complex plane after stereographic projection, we have that isometries in $\mathbb{S}^2$ are given by the Möbius transformations

$$f(z) = \frac{az + b}{cz + d}, \tag{2.15}$$

where a, b, c, and d are contained in $\mathbb{C}$, and $ad - bc = 1$. These transformations maps circles to circles, and preserve packings on $\mathbb{S}^2$.

Chapter 3

Foundations

As noted in the introduction, circle packings are configurations of circles with a specified pattern of tangencies. To describe these patterns, we may relate the circle packing to a simple graph by allowing nodes to represent centers of circles, and allowing an edge to join two nodes if and only if they represent the centers of two circles that are externally tangent to each other.

Starting with a circle packing and building a graph in this manner is obviously not too difficult. However, one more often has to start with a graph and construct a packing, which is significantly more challenging! If we are to start in this manner, how can we confirm that we have a circle packing if we think we have obtained one? What qualities should we start with to determine a unique circle packing? We shall start with the basics first.

3.1 Definitions and Preliminaries

Closed topological triangles are simply connected surfaces bounded by three vertices and the edges connecting them. A *triangulation* is a means of breaking down a topological surface into a collection of closed topological triangles such that any two triangles are either disjoint, intersect only at a vertex, or intersect only at a single edge and its endpoints. Any edge not contained in two closed topological triangles within a triangulation is a *boundary edge*, and any vertices which lie on boundary edges are *boundary vertices*. Any vertices within the triangulation that are not boundary vertices are *interior vertices*, and these vertices connect to other vertices only by *interior edges*.

The graphs that we work with are generally triangulations of 2-dimensional topological surfaces, which are types of simplicial 2-complexes. (The reason we work with triangulations alone will be addressed later.) We will refer to these simplicial 2-complexes as *complexes* (generally denoted by K), and to the topological triangles contained within a given complex

as *faces*. The union of all faces is called the *carrier* of the packing. Any sequence of faces $\{f_1, ..., f_n\}$ such that any element shares an edge with its successor and predecessor is referred to as a *chain*. A chain is *closed* if $f_1 = f_n$. A *null chain* is any chain that consists of a single face.

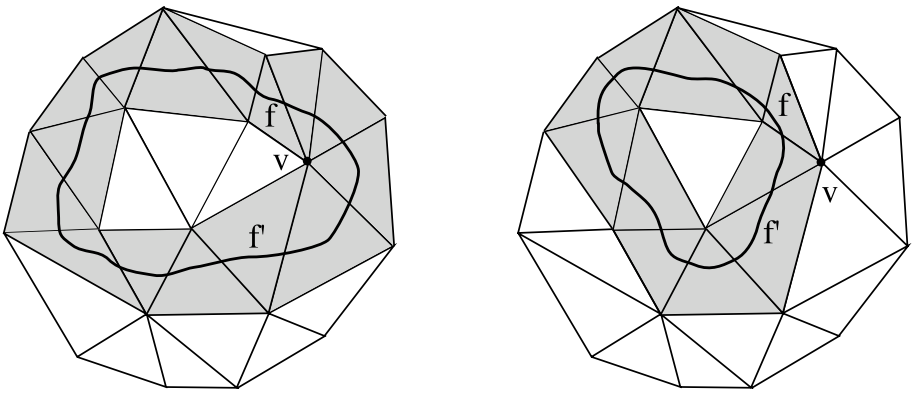

Figure 3.1: A local modification on a closed chain of faces.

In Figure 3.1, we illustrate some of the above definitions. Both images within the figure illustrate the same triangulation of a surface. Starting at face f and working counterclockwise about either grey loop until we reach f again yields a closed chain of faces. Notice that we may obtain the second chain in the figure by changing the direction the chain takes around the vertex v in heading from f' to f. Any such change where a sequence of faces in a chain is replaced with some subchain beginning and ending with the same faces is called a *local modification*. (Above, the beginning face is f' of the replaced subchain is and the ending face is f.) We also may obtain local modifications by pattern simplifications such as in the following:

$$\{..., f, f, ...\} \to \{..., f, ...\},$$
$$\{..., f, g, f, ...\} \to \{..., f, ...\}.$$

Two closed chains are *homotopic* if one may perform a finite number of local modifications on one to obtain the other. All closed chains on triangulations of simply connected surfaces are homotopic to the null chain.

Vertices in the complex are noted as neighboring whenever they are adjacent (connected by an edge) in the complex. These neighboring vertices are often addressed more simply as *neighbors*. We often use similar terminology for referring to the tangent circles within a circle packing which such vertices represent.

Any mutually tangent trio of circles is called a *triple*. The structure formed when an interior circle is grouped together with all its immediate neighbors is a *flower*. The center of this configuration is simply referred to as the center circle, while all of the outer circles are called *petals*. If we are dealing solely with vertices and not circles, the corresponding term

is *combinatorial flower* in lieu of flower. In either case, if describing petal circles in any sort of sequential form, the standard direction of ordering about the center circle is taken to be counterclockwise.

A label is a collection of potential radii $R = \{r_1, ..., r_n\}$ assigned to respective vertices $\{v_1, ..., v_n\}$ of K. We often write $R(v_k) = r_k$. Note that we make a distinction between "potential radii" and "radii" within this definition, as we do not know that the values in the label will actually yield a packing when paired with the complex–For all we know, these could be completely arbitrary values such that the circles do not "fit" together properly. The word "radius" in this work will be used in reference to both potential and true radius, with the assumption that the context will clarify which is implied.

It is often convenient to refer to specific angles at given vertices in a complex. Suppose that we have an interior vertex v in the complex K, and that K has been paired with some label R that gives v the radius r. Allow that v has petal radii $\{r_1, ..., r_k\}$ ordered so that (r, r_j, r_{j+1}) is a triple of radii for all $1 \leq j \leq k$ when we assume the convention that $r_{k+1} = r_1$. This convention reflects the fact that when we have ordered our triple like this, our "first" and "last" petal circles must form a triple with the center for our center vertex to be interior. Let $\alpha(r; r_j, r_{j+1})$ refer to the angle centered at v when we lay out the face determined by the triple of circles of these radii. (As noted in the geometries section, it is possible to determine this angle using the law of cosines in whichever space the packing is to be laid out in.) The *angle sum* at v using label R, $\theta_R(v)$ is then given by equation 3.1.

$$\theta_R(v) = \sum_{i=1}^{k} \alpha(r; r_i, r_{i+1}) \tag{3.1}$$

To better reflect the dependence of the angle sum on the radii of the petal vertices, we may also use the equivalent notation $\theta_R(v; v_1, ..., v_k)$ or $\theta(v; v_1, ..., v_k)$ for $\theta_R(v)$.

Note that the definition of angle sum refers only to vertices that are the centers of combinatorial flowers–thus, we have only referred to angle sum for interior vertices. We may extend the notion of angle sum to vertices that lie on the boundary if we allow $\{r_1, ...r_k\}$ to be neighboring radii, and only take the sum to $k - 1$.

That said, the angle sum at the interior vertices often tends to be a more useful quantity to know: given some complex K a label R is said to meet a *packing condition* at a vertex $v \in K$ if the angle sum at v is an integer multiple of 2π. A *packing label* for K is any label that meets a packing condition at all interior vertices of K. Packings meeting a packing condition by having interior angle sums of 2π are termed *locally univalent*, whereas any packing that is not locally univalent is called a *branched packing*. A circle packing is *globally univalent* (or *univalent*) if the interiors of all circles in the packing are disjoint. Univalent packings must be locally univalent, whereas the converse is not always the case.

We illustrate several of the above terms with Figure 3.2. The three images within the figure represent the same flower, but with different labels assigned to the center vertex. The petal

circles have fixed labels here. Also note that the "first" and "last" petal circles are the same: the dashing is to represent that the circle cannot be in two locations at once, even though we must calculate the angles given by both of the triples it is contained in.

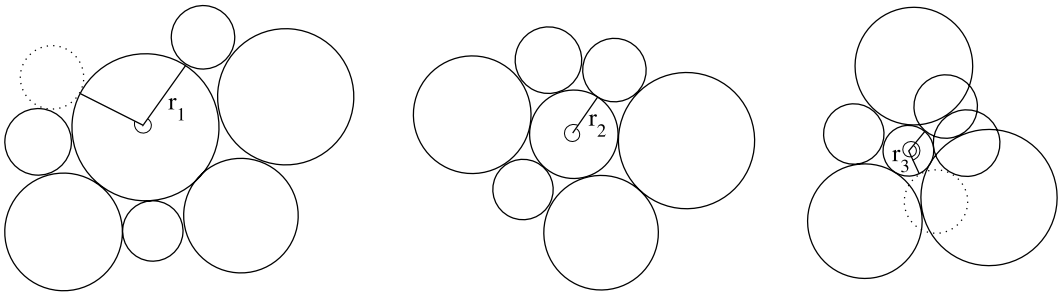

Figure 3.2: The angle sum at the center vertex changes as we alter its label, r.

In the first image within Figure 3.2, we see that the angle sum is less than 2π when the label at the center is r_1. When label r_2 is assigned, a packing condition is met. When label r_3 is assigned, the angle sum is greater than 2π. The change in angle sum as a result of changing radii is a consequence of Monotonicity I applied across all triples within the flower. An elementary way to think of this relationship between angle sum and radii is that angle sums which are "too small" imply we have central radii that are "too big," and vice versa. That we were able to find a label for the center vertex such that a packing condition was met is a consequence of Monotonicity II, similarly applied across all triples within the flower.

Finally, we have the language to determine when we have a packing!

3.2 Monodromy

As simple as it sounds, one cannot have a circle packing for a complex K without ensuring that the circles in the packing match the patterns of tangency specified in K. Recall that the distinction between "radii" and "potential radii" is made here: if the radii given in a label are such that the resulting circles do not fit together properly, there is no packing. Thus, we may also say there is no circle packing for K unless our label complements the combinatorial information specified by K. Circumstances for when this match can be guaranteed to occur are given by Theorem 3.1.

Theorem 3.1 (Monodromy)**.** *Let K be a simply connected complex. There exists a circle packing for K if and only if K has a packing label. This packing is unique up to isometries.*

Proof. We follow the method given in [7].

($\Longrightarrow$) The forward direction of this proof is fairly straightforward. If we have laid out a circle packing for K, we may take the radii given in this packing to be a label. Given any interior

circle at vertex v, we may enumerate its petal vertices $\{v_1, ..., v_k\}$ such that the triangle determined by v, v_j and v_{j+1} form a face f_j for all $1 \leq j \leq k$ if $v_{k+1} = v_1$.

This means that $\{f_1, ..., f_k\}$ form a chain of faces containing v that surround v. This chain must do so without changing direction, as all of our tangencies have been declared to be external. Further, as our faces begin and end on the same edge (the one connecting v to v_1), the chain must circumnavigate v at least once. We then have that the angle sum at v is 2π times the number of times the chain of faces circles v.

This holds for any interior vertex v, thus our label must be a packing label.

($\Longleftarrow$) Given a complex K and a packing label R, place one circle, and any one of its neighbors. We may liken the placement of any subsequent circle to the placement of its vertex, as pairing this information with the radius from the packing label will determine a circle. Further, since K is a triangulation of a surface, placing a vertex based on two mutual neighbors is akin to the placement of a face. Given two previously placed neighboring vertices and an unplaced neighbor of both, the combinatorial information of K, the radii given in R, and the geometry of the underlying space force a shape and location for the unplaced face–thus forcing a location for the new vertex. We may place the rest of the circle packing in this manner; working circle by circle. This yields a circle packing of some sort, but it remains to show that this packing is unique.

We will consider a packing to be unique if any two such orderings of placement will place a given circle in a unique location, up to isometries. In other words, if any two chains of faces starting at f_1 and ending at f_n ultimately drop f_n in the same spot, we have unique placement.

Let $\{f_1, ..., f_j\}$ and $\{g_1, ..., g_k\}$ be any two distinct chains of faces resulting from differing placement sequences such that $f_1 = g_1$ and $f_j = g_k$. (These sequences need not contain the same number of elements; they just need to start and end at the same respective faces.)

Consider the chain created by adjoining the first chain to the reverse-ordered elements of the second chain: $\{f_1, f_2, ..., f_{j-1}, f_j, f_j, g_{k-1}, ..., g_2, f_1\}$. If and only if we have unique placement of f_j, heading out along the first chain from f_1 to f_j and back down the second from f_j to f_1 places us back in the same spot. If we do not have unique placement, our final f_1 may be in a different location from our initial f_1. However, since K is a simply connected complex, closed chains are homotopic to the null chain. This means that if the null chain $\{f_1\}$ places f_1 in its initial location–and it is clear that it does–then chains homotopic to$\{f_1\}$ also place f_1 in its initial location. — the null chain clearly puts the original face in its original location, thus so must the original chain.

Therefore, we have unique placement of f_j, unique placement of the circles associated with placing f_j, and a circle packing unique up to isometries. □

Given the rigidity of this link between angle sum and circle packings, it is useful for us to develop a notion of error, or *curvature*, in packings. The difference between actual angle sum

and the desired packing condition at a given vertex may be considered a measure of error at that vertex. If $A(v)$ is the target packing condition at v, curvature $e(v)$ for a packing label R may be given by:

$$e(v) = |\theta_R(v) - A(v)|, \tag{3.2}$$

and with overall error for the label $E(R)$ given by:

$$e(v) = \sum_{i=1}^{k} e(v_i), \tag{3.3}$$

where the v_i represent interior vertices. Clearly, curvature then concentrates at vertices where the angle sum does not meet a packing condition.

Note that K must be a simply connected triangulation of a surface for us to guarantee that a circle packing can be obtained from a packing label, though. If K fails to be a triangulation, the packing becomes unstable–we do not have that the label and geometry force shapes and locations of our "faces," as in Figure 3.3. If K does not triangulate a simply connected surface, there may be chains of faces which are not null-homotopic, as in Figure 3.4. Either case causes the proof to fail. This is *not* to say graphs that are not triangulations cannot have circle packings; we just cannot verify the existence of unique packings of this sort with Theorem 3.1.

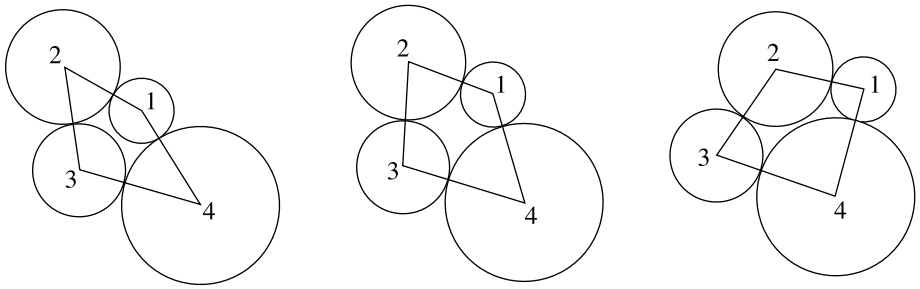

Figure 3.3: If faces are not triangles, the packing becomes unstable.

Monodromy thus serves to show that circle packings and packing labels go together hand in hand–but we need to determine that *either* may exist in general before we attempt to build a circle packing from a complex.

3.3 Existence of Packings

Given a complex K that contains only five or six vertices, we often may be able to sketch out on paper a potential configuration of circles which approximates a packing for K. That circle packings and packing labels exist for smaller complexes may seem intuitive in this

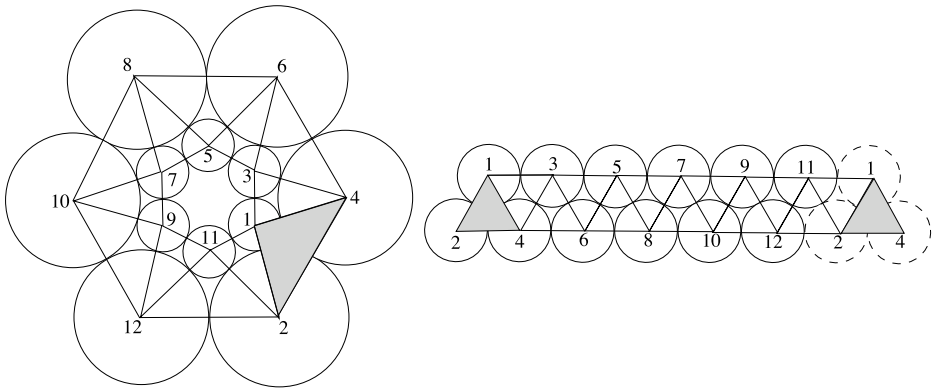

Figure 3.4: Locally equivalent, but globally dissimilar: there exists a non null-homotopic chain of faces in the original complex.

manner–but clearly it is not practical to attempt this on a complex with several thousand vertices!

For this reason, we must show that packings exist given a complex. We attempt this in a slightly roundabout manner by starting with *maximal packings* in $\mathbb{D}$–packings whose boundary circles are horocycles.

Lemma 3.2. *Suppose there exists some packing for the complex K in $\mathbb{D}$. Then, there exists a unique univalent maximal circle packing in $\mathbb{D}$ for K.*

Proof. We follow methods given in [7], employing a *Perron method*. The steps in using this Perron method are as follows:

- Show that we can construct a partially ordered set of labels from which we seek to find a solution, and that this collection is nonempty.
- Develop the notion of taking a maximum in this set, and show that the maximum of two elements is contained within the set.
- Show that the set is non-degenerate (i.e. that the maximum is not a trivial element).
- Develop the notion of a supremum on this set, and show that the supremum of the set is our solution.

Choose a packing for K in $\mathbb{D}$. Let R_0 be the label of this packing.

We now define our partial ordering and our set. For two labels R_1 and R_2, let $R_1 \leq R_2$ denote that $R_1(v) \leq R_2(v)$ for all vertices v in the complex K. Now, allow the term *subpacking label* to refer to any label for K in which the angle sum at all interior vertices is greater than or equal to 2π. Then, we may define a collection of labels $\mathcal{R}$ as follows:

$$\mathcal{R} = \{R : R \text{ is a subpacking label}\}$$

As R_0 is a packing label for K, all its interior vertices meet a packing condition and must have angle sums greater than or equal to 2π. Thus, R_0 is an element of $\mathcal{R}$, and $\mathcal{R}$ is nonempty.

Now, let us develop a notion of a maximum on $\mathcal{R}$: If R_1 and R_2 are elements of $\mathcal{R}$, let $R_3 = max\{R_1, R_2\}$ be defined at each vertex by $R_3(v) = max\{R_1(v), R_2(v)\}$ for all v in K. It is clear that $R_1, R_2 \leq R_3$, but does $\mathcal{R}$ actually contain R_3?

The answer is yes. Suppose that $R_3(v) = R_1(v)$, relabelling indices if necessary. If v is interior, the radii at the neighbors of v may only increase from the values given in R_1. By monotonicity, this will only cause the angle sum at v to increase–so the angle sum at v remains $\geq 2\pi$. Thus, our set $\mathcal{R}$ is closed under maximums.

Suppose, extending this notion of maximums, that $\hat{R} = \sup \mathcal{R}$. To show that $\hat{R}$ is non-degenerative with respect to our constructions is to show that $\hat{R}$ is finite at all interior vertices.

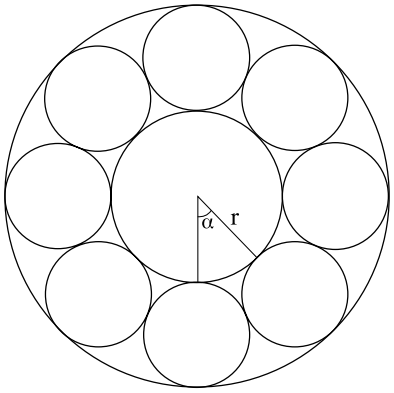

Figure 3.5: A maximal interior hyperbolic circle with n=8 petals.

In hyperbolic space, any circle which has n neighbors has a maximum radius when it and its flower are arranged in the configuration shown in Figure 3.5. It is clear that the angle α as in the picture is $\frac{2\pi}{n}$, but by our hyperbolic angle formulas, this central angle is also $\arccos 1 - 2e^{-2r}$. Setting these two expressions equal and solving for r then yields the upper bound $r \leq \log \frac{1}{\sin \frac{\pi}{n}}$ for all interior radii with n petals. Therefore, interior radii are bounded, and $\hat{R}$ is non-degenerate.

Is $\hat{R}$ actually the packing we are looking for? First, we examine the boundary radii: replacing the boundary radii in any label R in $\mathcal{R}$ with ∞ will only result in increased angle sums at interior vertices by Monotonicity I. Thus, this new label is contained in $\mathcal{R}$. As $\hat{R}$ is a supremum, the boundary radii of $\hat{R}$ must also be ∞.

As for the interior radii, consider the subset $\mathcal{R}'$ of $\mathcal{R}$ generated by increasing each of the boundary radii to the desired value ∞, as above. Since $\hat{R}$ is the supremum of $\mathcal{R}$ (and of $\mathcal{R}'$), given n in $\mathbb{N}$, for each interior vertex v there exists some label R such that $\hat{R}(v) - R(v) < \frac{1}{n}$. Taking the maximum among such R to be R_n, we have that $\hat{R}(v) - R_n(v) < \frac{1}{n}$ for each

interior vertex in K. Therefore, $\lim_{n\to\infty} R_n(v) = \hat{R}(v)$ for all v in K.

We have that angle sums, as given by the law of cosines, are continuous functions of the radii involved. Thus, $\lim_{n\to\infty} R_n(v) = \hat{R}(v)$ implies that $\lim_{n\to\infty} \theta_{R_n}(v) = \theta\hat{R}(v)$. As elements of $\mathcal{R}'$ have angle sums bounded below by 2π, $\hat{R}$ must also have angle sums bounded below by 2π. If, in fact $\hat{R}(v) > 2\pi$ for some vertex v, this implies that we should be able to make a minute increase to the radius at v without having the angle sum at v dip below 2π. (Yet again this is possible because angle sums are continuous functions of the radii involved, as per Monotonicity II.) This, however, contradicts that $\hat{R}$ is the supremum of $\mathcal{R}$. Thus the angle sum at each interior vertex of K must be 2π.

To show that $\hat{R}$ is unique, suppose to the contrary: that there exists some other maximal packing label R' such that $R' \neq \hat{R}$. Since R' clearly belongs to $\mathcal{R}$, we must have that $R' \leq \hat{R}$, since $\hat{R}$ is the supremum of $\mathcal{R}$.

Consider the area of the carriers of the packings given by these two labels. Since K is a triangulation, we have that the area of a triangle (thus the area of the carriers of our packings) can be determined by angle sums. If we allow $\{v_1, ..., v_p\}$ to denote the interior vertices and $\{w_1, ..., w_q\}$ to denote the boundary vertices, then repeated use of the triangle area formula given by equation 2.8 states that the area of our carriers in R' and $\hat{R}$ are given by

$$Area_{R'}(K) = F\pi - \sum_{i=1}^{p} \theta_{R'}(v_i) - \sum_{j=1}^{q} \theta_{R'}(w_j)$$

$$Area_{\hat{R}}(K) = F\pi - \sum_{i=1}^{p} \theta_{\hat{R}}(v_i) - \sum_{j=1}^{q} \theta_{\hat{R}}(w_j)$$

But, as R' and $\hat{R}$ are both locally univalent packing labels, interior angle sums are 2π. Further, as they are also both maximal packings, boundary circles are horocycles with radius ∞. By our special case angle formulas, boundary angle sums are then zero. Thus,

$$Area_{R'}(K) = F\pi - 2p\pi = Area_{\hat{R}}(K).$$

By Monotonicity I, we have that areas of faces are strictly monotonically increasing in radii. Therefore, as R' results from decreasing interior radii from values in $\hat{R}$, we have that the areas of faces of K in R' are strictly less than the areas of the same faces in $\hat{R}$. Therefore,

$$Area_{R'}(K) < Area_{\hat{R}}(K),$$

a contradiction. We must have that $\hat{R}$ is a unique maximal circle packing in $\mathbb{D}$. □

Now, we may show existence of packings given complexes alone.

Theorem 3.3. *If K is a complex which is finite, simply connected, and with nonempty boundary, then K has a univalent circle packing. Thus, there exists some packing label for K.*

Proof. We follow the method given in [7].

This proof is by induction on V, the number of vertices in the complex K. The base case is given by $V = 3$, where K is merely one topological triangle. We may position any three circles together so as to be mutually tangent, so any label will yield a packing here.

For our inductive hypothesis, assume that our proposition holds for any complex having V or fewer vertices. Suppose that we are given a complex K fitting our hypothesis, and that K has $V + 1$ vertices. As our complex has nonempty boundary and more than 3 vertices, we may select some boundary vertex w which lies on an interior edge. From here we have two cases to consider:

Case 1. Suppose that w lies on some interior edge whose other endpoint is also a boundary vertex, u. We may obtain two complexes, K_1 and K_2 by cutting K in two pieces along this edge, as in Figure 3.6.

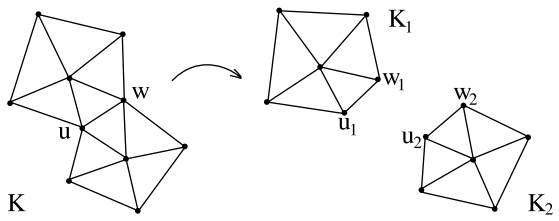

Figure 3.6: The complex K is cut into two pieces.

Since splitting K in this manner produces two complexes with fewer vertices than K, by our induction hypotheses, there exist circle packings for K_1 and K_2. By Lemma 3.2, we can extend this to say that we have maximal packings for K_1 and K_2 in $\mathbb{D}$. We may then find the transformation that sends the horocycle c_{u_1} to the circle $c_u = \{|z - \frac{1}{2}| = \frac{1}{2}\}$, and c_{w_1} to the circle $c_w = \{|z - \frac{1}{2}| = \frac{1}{2}\}$. (This is an isometry of the packing for K_1, as it sends the point of tangency of the horocycles c_{u_1} and c_{w_1} to the origin, and rotates the packing such that the center of c_{w_1} lies on the positive x-axis.) Similarly, we may find an isometry of the packing for K_2 that sends the horocycles c_{u_2} and c_{w_2} to c_u and c_w. Given that complexes preserve orientation, the circles not sent to c_u and c_w in the packings of K_1 and K_2 are disjoint–they lie on opposite sides of the x-axis. We may then superimpose the two packings for K_1 and K_2 upon each other to obtain a univalent hyperbolic packing for K, as in Figure 3.7.

Case 2. Suppose that any edge w lies on contains an interior vertex. We may create a new complex K' by removing the combinatorial star at w–i.e., w and all edges containing w–from the complex K. Since this new complex contains V vertices, by the induction hypothesis, we

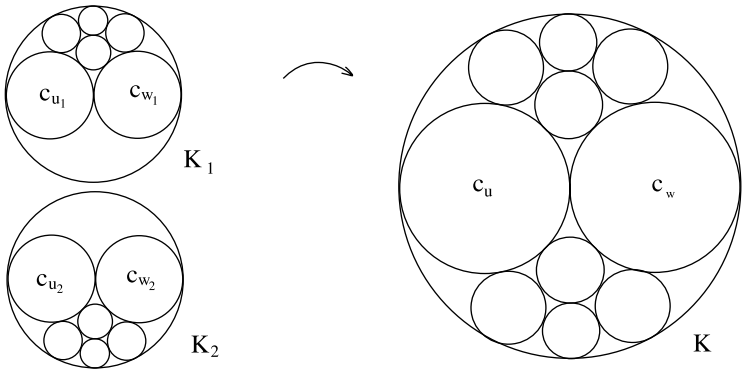

Figure 3.7: The packing for K is built from two superimposed pieces.

may find a circle packing for K' in $\mathbb{D}$. Then, by Lemma 3.2, there exists a maximal univalent packing for K' in $\mathbb{D}$.

Suppose we project this maximal packing along with the boundary disc to the unit sphere $\mathbb{S}^2$. Then, the compliment of $\mathbb{D}$ maps to a circle which covers a hemisphere of $\mathbb{S}^2$ and is externally tangent to all of the boundary circles in the packing for K'. If we allow this new circle to represent c_w and ignore extraneous tangencies, this new packing on $\mathbb{S}^2$ contains the combinatorics of K. Thus, we may select any circle on $\mathbb{S}^2$ disjoint from the carrier or the circles of the packing and use a Möbius transformation to send this circle to the southern hemisphere of $\mathbb{S}^2$. After this, projecting back to the unit disc gives us a univalent hyperbolic packing for K.

Both cases yield a univalent hyperbolic packing for K. Inductively, we may conclude that any finite, simply connected complex with nonempty boundary has a univalent hyperbolic packing. As we "draw" hyperbolic figures in the Euclidean plane, that there exist univalent Euclidean packings for such complexes follows from the hyperbolic case. □

Even after all of this effort, one may wonder if showing that a general circle packing exists for K may have just been busy work. However, there are graphs–obviously not complexes by our definition–composed of topological triangles that do not admit circle packings in either $\mathbb{C}$ or $\mathbb{D}$. Figure 3.8 for example, requires that two circles be tangent to each other exactly twice, which is impossible on a flat surface using any standard definition of "circle."

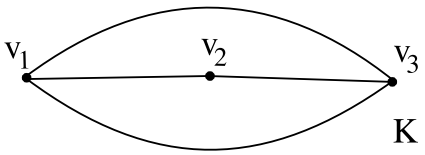

Figure 3.8: A complex that does not admit a packing in $\mathbb{C}$ or $\mathbb{D}$.

To build a packing from the bottom up, a complex is then a very important requirement

indeed! Now that we have existence of circle packings, we know that K guarantees existence of packing labels by Monodromy. We now investigate how to go about finding a packing label.

3.4 Generalizing from the Boundary

If one is familiar with complex analysis, one will certainly recognize the Dirichlet problem: given a continuous function on the boundary of a region, find a function that is harmonic over the interior, continuous over the entire region, and matches the initial function on the boundary. As the roots of circle packings lie in approximation of concepts from complex analysis, we should hope that there exists an analogous discrete result for packings–and as it turns out, there is one.

As many of the programs we reference later restrict to packings with a finite number of vertices, finite radii, and angle sums of 2π, in this proof we also restrict to the case that any packing label we target is one that has a finite number of elements which all yield interior angle sums of 2π.

Theorem 3.4 (Boundary Value Theorem)**.** *Let the complex K be a finite triangulation with nonempty boundary such that there exists some locally univalent packing label R_0 for K. Write $w_1, ..., w_q$ for the boundary vertices of K. Then, given any numbers $r_1, ..., r_q$ in $(0, \infty)$, there exists a unique packing label $\hat{R}$ for K such that $\hat{R}(w_j) = r_j$ for $j = 1, ..., q$. This result holds for both Euclidean and hyperbolic packings.*

Proof. We follow methods given in [4] and [6], again employing a Perron method. Details of the proof which strongly resemble those in the proof of Lemma 3.2 shall be omitted.

Let a collection of boundary radii $r_1, ..., r_q$ corresponding to boundary vertices $w_1, ..., w_q$ in K be given, and assume that there exists a packing label R_0–not necessarily with these boundary radii–for the complex K.

We may define a collection of labels $\mathcal{R}$ as follows:

$$\mathcal{R} = \{R : R \text{ is a subpacking label with } R(w_j) \leq r_j \text{ for } j = 1, ..., q\}$$

If R_0 happens to be contained in $\mathcal{R}$, then $\mathcal{R}$ is clearly nonempty. If not, then consider Euclidean scalings of the circle packing generated by R_0. If r_k is the smallest of our boundary radii, we can pick a small enough scaling factor to fit the entire packing within a circle of radius r_k. The interior angle sums will still all be 2π, as scaling does not change angles. Further, all boundary radii in the new label are forced to be less than the smallest possible prescribed boundary radii. Therefore, $\mathcal{R}$ contains at least this label, and is nonempty.

If R_1 and R_2 are elements of $\mathcal{R}$, let $R_3 = max\{R_1, R_2\}$. By a previous argument, the angle sum at all interior v in K remains $\geq 2\pi$ in the label R_3. If v is exterior, that $R_1(v)$ was

contained in $\mathcal{R}$ implies that $R_3(v)$ meets the boundary criteria for R_3 to be an element of $\mathcal{R}$. Therefore, our set $\mathcal{R}$ is closed under maximums.

Suppose that $\hat{R} = \sup \mathcal{R}$. To show that $\hat{R}$ is non-degenerative is to again show that $\hat{R}$ is finite at all interior vertices. It has already been shown why this holds in the hyperbolic case, and the Euclidean case is fairly simple: all interior circles must be contained within the carrier of the packing. The carrier has a perimeter yielded by $2\sum_{i=1}^{q} r_i$, which is finite. No circle of infinite radius may fit in such a bounded region! As interior radii are bounded in both the hyperbolic and Euclidean case, we have $\hat{R}$ is non-degenerate.

All signs point toward $\hat{R}$ as the solution to our discrete Dirichlet problem, but we must show that it actually forms a solution. First, we show that $\hat{R}$ has the correct boundary radii:

Select any element R in $\mathcal{R}$, and let R' be the label that results from replacing the boundary radii of R with the desired boundary values. Using R' as a label, any interior vertices of K that only have interior vertices as neighbors have unchanged angle sums. Any other interior vertices of K have angle sums that can only stand to increase, by Monotonicity II. R' is a subpacking with boundary radii meeting the criteria for elements of $\mathcal{R}$, thus R' is contained in $\mathcal{R}$. As the boundary radii are now bounded above *and* below by $r_1, ..., r_q$ we must have that the boundary radii of $\hat{R}$ match our prescribed values.

For the interior radii, following the argument from Lemma 3.2 shows that the angle sum at each interior vertex of K must be 2π. Therefore, $\hat{R}$ is locally univalent and meets a packing condition at each vertex v of K, and $\hat{R}$ is a packing label. All that is left is to show that $\hat{R}$ is unique.

Suppose to the contrary: that there exists some other packing label R' such that $R' \neq \hat{R}$. Since R' clearly belongs to $\mathcal{R}$, we must have that $R' \leq \hat{R}$, since $\hat{R}$ is the sup of $\mathcal{R}$.

In the Euclidean case, we consider the angle sums on the boundary. K is a triangulation, and Euclidean triangles have angles which sum to π. Thus, the sum of all boundary angle sums is given by $F\pi - 2p\pi$ where F is the number of faces and p is the number of interior vertices. As the number of faces and interior vertices is constant in K, even after switching labels, we have then that

$$\sum_{j=1}^{q} \theta_{\hat{R}}(w_j) = F\pi - 2p\pi = \sum_{j=1}^{q} \theta_{R'}(w_j)$$

as both labels are packing labels, and interior angle sums are all 2π.

Now, examine the difference in boundary angle sums due to changing the label on K from R to R': we decrease all interior radii, while maintaining the boundary radii. By repeated application of Monotonicity I to the radii of interior vertices neighboring the boundary, we

have that the boundary angle sums will *all* be strictly decreasing. Thus,

$$\sum_{j=1}^{q} \theta_{\hat{R}}(w_j) > \sum_{j=1}^{q} \theta_{R'}(w_j),$$

a contradiction. Therefore, in the Euclidean case we have that $\hat{R}$ is a unique solution to the boundary value problem.

In the hyperbolic case, we have similar to before that

$$Area_{R'}(K) < Area_{\hat{R}}(K).$$

If we allow $\{v_1, ..., v_p\}$ to denote the interior vertices, repeated use of the triangle area formula given by equation 2.8 then yields

$$F\pi - \sum_{i=1}^{p} \theta_{R'}(v_i) - \sum_{j=1}^{q} \theta_{R'}(w_j) < F\pi - \sum_{i=1}^{p} \theta_{\hat{R}}(v_i) - \sum_{j=1}^{q} \theta_{\hat{R}}(w_j)$$

But, as R' and $\hat{R}$ are both locally univalent packing labels the sum of interior angle sums using either label is given by the same value. Thus,

$$\sum_{j=1}^{q} \theta_{\hat{R}}(w_j) < \sum_{j=1}^{q} \theta_{R'}(w_j).$$

Similar to the Euclidean case however, we have by Monotonicity I that the boundary angle sums decrease from $\hat{R}$ to R', or

$$\sum_{j=1}^{q} \theta_{\hat{R}}(w_j) > \sum_{j=1}^{q} \theta_{R'}(w_j);$$

a contradiction. This implies that in the hyperbolic case $\hat{R}$ is a unique solution to the boundary value problem. □

What the Boundary Value Theorem boils down to then, is this key point: If K is a complex, we can specify boundary values to mold a packing with the combinatorics of K into a form that we desire. By Theorem 3.3, we need a complex to guarantee a packing exists at all. Considering Figure 3.8 again, it should be intuitive that if we cannot pack a surface while guided by the combinatorics of some K, we should not reasonably expect to be able to select boundary radii values for this K and get a packing!

Chapter 4

Iterative Methods for Labels

With the proof of Theorem 3.4, we should come away with the idea that local corrections may cause an overall reduction in curvature in a packing. This is the fundamental idea behind many algorithms which determine packings.

4.1 Why Iterate?

If we are given a list of desired boundary radii along with a set of packing conditions, there is certainly a task ahead of us if we wish to determine a packing label.

Suppose we want to give the process a shot using brute force–i.e., the direct method. As we have seen before, the basic idea in checking whether a label constitutes a packing label is the notion of the packing condition. If, for example, we desire our end packing to be locally univalent, we know that the angle sum at each interior vertex must be 2π. Thus, given any interior vertex v, we have $\theta_R(v) = 2\pi$.

This looks simple, but the notation suppresses all of the menial calculations that go into determining an angle sum. Since we essentially determine the angle sums at each interior vertex in the packing using the law of cosines, at any interior vertex v with k neighbors and radius r, we have in Euclidean space:

$$2\pi = \sum_{i=1}^{k} 2 \arcsin \sqrt{\frac{2r_i r_{i+1}}{(r + r_i)(r + r_{i+1})}},$$

and in hyperbolic space:

$$2\pi = \sum_{i=1}^{k} 2 \arcsin \sqrt{\frac{r^*(1 - r_i^*)(1 - r_{i+1}^*)}{(1 - r^* r_i^*)(1 - r^* r_{i+1}^*)}}.$$

If we keep in mind that an attempt at direct solution yields a system of as many of these equations as there are interior vertices, it is fairly easy to see why this brute force method isn't very popular: For even fairly small circle packings, attempting to find a solution to the system may be computationally taxing, if not impossible. On the other hand, for packings involving several hundred thousand circles or more, it becomes an absolute nightmare.

As it turns out, it is much, much quicker to use iterative methods to determine a circle packing.

4.2 General Iterative Methods

We mentioned very briefly at the end of the proof of Theorem 3.4 that one may make adjustments in radii to "fix" an angle sum, and reduce curvature at a given vertex. The set of labels that we were working with was very restrictive in that all possible angle sums were $\geq 2\pi$, though. It turns out that if we make similar adjustments on general labels, however, that we still are given a reduction in overall curvature!

Theorem 4.1. *Let R be a label for the complex K, and assume that a packing label can be found for K. If at any vertex v in K we have curvature $e(v) > 0$, total curvature $E(R)$ is monotone decreasing with label corrections to $R(v) = r$ that do not allow angle sums to "overshoot" the target packing condition at v. That is, if R' is a label that matches R at every vertex except for where we have made the label correction $R'(v) = r'$, then we have $\theta_R(v) > \theta_{R'(v)} \geq A(v)$ and $\theta_R(v) < \theta_{R'(v)} \leq A(v)$ imply $E(R) \geq E(R')$. This result holds for both Euclidean and hyperbolic labels.*

Proof. We fill in details from a proof sketch suggested by [4]. There are two cases that allow $e(v) > 0$: either the angle sum $\theta_R(v)$ at v is strictly greater than or strictly less than the target packing condition $A(v)$.

Assume that $\theta_R(v) > A(v)$. (The proof follows very similarly for the opposite case.) This implies that the radius at v is too small. As angle sums are continuous functions of the radii involved, we may increase the value of $R(v) = r$ to some $R'(v) = r'$ such that $\theta_r(v) > \theta_{R'}(v) \geq A(v)$. In doing so we have made a change to the label R such that our correction does not allow the new angle sum to overshoot our target packing condition. Consider the changes that this label adjustment yields on any face containing the vertex v.

If we are in Euclidean space, the angles of a triangle sum to π. Thus,

$$\alpha + \beta + \gamma = 2\pi = \alpha' + \beta' + \gamma',$$

which implies

$$\alpha - \alpha' = \beta' - \beta + \gamma' - \gamma.$$

As $\alpha - \alpha'$ yields the correction in angle sum at v due to the change in this face, we see that over all faces containing v, the label correction yields at most an equal increase in the angle sums of neighboring vertices. At best, this yields improvement in the curvatures of neighboring vertices of v; at worst we have retained our old total curvature but changed the vertices at which it concentrates. Thus, $E(R) \geq E(R')$.

If we are in hyperbolic space, angles of faces do not sum to a constant term–but they do determine triangle area. As the face induced by our label adjustment has greater area than our initial face by Monotonicity I, equation 2.8 yields:

$$\pi - \alpha - \beta - \gamma < \pi - \alpha' - \beta' - \gamma',$$

which implies

$$\alpha - \alpha' > \beta' - \beta + \gamma' - \gamma.$$

As $\alpha - \alpha'$ yields the correction in angle sum at v due to the change in this face, we see that over all faces containing v, the label correction yields an increase in the angle sums of neighboring vertices which amounts to less than the correction in angle sum at v. At best, this yields improvement in the curvatures of neighboring vertices of v; at worst the curvature correction at v still surpasses any new curvature generated at the petal vertices of v. Thus, $E(R) \geq E(R')$. □

This forms the basic principle upon which we iterate: Sequences of error terms generated upon subsequent corrections to a label are monotonically decreasing. As error is clearly bounded below by zero, we should get that this sequence converges to some value of error. If we are working with subpacking labels (or superpacking labels where all interior vertices have angle sums $\leq 2\pi$ for that matter), then we may use arguments similar to those in the proof of Theorem 3.4 to show that this error converges to zero. For general packing labels we are somewhat stuck out on a limb–but experimental observations in packing circles have generally shown that the rigidity of the geometry involved makes iterative processes of this sort "so stable that almost any iterative procedure will succeed." [4]

We know now that iterative procedures will give results–all that is left is to develop such an iterative procedure.

4.3 The Uniform Neighbor Model

We develop a correction scheme based on the first of two label correction algorithms from Collins and Stephenson's "A Circle Packing Algorithm." Let R be a label for the complex K such that the boundary radii have been assigned desired target values. If $e(v) > 0$ for some interior vertex v in K, we make label corrections to v as follows:

- Calculate the angle sum $\theta_R(v)$ at v.

- Count the petal vertices of v. If v has n petal vertices, determine the radius $\hat{r}$ which would be necessary for v to have an angle sum of $\theta_R(v)$ if its n neighbors had uniform radius $\hat{r}$.
- Determine the value r' such that a vertex with radius r' surrounded by neighbors with radius $\hat{r}$ has an angle sum that matches the packing condition.
- Let r' be the label correction in R for the vertex v.

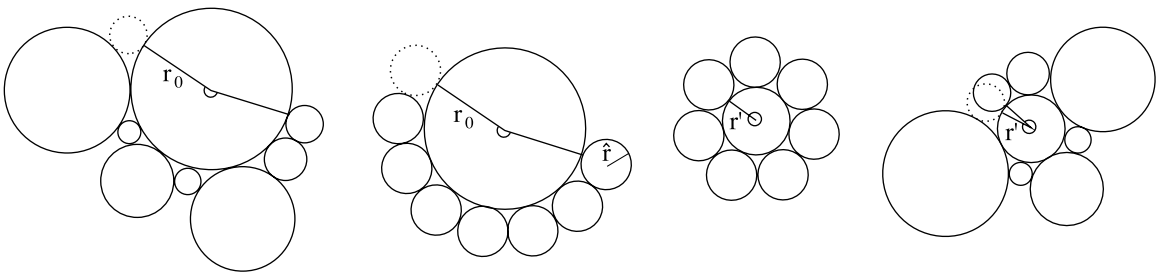

Figure 4.1: Steps of the Uniform Neighbor Model.

That this is dubbed the Uniform Neighbor Model now makes sense: we base our correction to r at v off of an approximation to the current label where the radii of neighbor vertices to v are uniform. Given r and $\theta_R(v)$, the calculations for $\hat{r}$ and r' are then reasonably simple.

Let k be the number of petal circles to v, A be the target packing condition at v, $\beta = \sin\frac{\theta}{2k}$, and $\delta = \sin\frac{A}{2k}$.

We have for the Euclidean setting that $\hat{r}$ and r' are given by

$$\hat{r} = \frac{\beta}{1-\beta} r, \tag{4.1}$$

$$r' = \frac{1-\delta}{\delta}\hat{r}. \tag{4.2}$$

Further, taking a hyperbolic label to be transformed as in Section 2.2, we have for the hyperbolic setting that $\hat{r}$ and r are given by

$$\hat{r} = \frac{\beta - \sqrt{r}}{\beta r - \sqrt{r}}, \tag{4.3}$$

$$r' = \left(\frac{2\delta}{\sqrt{(1-\hat{r})^2 + 4\delta\hat{r}} + 1 - \hat{r}}\right)^2. \tag{4.4}$$

Deriving these formulas require not much more than routine algebra and trigonometry. As they are not directly clear however, for the interested reader I sketch the derivation for the these formulas in the appendices.

As it turns out, this method of correction yields estimates with a very useful property, stated by Theorem 4.2.

Theorem 4.2. *Let $\theta(r) = \theta(r; r_1, ..., r_k)$, and $= \hat{\theta}(r) = \theta(r; \overbrace{\hat{r}, ..., \hat{r}}^{k})$. Let $\hat{r}$ be chosen such that $\theta(r_0) = \hat{\theta}(r_0)$ for some $r_0 > 0$. Assuming that $\{r_1, ..., r_k\}$ are not all equal (thus not all equal to $\hat{r}$), then*

$$\frac{d\hat{\theta}}{dr}(r_0) < \frac{d\theta}{dr}(r_0). \tag{4.5}$$

These results hold for both Euclidean and Hyperbolic labels.

We do not prove this theorem here, as for the most part it is a direct proof involving partial derivative calculations performed on our angle sum formulas. The interested reader may refer to [4] for details. Although the computations are not hard, mathematics software of some sort is recommended if one wishes to verify the result, as the steps leading up to the conclusion are quite messy.

Corollary 4.3. *Assuming the conditions given in Theorem 4.2, we have that $\theta(r) < \hat{\theta}(r)$ for $0 < r < r_0$ and $\theta(r) > \hat{\theta}(r)$ for $r_0 < r$. In other words, label adjustments to R made by the Uniform Neighbor Model do not allow angle sums to overshoot target packing conditions.*

Proof. This result is also given in [4]. Given the conclusions of Theorem 4.1, since $\theta(r_0) = \hat{\theta}(r_0)$, the above inequalities follow.

To show that the Uniform Neighbor Model does not overcorrect in its label adjustments, suppose that $\theta(r_0)$ is greater than the target packing condition $A(r)$. (The proof follows similarly if the opposite is true.) We have $\theta(r_0) = \hat{\theta}(r_0)$ for some $\hat{r}$, which implies that the label r_0 is too small for the flower $(v; \hat{v}, ..., \hat{v})$ by Monotonicity II. Corrections by the Uniform Neighbor Model will then replace r_0 with some label $r > r_0$ such that $\hat{\theta}(r)$ is equivalent to the packing condition at v. However, by the above inequalities this implies $\theta(r) > \hat{\theta}(r)$–that the label correction still leaves us with an angle sum at v greater than the packing condition.

Therefore, we have that the Uniform Neighbor Model does not allow label corrections which allow subsequent angle sums to overshoot target packing conditions. □

By Theorem 4.1, label adjustments to R which do not "overcorrect" any radii will lead to a reduction in curvature in a packing. This is a tremendous improvement over means of label correction such as Newton's Method which are computationally intensive and where we cannot guarantee that our label adjustments do not overshoot the target packing label. As the Uniform Neighbor Model makes *only* corrections of the type called for by Theorem 4.1, we will be able to reduce curvature in an attempt to approximate a packing for K. This leaves the general iterative process for determining a new label as follows:

- Select any label R_0 that assigns the desired boundary radii to their appropriate vertices in K. Set $\epsilon > 0$.

- For any label R_n, cycle through all interior vertices (in no necessary order) performing label corrections as per the Uniform Neighbor Model. Let R_{n+1} denote the resulting label, and count this sequence of steps as one iteration.
- In some predetermined manner, calculate the angle sum error of the current label. If this error is less than ϵ, we declare the current label to be our approximation to the packing label. (Sums of curvature work as an error calculation, as do measures of the "curvature vector", $[\sum_{i=1}^{k} e(v_i)^2]^{\frac{1}{2}}$. The former is more strict, but both will serve our purposes here.)
- If the error is greater than ϵ, then perform more iterations until we have reached an approximation for the packing label.

4.4 Acceleration

We are surely not the first to have attempted to design a circle packing program. As in [4], the Uniform Neighbor Model has been implemented on larger scales, and computational experiments have shown that convergence after large numbers of iterations appears uniform: i.e. if $\bar{R}$ is a solution to the boundary value problem, and R_l and R_{l+1} are consecutive approximations made with the Uniform Neighbor Model, then for some large l

$$R_{l+1} - \bar{R} \approx \lambda(R_l - \bar{R}) \tag{4.6}$$

holds element by element in the labels for some $\lambda < 1$. Solving for R_{l+2} and R_{l+1} in this format and determining the difference yields

$$R_{l+2} = R_{l+1} + \lambda(R_{l+1} - R_l). \tag{4.7}$$

More manipulation to the initial equation also yields

$$\bar{R} = R_{l+1} + \frac{\lambda}{1-\lambda}(R_{l+1} - R_l). \tag{4.8}$$

As equation 4.7 targets the next iterate, we denote any approximation using this formula *acceleration*. As equation 4.8 targets the final solution, we denote any approximation using this formula *super acceleration*.

As approximation using these methods requires information on the last two iterations (R_l and R_{l+1}), we give the algorithm in its full sequence as follows.

- Select any label R that assigns the desired boundary radii to their appropriate vertices in K. Set ϵ and $\delta > 0$, $Error = \epsilon + 1$, $\lambda = -1$, and $flag = 0$.

- Set $Error_0 = Error$, $\lambda_0 = \lambda$, and $R_0 = R$. This will maintain a record of the values of elements from the last iteration.
- Perform one iteration of corrections on the label using the Uniform Neighbor Model. Determine the error estimate $Error = [\sum_{i=1}^{k} e(v_i)^2]^{\frac{1}{2}}$, where $\{v_1, ..., v_k\}$ are the interior vertices of the packing.
- Set $\lambda = \frac{Error}{Error_0}$. (Note from equation 4.6 that we may take new error divided by old error as an estimate for λ.) Then, set $flag = 1$.
- If $flag_0 = 0$ and $\lambda < 1$, we may attempt to perform acceleration. The point of setting $flag$ and $flag_0$ is to not have our program attempt to run acceleration too many times. As we gather information on λ based on the approximations made by the Uniform Neighbor Model, performing several iterations of acceleration in a row may negate the purpose and cause our error to increase!
  - Set $Error = \lambda Error$.
  - If it appears that λ is converging to some value–i.e., if $|\lambda - \lambda_0| < \delta$–then set $\lambda = \frac{\lambda}{1-\lambda}$ in preparation for running super acceleration.
  - Determine the largest potential value of λ^* such that $R + \lambda^*(R - R_0)$ yields legal values of radii.
  - Set λ equal to the minimum of $\{\lambda, .5\lambda_*\}$. We wish to obtain fast convergence, but we do not wish to obtain values that are not legal radii on any iteration!
  - Set the new label R equal to $R + \lambda(R - R_0)$, and set $flag = 0$.
- If the error is greater than ϵ, then perform more iterations until we have reached an approximation for the packing label.

Chapter 5

Programming Considerations and Placement

5.1 Input

In the development of our programs, we took the liberty of making the assumption that our given complex represented a simply connected triangulation of a surface. Further, as in Theorem 3.4, we assume that any complex we use will be packable.

All programs modeled on the above methods were implemented in *Mathematica*, version 5.2. Relevant code for our four programs may be found in the appendices.

All input files used in testing the algorithms were generated by the programs *subdivide.c* and *tilepack.c* as written by J. W. Cannon, W. J. Floyd, and W. R. Parry. The first program, *subdivide.c*, uses specified finite subdivision rules to tile a quadrilateral. The second program, *tilepack.c*, forms a triangulation based on the configuration of tiles given in the first program and outputs it in a format usable for K. Stephenson's *CirclePack*. (See [2] and [3] for the first two programs, and [5] for *CirclePack*.)

Files output by *tilepack.c* were stripped of their heading information to obtain raw data of the form used in Table 1. We assume the default enumeration of vertices as specified by the program. Although this is a bit difficult to note in our example, the ordering of the vertices in these files tends to not follow any easily discernible pattern–i.e., our first vertex isn't always the center, and the ordering doesn't always work its way from the inside out. We essentially start from scratch with the bare minimum information on the structure of our complex.

On a given row, the first element refers to a vertex and the following elements provide information about that vertex: The second element labels number of triangular faces that the vertex is adjacent to, while the remaining elements list petal vertices in consecutive order.

Table 1: Input file formatting

vertex	faces	neighbors
1	6	2 3 4 5 6 7 2
2	2	7 1 3
3	2	2 1 4
4	2	3 1 5
5	2	4 1 6
6	2	5 1 7
7	2	6 1 2

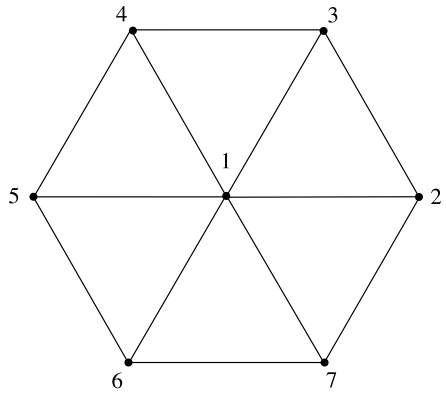

Figure 5.1: The complex from Table 1

This means any two consecutive numbers in this list paired with the original vertex will form a triangular face. If the vertex that a given row describes is interior, this list of petal vertices will include an element that appears twice–once at the beginning of the neighbor list, and once at the end. This reflects the fact that petals to interior vertices must neighbor at least two other petal vertices, and also serves to let us differentiate between boundary and interior vertices within the file.

5.2 Iterative considerations

To serve as somewhat of a standard, in testing each input file several items remained constant. On each test run, the prescribed boundary radii were set to 1, and the prescribed "dummy values" for interior radii were set to .1. In some of the hexagonal packings, a dummy value identical to the given boundary conditions causes very uninteresting results in the Euclidean setting–this automatically starts us off with a packing label, so we don't really get to test the program! There was no special reasoning behind this value, as it was selected simply

because it is not equal to 1.

The value for ϵ was held to a constant $5 \cdot 10^{-15}$. This value was small enough to allow most of the larger circle packings to pass visual inspection upon output, without taking an unreasonable amount of time on some of the slower programs. Error within the programs is measured by taking the magnitude of the "curvature vector", i.e. $(\sum_{i=1}^{k} e(v_i)^2)^{\frac{1}{2}}$. Although this is less stringent than total sum error $\sum_{i=1}^{k} e(v_i)$, the computations in acceleration require use of this form of error; so we make it the default.

A second note on the error measure from the programs: the total error for any iteration is actually an over-approximation of error on that iteration. For sake of efficiency, we take the curvature *prior* to correction at a vertex to be that vertex's error at the end of the iteration. As we recall from Theorem 4.1, label corrections of the sort given by the Uniform Neighbor Model are guaranteed to reduce or redistribute curvature within a packing. Thus, taking the error prior to the correction serves as an over-approximation of the error on that iteration. This was an original improvement to the algorithms suggested by [4].

Accuracy became a major issue for many of the packings. Keeping in mind that we often work with thousands of circles or more, rounding can throw answers off greatly–especially in the hyperbolic setting. Default accuracy settings often do not accommodate the particularly small values of ϵ needed for larger complexes to pack. Further, hyperbolic computations often have us dealing with very small and very large numbers simultaneously. (Suppose for example that one of the vertices in a complex has radius 50. As we work with transformed labels, this will eventually call for the usage of numbers such as e^{-100} and e^{100} in several computations!) The programs we used instructed *Mathematica* to set accuracy to 30 places on all test runs.

In cycling through vertices to make corrections via the Uniform Neighbor Model, the order of the cycling is given by the default ordering of the vertices in the input file, skipping boundary vertices. This was done for consistency. It would, for example be possible to test a random walk variant, or a scheme that selects random interior vertices to correct–but random schemes give different iteration counts on different trials, making it hard to measure effectiveness. It would also be possible to form an algorithm that iterates based on something more purposeful, such as location of the vertex in the complex; perhaps starting near the boundary and working inward, or iterating first on circles with a specified number of neighbors. At any rate, programs that dictate that each vertex be met in a given order on each pass do not differ much from the current. Using the Uniform Neighbor Model, error generally decreases no matter what the cycling order is, so it has not been made an issue here.)

In discussing effectiveness of the programs, we mainly discuss iteration counts. Recall that acceleration is really a pass-through of Uniform Neighbor Model paired with possible ac-

celeration and/or super acceleration calculations. For sake of comparison, we denote one iteration in any of the programs as a step involving one pass through all interior vertices of K making corrections using the Uniform Neighbor Model. This makes one pass through the interior vertices of K in a UNM program one iteration, and one pass through K paired with acceleration calculations just one iteration. We may specify a count of acceleration iterations and super acceleration iterations by the number of times these calculations are actually performed. Note that while we take a count of iterations to determine effectiveness of the programs, it should be expected that the number of iterations increases as the number of circles in a packing increases.

5.3 Circle Placement

Radii have been discussed much within this work, but recall that the point of many of these exercises was to be able to construct a model. It turns out that this goal that we've been building toward is actually rather anticlimactic: we've already detailed a rough variant of our placement scheme in the 3rd chapter! Although the geometry that the packing lies in determines the details, our placement scheme follows from that given in the proof of Theorem 3.1. We start with two circles, and then place circles which are mutually tangent to any two already placed.

It is convenient to start with the origin as the center of the first circle, and place the center of a tangent neighbor on the positive x-axis. From here, the method of placing remaining circles depends on the space we are working in.

Given two placed tangent circles c_1 and c_2, and a mutual unplaced neighbor c_3, the process for placing c_3 in Euclidean space is intuitive. Form the vector from the center of c_1 to that of c_2 and determine the angle of the face of this triple at c_1: then rotate this vector by the determined angle, and scale the vector to the desired length, $r_1 + r_3$. The endpoint of the vector yields our new circle center.

In hyperbolic space we use a similar method, but in a more roundabout manner. We perform the Möbius transformation that sends the center of c_1 to the origin on both c_1 and c_2. Then we perform the above method to determine a center for c_3, and perform the inverse Möbius transformation on all three circles to send them back to their proper respective locations. (We only relocate these two or three circles at a time instead of moving the entire packing as the latter is likely to introduce unwanted error into our center calculations.)

This is not quite the end of the process for drawing a hyperbolic packing. This gives us the *hyperbolic* centers of the circles in the packing, which generally are different from the Euclidean centers of the circles–and to draw the hyperbolic circle packing, we need a Euclidean radius and a Euclidean center. Suppose we wish to find this information for c. Unlike most hyperbolic isometries, rotation about the origin is a Euclidean isometry. So, an easy way to determine this information is to perform another Möbius transformation on our circle: more

specifically, the one which rotates the circle about the origin until the center of c lies on the x-axis at some coordinate $(x, 0)$. A diameter of our circle then lies on the x-axis, with endpoints $(x - a, 0)$ and $(x + b, 0)$ for some a and b such that $0 < x - a, x + b < 1$. (Recall that the hyperbolic metric causes hyperbolic circles to have centers that generally differ from those of their Euclidean counterparts.) Using the distance metric for hyperbolic space, it is not difficult to determine Euclidean coordinates for these points. From the Euclidean coordinates, a radius and center are easily calculated for the circle lying on the x-axis. Performing a rotation on this coordinate in the reverse direction gives us our desired Euclidean circle center.

The drawing process is fairly simple, but as we know that the labels resulting from our programs are *approximations* to packing labels, there are a few bugs to contend with. The proof of the Monodromy Theorem from which we draw this method rests on the hypothesis that we have a packing label. Although the label generated by the computer program may be close, it will likely be only an ϵ-approximation to the packing label. This means that each circle will likely not be placed in the same location as the packing label would demand, even after taking isometries. This may eventually introduce very visible error in the packing once it is printed, as we place all circles based on the location of previously placed circles.To counter this, we have a very useful strategy: finding a "center" circle of maximum radius to place first.

This works much better than one would suspect. Figure 5.2 was placed using this strategy. Even though we used the same exact approximating label to create the packings in Figure 5.3, those two obviously do not turn out as well!

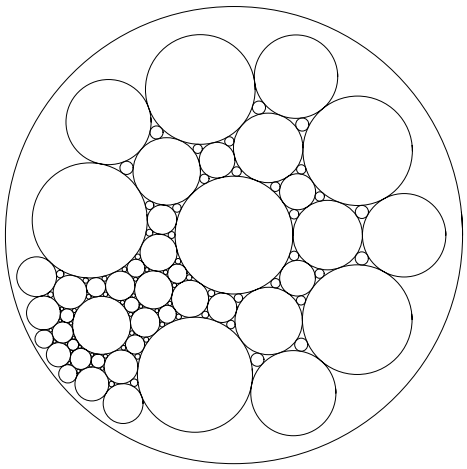

Figure 5.2: The first placed circle was a center with maximum radius.

To explain why center circles place best, we must first develop a notion of what it means to be in the center of a packing. Define a vertex's distance from the boundary to be the fewest number of edges that must be passed through to travel from it to a boundary vertex.

Then, the center circle (or circles) in the packing are those possessing the maximum possible distance from the boundary within the packing. If this distance is n, we must place at least n triples before placing those on the boundary. (The number may obviously be larger if we have more than one "center" circle.) Therefore, we have at least n generations of error built into the packing before we are able to place the outermost circles.

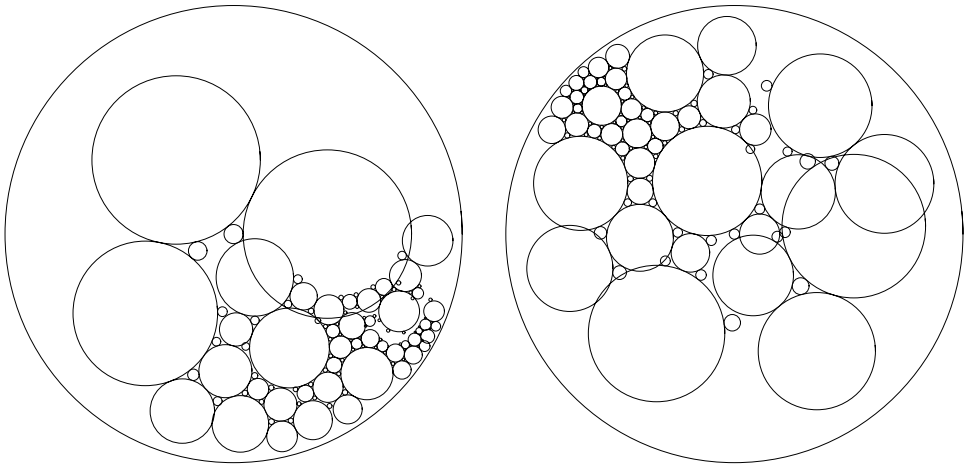

Figure 5.3: On the left, we started with a non-center circle. On the right, we started with a circle of small radius.

If we were to *not* start at the center, then the maximum possible distance between the initial circle and any other is greater than n–thus we have more potential generations of error built into the packing by the time the boundary circles are placed.

To understand why circles with larger radii work better as first placements, keep in mind the accuracy issues discussed earlier. If our radius approximation to a large circle is off by a few decimal places, this does not yield a high percent error for that circle. If our radius approximation to a small circle contains the same inaccuracy, it reflects a much larger percent error–thus smaller circles tend to be unreliable in placement considerations.

What if we wish to not have the recommended starting circle at the origin? This is simple–circle packings are unique up to isometries. Once our placement scheme is done, we may perform rotations, translations, or compositions thereof within our given geometry in order to obtain the desired orientation of our packing.

5.4 Output

The program yields three main forms of output. The first, obviously, is the circle packing.

Continuing our example from earlier, using the information in Table 1 with boundary radii set to 1 yields Figure 5.4. The circles in the packing have been labeled here to correspond

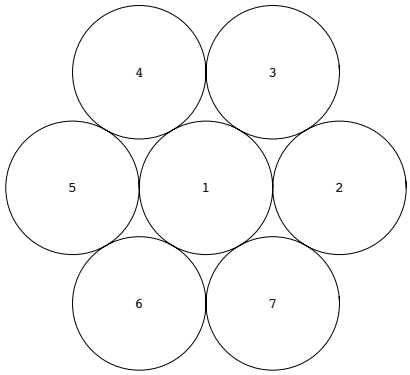

Figure 5.4: An example, continued.

with their respective vertices, but the default in our program is to provide packings without this information. Further output includes the error on iteration file, as in Table 2. Note that it only took two iterations: In this case, all the boundary radii were uniform, so the UNM correction turns out to be the exact answer here! (The error in approximation packings generally ends up as zero only in cases such as this.) On the first iteration, we have *technically* found the proper label, but the over-approximation given for our error forces one extra iteration to check.

Table 2: Error output for our example.

vertex	error upon iteration
1	4.248451723438127309057091826 88
2	0

The last output is the complex K, paired with information on radii, circle centers, and placement. The file appears in the formatting given by Table 3. For formatting concerns, the values in the table are truncated from the actual output of the program.

Table 3: K with label and center information.

vertex	placed	coordinate	location	radius	neighbors
1	1	(0, 0)	1	1.	2 3 4 5 6 7 2
2	1	(2., 0)	0	1.	7 1 3
3	1	(1., 1.732)	0	1.	2 1 4
4	1	(-0.999, 1.732)	0	1.	3 1 5
5	1	(-2., 0)	0	1.	4 1 6
6	1	(-1.000, -1.732)	0	1.	5 1 7
7	1	(0.999, -1.732)	0	1.	6 1 2

Chapter 6

Observations and Conclusions

As can be expected, the observations gathered align with that of Collins and Stephenson in [4]. Acceleration paired with the Uniform Neighbor Model generally produces results with far fewer iterations than the Uniform Neighbor model alone; in some cases yielding a packing after one tenth of the number of iterations. As this often saves several hundred (if not several thousand) iterations, this is a tremendous improvement on UNM as far as computational requirements to get a packing label.

Larger packings (those involving more circles) took much longer to pack than those with fewer circles. Having more vertices in a complex implies more places to gather curvature–but in most cases, simply allowing the program to keep iterating ended in a result. See Tables A.1, A.2, and A.3 for details. These tables are accompanied by several of the packings they refer to in the appdendices.

It also turns out that convergence occurs much faster in the hyperbolic setting than in Euclidean. This aligns with a result given in Theorem 4.1: in Euclidean space, adjustments made by UNM may simply "push curvature around" without reducing it. This cannot happen in hyperbolic space as the geometry is such that proper corrective adjustments will always decrease error. If we are creating a packing for a reason that is not explicitly tied to having specific boundary vertices–for example, if we want to embed a graph and do not care about edge length–this implies that it may be more efficient to work in $\mathbb{D}$ rather than $\mathbb{C}$. Even though the formulas for working in hyperbolic space may be more computationally taxing, the number of iterations required when working in Euclidean space were often two or three times that of running the same complex through a program working in hyperbolic space. Again, consult Tables A.1, A.2, and A.3 for details.

For many of the packings, setting the error tolerance to $\epsilon = 5 \cdot 10^{-15}$ yielded results that were visually passable as a circle packing. Packings containing as few as ≈ 500 circles began to have errors in placement that accumulated mostly near the boundary of the packing, as in Figure 6.1. This aligns with the statements in Section 5.3–that more "generations of error"

have been built into the packing by the time the circles furthest from the first are placed.

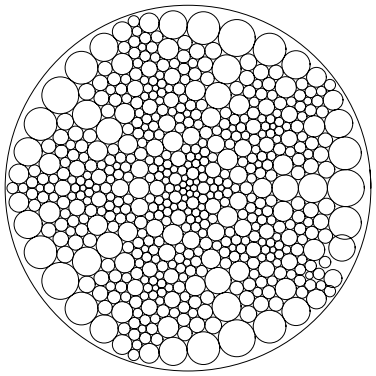

Figure 6.1: The packing accumulates error near the boundary.

A few corrective measures have been suggested in [7] for countering placement error in circle packings. For one, if a circle to be placed has more than one pair of mutually tangent neighbors laid out, we may take the placement of the new circle to be the "average" of the placements suggested by all its neighbors. (In our current program, we cycle through the complex, and all circles are placed based on the first suggested location given.) Another suggestion is to not place circles based on "poorly placed" neighbors. This involves checking the center locations of old circles against new placements in a method similar to the above. Circles which are furthest away from their suggested location may be relocated, or ignored in further placement considerations.

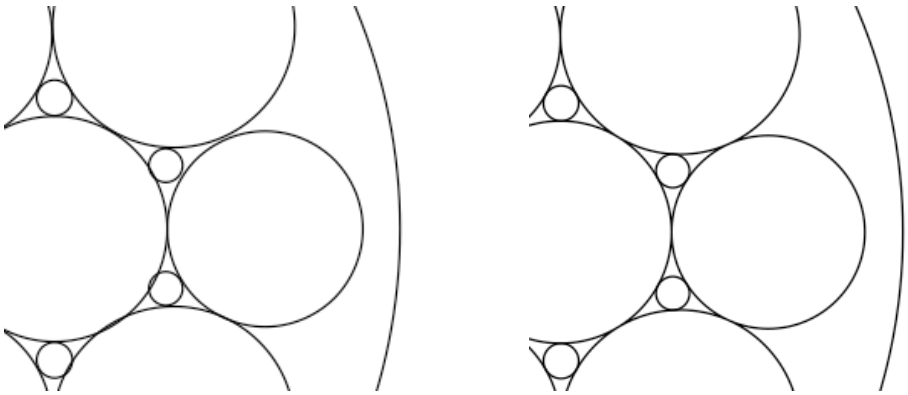

Figure 6.2: Two different computers running the same program on the same file.

Multiple computers were used in collecting output data for these tests. Error manifested itself in different ways dependent upon the particular computer running the programs; even if the same data sets were used to generate output. Although it is not certain, a possible reason for this may be that each machine was running a different version of Mathematica. In Figure 6.2, the output on the left was generated using version 6.0, while the output on the

right was generated using the version the program was written in, Mathematica 5.2. Despite this, the different output files still managed to reflect that error begins to accumulate rapidly as circles are placed further and further away from the initial circle.

Although we have succeeded in developing a rudimentary means of approximating circle packings via the method suggested by Thurston several years ago, much work is yet to be done. Our conclusions on the usefulness of the programs discussed within this work are based on several packings containing a few thousand circles at most–meanwhile, many packings done on an industrial scale may contain several hundred thousand!

The existence and uniqueness criteria upon which our program is based build a strong foundation for future attempts at circle packings. However, recall that some of our methods result from experimental observation of circle packing complexes. Whether the next great improvement in circle packing methods has theoretical or experimental roots may be uncertain–but it is clear that either way, we have much more to learn about creating and manipulating these fascinating structures.

Appendix A

Law of Cosines Derivations

A.1 Euclidean Case

By equation 2.2, we have:

$$\cos\alpha = 1 - \frac{2r_1r_2}{(r+r_1)(r+r_2)}.$$

By the half angle trigonometric identity:

$$\begin{aligned}\sin\frac{\alpha}{2} &= \sqrt{\frac{1-\cos\alpha}{2}},\\ \sin\frac{\alpha}{2} &= \sqrt{\frac{r_1r_2}{(r+r_1)(r+r_2)}}.\end{aligned}$$

Thus we get equation 2.3:

$$\alpha = 2\arcsin\sqrt{\frac{r_1r_2}{(r+r_1)(r+r_2).}}$$

A.2 Hyperbolic Case

We use the identities $\cosh x = \frac{e^x+e^{-x}}{2}$ and $\sinh x = \frac{e^x-e^{-x}}{2}$. Then, by equation 2.10,

$$\alpha = \arccos\frac{(e^{r+r_1}+e^{-r-r_1})(e^{r+r_2}+e^{-r-r_2}) - 2(e^{r_1+r_2}+e^{-r_1-r_2})}{(e^{r+r_1}-e^{-r-r_1})(e^{r+r_2}-e^{-r-r_2})}.$$

Expanding and reducing yields

$$\cos\alpha = 1 - 2e^{-2r}\frac{(e^{-2r_1}-1)(e^{-2r_2}-1)}{(e^{-2r-2r_1}-1)(e^{-2r-2r_2}-1)},$$

and allowing usage of transformed labels yields

$$\cos\alpha = 1 - 2r^* \frac{(r_1^* - 1)(r_2^* - 1)}{(r^* r_1^* - 1)(r^* r_1^* - 1)}.$$

As before, the half angle trigonometric identity provides the rest.

$$\begin{aligned} \sin\frac{\alpha}{2} &= \sqrt{\frac{1-\cos\alpha}{2}}, \\ \sin\frac{\alpha}{2} &= \sqrt{r^* \frac{(r_1^* - 1)(r_2^* - 1)}{(r^* r_1^* - 1)(r^* r_1^* - 1)}}. \end{aligned}$$

Thus, we have equation 2.11.

$$\alpha = 2\arcsin\sqrt{\frac{r^*(1-r_1^*)(1-r_2^*)}{(1-r^* r_1^*)(1-r^* r_2^*)}}$$

Appendix B

Uniform Neighbor Model

As before, let k be the number of petal circles to v, θ the current angle sum, A the target packing condition at v, $\beta = \sin\frac{\theta}{2k}$, and $\delta = \sin\frac{A}{2k}$.

Assume the angle sum of the left flower in Figure B.1 matches that of θ. If we are attempting to solve for $\hat{r}$, note that $\beta = \sin\frac{\theta}{2k}$ may also be given in the Euclidean case by $\frac{\hat{r}}{\hat{r}+r}$, as the half face shown is a right triangle. Substituting this for β in $\frac{\beta}{1-\beta}r$ and reducing then yields $\hat{r}$. As for the hyperbolic case, equation 2.11 gives $\beta = \sqrt{\frac{r(1-\hat{r})^2}{(1-r\hat{r})^2}}$ if we take r and $\hat{r}$ to be transformed hyperbolic radii. Similarly, substituting this into $\frac{\beta-\sqrt{r}}{\beta r-\sqrt{r}}$ and reducing yields the transformed hyperbolic radius $\hat{r}$.

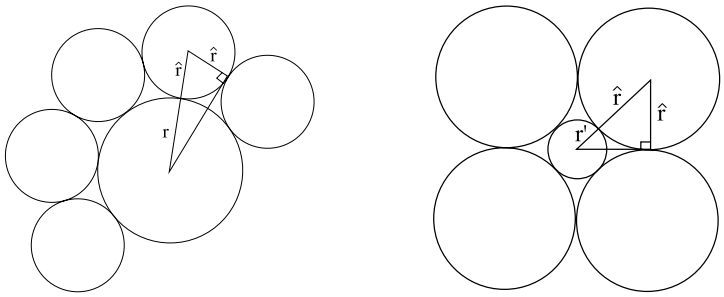

Figure B.1: Determining $\hat{r}$ and r'.

Assume the angle sum of the right flower in Figure B.1 matches that of our target packing condition, A. If we are attempting to solve for r', note that $\delta = \sin\frac{A}{2k}$ may also be given in the Euclidean case by $\frac{\hat{r}}{\hat{r}+r'}$, as the half face shown is again a right triangle. Substituting this for δ in $\frac{1-\delta}{\delta}\hat{r}$ and reducing then yields r'. As for the hyperbolic case, equation 2.11 again gives $\delta = \sqrt{\frac{r'(1-\hat{r})^2}{(1-r'\hat{r})^2}}$ for r' and $\hat{r}$ transformed hyperbolic radii. Substituting this into $(\frac{2\delta}{\sqrt{(1-\hat{r})^2+4\delta\hat{r}}+1-\hat{r}})^2$ and reducing yields the transformed hyperbolic radius r'.

Appendix C

Programs

C.1 Euclidean UNM with Acceleration

The following is Mathematica code used in testing Euclidean UNM with acceleration. Some comments have been left in; they appear (*like this*). The program has been written such that if one wishes to perform a test *without* acceleration or super acceleration, the appropriate sections of code can be removed or commented out. Comments on which sections to remove are included in the code.

The following should be compatible with Mathematica version 5.2.

```
Clear[accur,L,K,Epsilon,RadiiPack,x];
accur=30;
errorfilename="Desktop/error.txt";
outputfilename="Desktop/output7.txt";
packingfilename="Desktop/packing7.pdf";
K=Import["/Users/username/Desktop/input.txt","Table"];
(*location of the input file*)

BoundVert={};
InteriorVert={};
For[m=1,m<=Length[K],
If[K[[m,3]]==K[[m,Length[K[[m]]]]],AppendTo[InteriorVert,K[[m,1]]];m++,
AppendTo[BoundVert,K[[m,1]]];m++]];
(*the above sorts the boundary from interior vertices*)
L=Table[SetPrecision[1.,accur],{Length[BoundVert]}];
(*here I've set the boundary vertices to all have radius one.*)
K=ReplacePart[K,SetPrecision[.3,accur],Table[{n,2},{n,1,Length[K]}]];
(*a dummy value of .3 is set for all interior radii*)
```

```
For[b=1,b<=Length[BoundVert],
K=ReplacePart[K,L[[b]],{BoundVert[[b]],2}];
K=Insert[K,0,{BoundVert[[b]],2}];b++];
(*this marks each boundary vertex with 0*)
For[i=1,i<=Length[InteriorVert],
K=Insert[K,-1,{InteriorVert[[i]],2}];i++];
(*each interior vertex is marked with -1*)

Epsilon=SetPrecision[.000000000000005,accur];
Delta=SetPrecision[.05,accur];
(*here I've set epsilon and delta*)
ItCount=0;
ErrorPoints={};
AccelCount=0;
SuperAccelCount=0;
ErrorSum=Epsilon+1;
Lambda=-1;
Flag=0;
While[ErrorSum>=Epsilon,
PreErrorSum=ErrorSum;
PreLambda=Lambda;
PreLabel=Table[K[[v,3]],{v,1,Length[K]}];
PreFlag=Flag;
(*These all give ErrorSum, Lambda, and the list of radii from the
   last pass.*)
ErrorSum=SetPrecision[0.0,accur];
For[m=1,m<=Length[K],
If[K[[m,2]]==-1,
AngleSum=SetPrecision[0.0,accur];
r=K[[m,3]];
For[n=4,n<Length[K[[m]]],
r1=K[[K[[m,n]],3]];
r2=K[[K[[m,n+1]],3]];
AngleSum=SetPrecision[(AngleSum+
   ArcCos[((r+r1)^2+(r+r2)^2-(r1+r2)^2)/(2*(r+r1)*(r+r2))]),accur];
n++];
ErrorSum=SetPrecision[(ErrorSum+(2*\[Pi]-AngleSum)^2),accur];
PetalNumber=(Length[K[[m]]]-4);
a=Sin[Pi/PetalNumber];
b=Sin[AngleSum/(2*PetalNumber)];
K=ReplacePart[K,SetPrecision[(K[[m,3]]*(b-a*b))/(a-b*a),accur],{m,3}];
m++,
```

```
m++]];
ErrorSum=SetPrecision[ErrorSum^(1/2),accur];
 Lambda=SetPrecision[ErrorSum/PreErrorSum,accur];
Flag=1;
(*Starting here is code for acceleration:*)
(****************************************)
If[PreFlag==1&&Lambda<1,
ErrorSum=ErrorSum*Lambda;

(*Starting here is super acceleration:*)
(*----------------------------------*)
If[Abs[Lambda-PreLambda]<Delta,
Lambda=SetPrecision[Lambda/(1-Lambda),accur];
SuperAccelCount++];
(*----------------------------------*)

LambdaList1={};
LambdaList2={};
For[v=1,v<=Length[K],
If[K[[v,2]]==-1&&K[[v,3]]!=PreLabel[[v]],
AppendTo[LambdaList1,Max[SetPrecision[-K[[v,3]]/(K[[v,3]]-
   PreLabel[[v]]),accur],SetPrecision[(1-K[[v,3]])/(K[[v,3]]-
   PreLabel[[v]]),accur]]];
AppendTo[LambdaList2,Min[SetPrecision[-K[[v,3]]/(K[[v,3]]-
   PreLabel[[v]]),accur],SetPrecision[(1-K[[v,3]])/(K[[v,3]]-
   PreLabel[[v]]),accur]]]];
v++];
LambdaMax=Min[LambdaList1];
LambdaMin=Max[LambdaList2];

If[LambdaMax>LambdaMin,
Lambda=Min[Lambda,LambdaMax/2];
For[v=1,v<=Length[K],
If[K[[v,2]]!=0,
K[[v,3]]=K[[v,3]]+Lambda*(K[[v,3]]-PreLabel[[v]])];
v++];
AccelCount++]];
(****************************************)
(*Acceleration code ends here.*)

ItCount++;
AppendTo[ErrorPoints,{ItCount,ErrorSum}]];
```

```
(*this collects and writes the  error on each iteration to the set
"Errorpoints".*)

Print["Error = ",N[ErrorSum]];
Print["Iteration Count = ",ItCount];
(*this prints the iterations/error after the algorithm is done.*)
Export[errorfilename,ErrorPoints,"Table"];
(*this sends the error on each iteration data to a file.*)
Show[Graphics[Point/@ErrorPoints],
PlotRange->All,AspectRatio->1,Axes->True,PlotLabel->
   "Error on subsequent iterations"];
(*the above yields a graph of iteration count/error.*)


K=Insert[K,0,Table[{x,2},{x,1,Length[K]}]];
K=Insert[K,0,Table[{x,3},{x,1,Length[K]}]];
(*after this step:
-the first value in each sublist of K is the vertex number
-the second value denotes whether placement has occured
-the third value should be a coordinate
-the fourth value denotes interior or exterior
-fifth denotes radius at the given vertex
-sixth and on denote which vertices are neighbors to this vertex*)

For[v=1,Intersection[Table[K[[x,4]],{x,1,Length[K]}],{-1}]!={},
TestSet=Pick[Table[x,{x,1,Length[K]}],Table[K[[x,4]],{x,1,Length[K]}],-1];
(*this lists untested interior vertices*)
BoundSet=Pick[Table[x,{x,1,Length[K]}],Table[K[[x,4]],{x,1,Length[K]}],v-1];
(*this lists vertices v combinatorial generations away from the closest
   boundary vertex*)
For[m=1,m<=Length[TestSet],
If[Intersection[Take[K[[TestSet[[m]]]],{6,Length[K[[TestSet[[m]]]]]}],
   BoundSet]!={},
K[[TestSet[[m]],4]]=v;];
m++];
v++];
CenterVertices=Pick[Table[x,{x,1,Length[K]}],Table[K[[x,4]],
   {x,1,Length[K]}],v-1];
CenterVertex=First[Pick[CenterVertices,Table[K[[CenterVertices[[x]],5]],
   {x,1,Length[CenterVertices]}],Max[Table[K[[CenterVertices[[x]],5]],
   {x,1,Length[CenterVertices]}]]]];
(*this finds a set of "center" vertices and selects the element with the
```

```
   largest radii among this list to be the center vertex.*)

K[[CenterVertex,3]]={0,0};K[[CenterVertex,2]]=1;
(*this assigns the center vertex the origin as a center*)
K[[K[[CenterVertex,6]],3]]={(K[[CenterVertex,5]]+
   K[[K[[CenterVertex,6]],5]]),0};
K[[K[[CenterVertex,6]],2]]=1;
(*this assigns the second circle a spot on the x-axis*)
l=Length[K];

While[Table[K[[m,2]],{m,1,l}]!=Table[1,{m,1,l}],
For[m=1,m<=l,
If[K[[m,2]]==1,
For[n=6,n<Length[K[[m]]],
If[(K[[K[[m,n]],2]]==1)&&(K[[K[[m,n+1]],2]]==0),
r=K[[m,5]];
r1=K[[K[[m,n]],5]];
r2=K[[K[[m,n+1]],5]];
costheta=((r+r1)^2+(r+r2)^2-(r1+r2)^2)/(2*(r+r1)*(r+r2));
x=-K[[m,3,1]]+K[[K[[m,n]],3,1]];
y=-K[[m,3,2]]+K[[K[[m,n]],3,2]];
K[[K[[m,n+1]],3]]={x*costheta-y*(1-costheta^2)^(1/2),
   x*(1-costheta^2)^(1/2)+y*costheta}*(r+r2)/(x^2+y^2)^(1/2)+K[[m,3]];
K[[K[[m,n+1]],2]]=1];
n++]];
m++]];
(*the above loop assigns the centers of all other vertices*)


Export[outputfilename,Prepend[K,
   {vert,placed,coord,int,radius,neighbors}],"Table"]
(*this exports K to a file.*)
(*Show[Graphics[Point/@Table[K[[x,3]],{x,1,Length[K]}]],
   PlotRange->All,AspectRatio->1]*)
(*this shows the circle centers; it is currently commented out*)

Export[packingfilename,Show[Graphics[Table[Circle[{K[[t,3,1]],
   K[[t,3,2]]},K[[t,5]]],{t,1,l}],AspectRatio->Automatic]]];
(*The above command exports a graphic of the packing to pdf. It should
   output the packing in mathematica as well.*)
```

C.2 Hyperbolic UNM with Acceleration

The following is Mathematica code used in testing Hyperbolic UNM with acceleration. Some comments have been left in; yet again they appear (*like this*). This program has also been written such that if one wishes to perform a test *without* acceleration or super acceleration, the appropriate sections of code can be removed or commented out. Comments on which sections to remove are included in the code.

The following should also be compatible with Mathematica version 5.2.

```
LT[x_]:=E^(-2*x);
Clear[accur,L,K,Epsilon,RadiiPack,x];
accur=30;
errorfilename="Desktop/error.txt";
outputfilename="Desktop/output7.txt";
packingfilename="Desktop/packing7.pdf";
K=Import["/Users/username/Desktop/input.txt","Table"];
(*location of the input file*)

BoundVert={};
InteriorVert={};
For[m=1,m<=Length[K],
If[K[[m,3]]==K[[m,Length[K[[m]]]]],AppendTo[InteriorVert,K[[m,1]]];m++,
AppendTo[BoundVert,K[[m,1]]];m++]];
(*the above sorts the boundary from interior vertices*)
L=Table[SetPrecision[1.,accur],{Length[BoundVert]}];
(*here I've set the boundary vertices to all have radius one.*)
K=ReplacePart[K,SetPrecision[.3,accur],Table[{n,2},{n,1,Length[K]}]];
(*this sets a dummy value of .3 for all interior radii*)
For[b=1,b<=Length[BoundVert],
K=ReplacePart[K,Exp[-2*L[[b]]],{BoundVert[[b]],2}];
K=Insert[K,0,{BoundVert[[b]],2}];b++];
(*this replaces the radii w/ a transformed value of what was assigned in L
   and marks each boundary vertex with 0*)
For[i=1,i<=Length[InteriorVert],
K=Insert[K,-1,{InteriorVert[[i]],2}];i++];
(*this marks each interior vertex with -1*)

Epsilon=SetPrecision[.000000000000005,accur];
Delta=SetPrecision[.05,accur];
(*here I've set epsilon and delta*)
ItCount=0;
```

```
ErrorPoints={};
AccelCount=0;
SuperAccelCount=0;
ErrorSum=Epsilon+1;
Lambda=-1;
Flag=0;
While[ErrorSum>=Epsilon,
PreErrorSum=ErrorSum;
PreLambda=Lambda;
PreLabel=Table[K[[v,3]],{v,1,Length[K]}];
PreFlag=Flag;
(*These all give ErrorSum, Lambda, and the list of radii from the last
   pass.*)
ErrorSum=SetPrecision[0.0,accur];
For[m=1,m<=Length[K],
If[K[[m,2]]==-1,
AngleSum=SetPrecision[0.0,accur];
r=K[[m,3]];
For[n=4,n<Length[K[[m]]],
r1=K[[K[[m,n]],3]];
r2=K[[K[[m,n+1]],3]];
AngleSum=SetPrecision[(AngleSum+2*ArcSin[((r*(1-r1)*(1-r2))/
   ((1-r*r1)*(1-r*r2)))^(1/2)]),accur];
n++];
ErrorSum=SetPrecision[(ErrorSum+(2*Pi-AngleSum)^2),
   accur];

PetalNumber=(Length[K[[m]]]-4);
delt=Sin[Pi/PetalNumber];
beta=Sin[AngleSum/(2*PetalNumber)];
rhat=Max[(beta-r^(1/2))/(beta*r-r^(1/2)),0];
rnew=SetPrecision[((2*delt)/(((1-rhat)^2+4*delt^2*rhat)^(1/2)+
   (1-rhat)))^2,accur];
K=ReplacePart[K,rnew,{m,3}]];
m++];
ErrorSum=SetPrecision[ErrorSum^(1/2),accur];
 Lambda=SetPrecision[ErrorSum/PreErrorSum,accur];
Flag=1;

(*Starting here is code for acceleration:*)
(**************************************)
If[PreFlag==1&&Lambda<1,
```

```
ErrorSum=ErrorSum*Lambda;

(*Starting here is super acceleration:*)
(*-----------------------------------*)
If[Abs[Lambda-PreLambda]<Delta,Lambda=SetPrecision[Lambda/(1-Lambda)accur];
SuperAccelCount++];
(*-----------------------------------*)
(*Super acceleration code ends here.*)

LambdaList1={};
LambdaList2={};
For[v=1,v<=Length[K],
If[K[[v,2]]==-1&&K[[v,3]]!=PreLabel[[v]],
AppendTo[LambdaList1,Max[SetPrecision[-K[[v,3]]/(K[[v,3]]-
   PreLabel[[v]]),accur],SetPrecision[(1-K[[v,3]])/(K[[v,3]]-
   PreLabel[[v]]),accur]]];
AppendTo[LambdaList2,Min[SetPrecision[-K[[v,3]]/(K[[v,3]]-
   PreLabel[[v]]),accur],SetPrecision[(1-K[[v,3]])/(K[[v,3]]-
   PreLabel[[v]]),accur]]]];
v++];
LambdaMax=Min[LambdaList1];
LambdaMin=Max[LambdaList2];

If[LambdaMax>LambdaMin,
Lambda=Min[Lambda,LambdaMax/2];
For[v=1,v<=Length[K],
If[K[[v,2]]!=0,
K[[v,3]]=K[[v,3]]+Lambda*(K[[v,3]]-PreLabel[[v]])];
v++];
AccelCount++]];
(**************************************)
(*Acceleration code ends here.*)

ItCount++;
AppendTo[ErrorPoints,{ItCount,ErrorSum}]];
(*this collects the error on each iteration, and adds it to the set
   "Errorpoints".*)

Print["Error = ",N[ErrorSum]];
Print["Iteration Count = ",ItCount];
Print["Acceleration Count = ",AccelCount];
Print["Super Accel. Count = ",SuperAccelCount]
```

```
(*this prints the iterations/error etc. after the algorithm is done.*)
Export[errorfilename,ErrorPoints,"Table"];
(*this sends the error on each iteration data to a file.*)
Show[Graphics[Point/@ErrorPoints],
PlotRange->All,AspectRatio->1,Axes->True,PlotLabel->"Error on
   subsequent iterations"];
(*the above yields a graph of iteration count/error.*)

K=Insert[K,0,Table[{x,2},{x,1,Length[K]}]];
K=Insert[K,0,Table[{x,3},{x,1,Length[K]}]];
(*after this step:
-the first value in each sublist of K is the vertex number
-the second value denotes whether placement has occured
-the third value should be a coordinate
-the fourth value denotes interior or exterior
-fifth denotes radius at the given vertex
-sixth and on denote which vertices are neighbors to this vertex*)

For[d=1,d<=Length[K],
K=ReplacePart[K,-.5Log[K[[d,5]]],{d,5}];
d++];
(*this replaces the transformed labels of the form e^-2h with h, the actual
   hyperbolic radius*)

For[v=1,Intersection[Table[K[[x,4]],{x,1,Length[K]}],{-1}]!={},
TestSet=Pick[Table[x,{x,1,Length[K]}],Table[K[[x,4]],{x,1,Length[K]}],-1];
(*this lists untested interior vertices*)
BoundSet=Pick[Table[x,{x,1,Length[K]}],Table[K[[x,4]],{x,1,Length[K]}],v-1];
(*this lists vertices v combinatorial generations away from the closest
   boundary vertex*)

For[m=1,m<=Length[TestSet],
If[Intersection[Take[K[[TestSet[[m]]]],{6,Length[K[[TestSet[[m]]]]]}],
   BoundSet]!={},
K[[TestSet[[m]],4]]=v;];
m++];
v++];
CenterVertices=Pick[Table[x,{x,1,Length[K]}],Table[K[[x,4]],
   {x,1,Length[K]}],v-1];
CenterVertex=First[Pick[CenterVertices,Table[K[[CenterVertices[[x]],5]],
   {x,1,Length[CenterVertices]}],Max[Table[K[[CenterVertices[[x]],5]],
   {x,1,Length[CenterVertices]}]]];
```

```
(*this finds a set of "center" vertices and selects the element with the
   largest radii among this list to be the center vertex.*)

K[[CenterVertex,3]]={0,0};K[[CenterVertex,2]]=1;
(*this assigns the center vertex the origin as a center*)
K[[K[[CenterVertex,6]],3]]={1-2/(1+Exp[K[[CenterVertex,5]]
   +K[[K[[CenterVertex,6]],5]]]),0};
K[[K[[CenterVertex,6]],2]]=1;
(*this assigns the second circle a spot on the x-axis*)
l=Length[K];

MobiusTransform[Z_,c_]:={Re[(Z[[1]]+I*Z[[2]]-c)/(1-Conjugate[c]*(Z[[1]]+
   I*Z[[2]]))],Im[(Z[[1]]+I*Z[[2]]-c)/(1-Conjugate[c]*(Z[[1]]+I*Z[[2]]))]};
(*this defines the Mobius transformation taking the point c to the origin*)
While[Table[K[[m,2]],{m,1,l}]!=Table[1,{m,1,l}],
For[m=1,m<=l,
If[K[[m,2]]==1,
For[n=6,n<Length[K[[m]]],
If[(K[[K[[m,n]],2]]==1)&&(K[[K[[m,n+1]],2]]==0),

PlaceMe=K[[m,n+1]];
For[f=6,f<Length[K[[PlaceMe]]],

If[(K[[K[[PlaceMe,f]],2]]==1)&&(K[[K[[PlaceMe,f+1]],2]]==1),
c0=K[[K[[PlaceMe,f]],3,1]]+I*K[[K[[PlaceMe,f]],3,2]];
r=K[[K[[PlaceMe,f]],5]];
r1=K[[K[[PlaceMe,f+1]],5]];
r2=K[[PlaceMe,5]];
halfsine=((LT[r]*(1-LT[r1])*(1-LT[r2]))/
   ((1-LT[r]*LT[r1])*(1-LT[r]*LT[r2])))^(1/2);
sintheta=2*halfsine*(1-halfsine^2)^.5;

K[[K[[m,n+1]],3]]=MobiusTransform[{{(1-sintheta^2)^(1/2),sintheta},
   {-sintheta,(1-sintheta^2)^(1/2)}}.
   MobiusTransform[K[[K[[PlaceMe,f+1]],3]],c0]*(Exp[r+r2]-1)/
   (Norm[MobiusTransform[K[[K[[PlaceMe,f+1]],3]],c0]]*(Exp[r+r2]+1)),-c0];
K[[K[[m,n+1]],2]]=1;
f=Length[K[[PlaceMe]]],
f++]]];
n++]];
m++]];
Export[outputfilename,Prepend[K,
```

```
   {vert,placed,coord,int,radius,neighbors}],"Table"]
(*this exports K, radii, and circle centers for a packing approximation to
   a file.*)

(*Show[Graphics[Point/@Table[K[[x,3]],{x,1,Length[K]}]],
Graphics[Circle[{0,0},1]],
PlotRange->All,AspectRatio->1]*)
(*this shows the original hyperbolic centers; it is currently commented
   out*)
(*Show[Graphics[
{Table[Point[K[[t,3]]*2/(Norm[K[[t,3]]]^2+1+
   (1-Norm[K[[t,3]]]^2)*Cosh[K[[t,5]]])],{t,1,Length[K]}],
Circle[{0,0},1]},
PlotRange->All,AspectRatio->Automatic]];*)
(*this shows the euclidean centers--calculation of euclidean centers is
   entirely contained w/in this graphics command. the entire cell is
   currently commented out*)
Export[packingfilename,Show[Graphics[{Table[Circle[K[[t,3]]*2/
   (Norm[K[[t,3]]]^2+1+(1-Norm[K[[t,3]]]^2)*Cosh[K[[t,5]]]),
   ((1-Norm[K[[t,3]]]^2)*Sinh[K[[t,5]]])/(Norm[K[[t,3]]]^2+1+
   (1-Norm[K[[t,3]]]^2)*Cosh[K[[t,5]]])],{t,1,Length[K]}],
Circle[{0,0},1]},
PlotRange->All,AspectRatio->Automatic]]];
(*The above command exports a graphic of the packing to pdf. It should
   output the packing in mathematica as well.*)
```

Appendix D

Convergence Tables and Packings

The following are hyperbolic packings and tables of convergence for pentagonal, fracthex, and trhex subdivision rules. "Level" in the tables refers to iterations of subdivisions according to the respective rule.

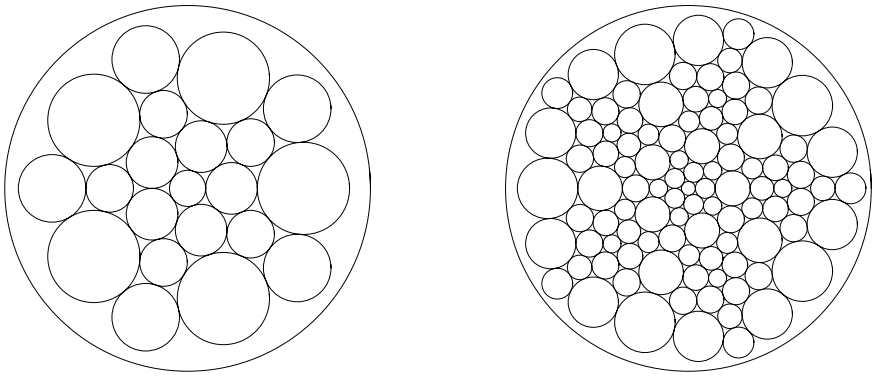

Figure D.1: Pentagonal subdivision packings at level 1 and 2.

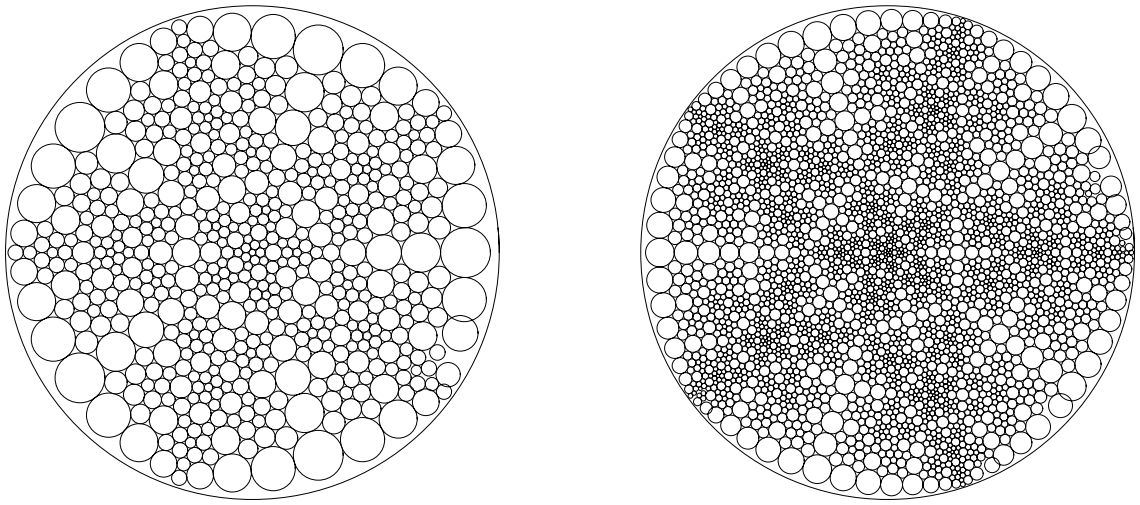

Figure D.2: Pentagonal subdivision packings at level 3 and 4.

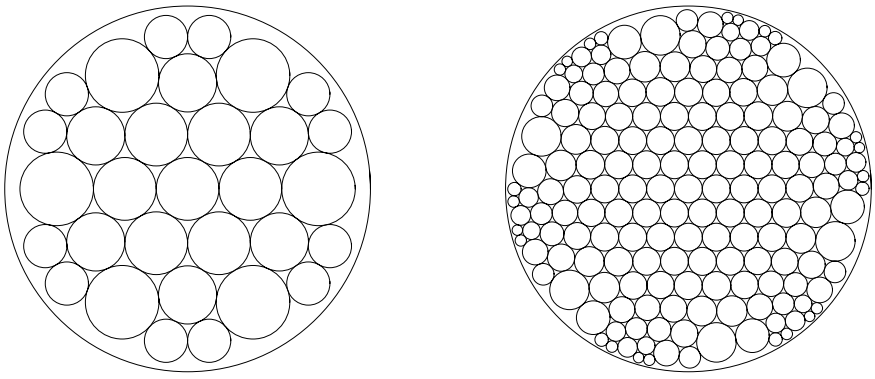

Figure D.3: Fracthex subdivision packings at level 1 and 2.

Table A.1: Pentagonal subdivision iteration counts

Level	Geometry	Number of circles	UNM iterations	Acceleration iterations
0	hyperbolic	6	2	2
1	hyperbolic	21	44	24
2	hyperbolic	101	190	60
3	hyperbolic	561	928	117
4	hyperbolic	3281	4934	302
0	Euclidean	6	2	2
1	Euclidean	21	66	36
2	Euclidean	101	391	92
3	Euclidean	561	2278	265
4	Euclidean	3281	13231	994

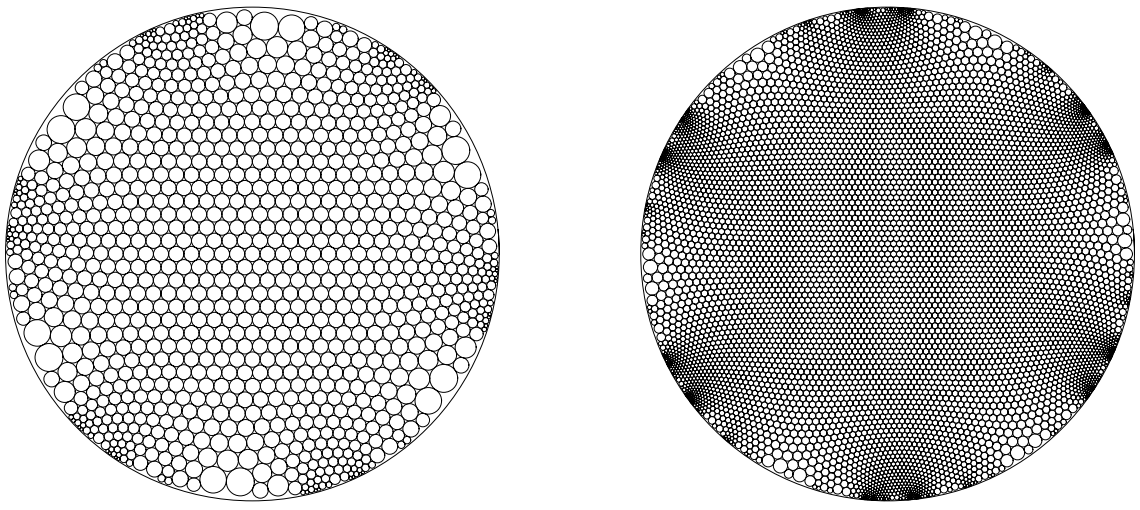

Figure D.4: Fracthex subdivision packings at level 3 and 4.

Table A.2: Fracthex subdivision iteration counts

Level	Geometry	Number of circles	UNM iterations	Acceleration iterations
0	hyperbolic	7	2	2
1	hyperbolic	31	32	21
2	hyperbolic	175	155	68
3	hyperbolic	1111	908	168
4	hyperbolic	7447	5823	573
0	Euclidean	7	2	2
1	Euclidean	31	57	36
2	Euclidean	175	408	146
3	Euclidean	1111	2798	523
4	Euclidean	7447	19075	2358

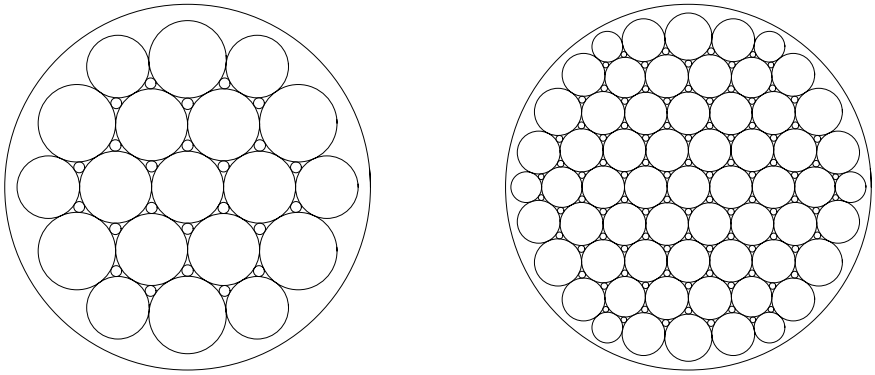

Figure D.5: Trhex subdivision packings at level 1 and 2.

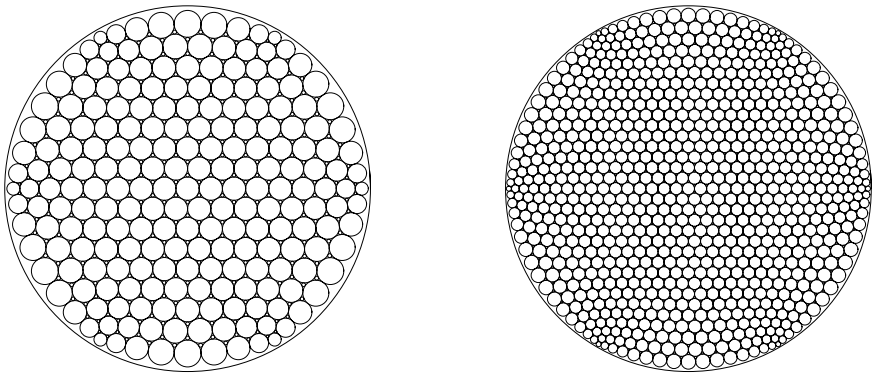

Figure D.6: Trhex subdivision packings at level 3 and 4.

Table A.3: Trhex subdivision iteration counts

Level	Geometry	Number of circles	UNM iterations	Acceleration iterations
0	hyperbolic	13	23	15
1	hyperbolic	43	53	22
2	hyperbolic	157	152	39
3	hyperbolic	601	496	91
4	hyperbolic	2353	1770	293
0	Euclidean	13	29	20
1	Euclidean	43	94	52
2	Euclidean	157	359	120
3	Euclidean	601	1400	243
4	Euclidean	2353	5477	828

Bibliography

[1] A. Beardon, *The Geometry of Discrete Groups*, Springer-Verlag, New York, 1983.

[2] J. W. Cannon, W. J. Floyd, and W. R. Parry, subdivide.c, software, available from http://www.math.vt.edu/people/floyd/research/software/subdiv.html

[3] J. W. Cannon, W. J. Floyd, and W. R. Parry, tilepack.c, software, available from http://www.math.vt.edu/people/floyd/research/software/subdiv.html

[4] C. Collins and K. Stephenson, *A Circle Packing Algorithm*, Computational Geometry: Theory and Applications, 25 (2003), 233-256 (electronic).

[5] K. Stephenson et. al., CirclePack, software, available from http://www.math.utk.edu/∼kens/CirclePack

[6] K. Stephenson, *Circle Packings: Existence and Uniqueness*, Course lecture notes, University of Tennessee, 1995.

[7] K. Stephenson, *Introduction to Circle Packings*, Cambridge University Press, New York, 2005.