Algorithms for Congruent Sphere Packing and Applications

Danny Z. Chen[*], Xiaobo (Sharon) Hu[†], Yingping Huang[*], Yifan Li[*]
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
{chen, shu, yhuang3, yli3}@cse.nd.edu

Jinhui Xu[‡]
Department of Computer Science and Engineering
State University of New York at Buffalo
201 Bell Hall
Buffalo, NY 14260, USA
jinhui@cse.buffalo.edu

ABSTRACT

The problem of packing congruent spheres (i.e., copies of the same sphere) in a bounded domain arises in many applications. In this paper, we present a new pack-and-shake scheme for packing congruent spheres in various bounded 2-D domains. Our packing scheme is based on a number of interesting ideas, such as a trimming and packing approach, optimal lattice packing under translation and/or rotation, shaking procedures, etc. Our packing algorithms have fairly low time complexities. In certain cases, they even run in nearly linear time. Our techniques can be easily generalized to congruent packing of other shapes of objects, and are readily extended to higher dimensional spaces. Applications of our packing algorithms to treatment planning of radiosurgery are discussed. Experimental results suggest that our algorithms produce reasonably dense packings.

1. INTRODUCTION

In this paper, we consider the following sphere packing problem: Given a polygonal (or polyhedral) region R (called the *container* or *domain*) in two (or higher) dimensional space and an infinite *object set* $\mathcal{O}$ of "solid" unit spheres, find a sphere packing SP for R using the spheres in $\mathcal{O}$ such that (i) each sphere in SP is inside R, (ii) no two spheres in SP intersect each other in their interior, and (iii) the volume of R covered by SP (called the *density*) is maximized.

Packing is a venerable topic in mathematics. Various versions of packing problems have been studied [16], depending on the shapes of the domains, the types of objects, the position restrictions on the objects, the optimization criteria, etc. Originating from number theory and crystallography, sphere packing has mysterious connections with hyperbolic geometry, Lie algebras, and the Monster simple group, and finds direct applications in number theory and pure geometry [16]. It also arises in numerous applied areas such as digital communications, cryptography, numerical evaluation of integrals, physics, chemistry, biology, antenna design, X-ray tomography, and statistical analysis on spheres [16]. Packing problems in lower dimensions d-D ($d \leq 3$) with a bounded or unbounded domain R appear in manufacturing [4, 5, 6, 13, 15, 17, 18, 19, 20, 29, 30, 34, 44, 48, 49, 50, 51, 59] (e.g., stock and cloth cutting, part nesting, compaction, and containment), mesh generation [9, 11, 12, 43, 52, 57, 58], VLSI layout [36], logistics [29], scheduling [4, 5, 6, 13, 30, 38, 42, 59], management [20], operations research, and image processing [8, 36].

Most packing problems exhibit substantial difficulties. Even very restricted versions (e.g., regular-shaped objects and domains in lower dimensional spaces) have been proved to be NP-hard. The 2-D problem of packing arbitrary-shaped objects in a bounded domain [49] has been shown to have very high time complexity (e.g., exponential in the size of the packing). For *congruent packing* (i.e., packing copies of the same object), it is known [26] that the 2-D cases of packing fixed-sized squares or disks in a simple polygon are NP-hard. Recently, Baur and Fekete [8] considered a closely related dispersion problem: Pack k congruent disks in a polygon such that the radius of disks is maximized. They proved that the dispersion problem cannot be approximated arbitrarily well in polynomial time unless P = NP, and gave a

[*]This research was supported in part by the National Science Foundation under Grants CCR-9623585 and CCR-9988468.

[†]This research was supported in part by the National Science Foundation under Grants MIP-9701416 and CCR-9988468, and by HP Labs, Bristol, England, under an external research program grant.

[‡]Corresponding author. This research was supported in part by a faculty start-up fund from the CSE Dept., SUNY at Buffalo.

$\frac{2}{3}$-approximation algorithm for the L_∞ disk case with a time bound of $O(n^{38})$.

The sphere packing problem we study is motivated by medical applications in radiosurgery. Radiosurgery is a minimally invasive surgical procedure that uses radiation to destroy tumors inside human body. Gamma Knife is a radiation system that contains 201 Cobalt-60 sources [10, 63]. The gamma-rays from these sources are all focused on a common center, creating a spherical volume of high radiation dose. A key geometric problem in Gamma Knife treatment planning is to fit multiple balls into a 3-D irregular-shaped tumor [10, 61, 63]. In this situation, overlapping balls may cause overdose, and a low packing density may result in underdose and a non-uniform dose distribution. Note that Gamma Knife currently produces spheres of four different radii (4mm, 8mm, 14mm, and 18mm), and hence the Gamma Knife sphere packing is in general not congruent. However, in practice, a commonly used approach is to pack larger spheres first, and then fit smaller spheres into the remaining subdomains, in the hope of reducing the total number of spheres involved and thus shortening the treatment time. Therefore, congruent sphere packing can be used as a key subroutine for such a common approach.

Much work on congruent sphere packing studies the case of packing spheres into an unbounded domain or even the whole space [16] (e.g., Mount and Silverman's algorithm [54]). There are also some results on packing congruent spheres into a bounded region. Hochbaum and Maass [36] presented a unified and powerful *shifting technique* for designing pseudo-polynomial time approximation schemes for packing congruent squares into a rectilinear polygon; but, the high time complexities (e.g., $O(n^{38})$ in [8]) associated with the resulting algorithms restrict its applicability in practice. Graham and Lubachevsky [31, 32] considered the problems of packing k L_2 disks into an equilateral triangle or square to maximize the radius of disks, producing a number of best known packings for different constants k. Friedman [27] obtained many best known solutions for packing k unit-squares into the smallest square. A common feature of this type of algorithms is to transform a packing problem into some non-linear optimization problems, and resort to available optimization software to generate packings. In general, such an approach guarantees neither a fast running time nor a provably good quality of solutions, and it works well only for small problem sizes and regular-shaped domains.

To reduce the running time yet achieve a dense packing, a common idea is to let the objects form a certain lattice or double-lattice. A number of results were given on lattice packing of congruent objects in the whole (especially high dimensional) space [16]. For a bounded rectangular 2-D domain, Milenkovic [51] adopted a method that first finds the densest translational lattice packing for a set of polygonal objects in the whole plane, and then uses some heuristics to extract the actual packing.

In this paper, we present a new scheme, called *pack-and-shake*, for packing congruent spheres in an irregular-shaped 2-D or 3-D bounded domain. Our scheme consists of three phases. In the first phase, the d-D ($d = 2, 3$) irregular-shaped domain R is partitioned into some convex cells. The set of cells defines a dual graph G_D (each vertex v of G_D is for a cell $C(v)$, and an edge connects two vertices if their cells share a $(d-1)$-D face). In the second phase, the algorithm repeats the following *trimming and packing* process until $G_D = \emptyset$: Remove the lowest degree vertex v from G_D and pack the cell $C(v)$. In the third phase, a *shake* procedure is applied to globally adjust the packing to obtain a denser one.

The objective of our trimming and packing procedure is that after each cell is packed, the remaining "packable" subdomain R' of R is always kept as a connected region. The rationale for maintaining the connectivity of R' is as follows. To pack spheres in a bounded domain R, two typical approaches have been used: (a) packing spheres layer by layer from the boundary of R towards its interior [43], and (b) packing spheres from the "center" of R, such as its medial axis, to the boundary [10, 61, 62, 63]. Due to the shape irregularity of R, both approaches may fragment the "packable" subdomain R' into more and more disconnected regions; however, at the end of packing each such region, a small "unpackable" area may eventually remain that allows no further packing. It could fit more spheres if the "packable" subdomain R' is lumped together instead of being divided into fragments, which is what the trimming and packing procedure aims to achieve.

Due to the packing of its adjacent cells, the boundary of a cell $C(v)$ that is to be packed may consist of both line segments and arcs (of packed spheres). Hence we need to consider the problem of packing spheres in cells bounded by certain curves. In the trimming and packing phase, we present several packing algorithms for different types of domain with or without a curved boundary. Our packing algorithms are based on certain lattice structures and allow the domain R to both translate and rotate. All our algorithms have fairly low time complexities. In certain cases, they even run in nearly linear time. Our algorithms can be easily generalized to congruent packing of other shapes, and are readily extended to higher dimensional spaces.

An interesting feature of the packings generated by the trimming and packing procedure is that the resulted spheres cluster together in the middle of the domain R, leaving some small unpackable areas scattered along the boundary of R. Our "shake" procedure thus seeks to collect these small areas together by "blowing" the spheres to the boundary of R, in the hope of obtaining some "packable" region in the middle of R. We experiment quite a few techniques for efficiently shaking the packed spheres in R and compare their performances based on randomly generated input data.

We implemented our 2-D pack-and-shake algorithms and present a set of experimental results.

Due to the space limit, we omit many details and proofs of lemmas from this extended abstract.

2. PRELIMINARIES

Let $U = \{\vec{u_1}, \vec{u_2}, \ldots, \vec{u_d}\}$ be a set of d independent vectors in the d-D space E^d, and M be the $d \times d$ matrix formed by the d vectors. For any integer vector $\vec{x} = (x_1, x_2, \ldots, x_d)^T$ (i.e., each x_i is an integer), $M\vec{x}$ is a vector (or called *lattice point*) in E^d. The set of vectors so *generated* by using all integer vectors $\vec{x}$ forms a *lattice* L_U in E^d. M is called the *generator matrix* of L_U, and U is called the *basis*.

For each lattice L_U, there is a polyhedron B in E^d formed by the set of vertices, $v_0, v_1, \ldots, v_d, v_{d+1}$, where v_0 is the origin, $v_{d+1} = M * (1, 1, \ldots, 1)^T$, and $v_i = M\vec{e_i}$ for $1 \leq i \leq d$ and $\vec{e_i} = (\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0)^T$. B is called the *basic block* of

L_U, which can be translated to form a tile of E^d. In 2-D, the basic block is a parallelogram, called the *basic parallelogram.*

We say that a lattice L_U admits a packing (in the whole space) of congruent spheres of radius r if the Euclidean distance between any pair of lattice points is $\geq 2r$.

We denote a sphere packing of a bounded domain R as SP_R, and the densest sphere packing of R (i.e., a packing with the maximum number of spheres in R) as SP_R^{Max}. If the center of each sphere in SP_R is at a lattice point of L_U, then SP_R is called a lattice sphere packing of R, denoted as LSP_R. Similarly, we can define LSP_R^{Max}.

We denote a translation of an object O (e.g., a point, polygon, sphere, etc.) in E^d with offset vector $\Delta X = (\Delta x_1, \Delta x_2, \ldots, \Delta x_d)$ as $O + \Delta X$. ΔX is called the offset point.

3. PACK-AND-SHAKE ON A POLYGONAL DOMAIN WITHOUT CELL PARTITIONING

In this section, we present several algorithms for packing 2-D unit spheres in different types of simple polygon or polygonal domain R, under the assumption that the whole domain R is treated as a single cell (i.e., no further cell partitioning is done on R). Our algorithms are based on certain lattice structures, and have low time complexities. Combining with a global *shake* refinement procedure, the resulted packings exhibit a fairly high density as shown by experimental results in a later section.

Let R be an n-vertex simple polygon or polygonal domain (i.e., a polygon with holes), and $U = \{\vec{u_1}, \vec{u_2}\}$ be the basis of a lattice L_U on the plane P such that L_U admits a unit sphere packing. Ideally, one would like to obtain the densest sphere packing SP_R^{Max}. However, as mentioned in Section 1, this problem is NP-hard. Our goal thus is to seek efficient algorithms for producing dense packings SP_R.

Our approach is to first obtain the densest lattice unit sphere packing LSP_R^{Max}, and then use a shake procedure to globally adjust LSP_R^{Max} to generate a denser packing in R. Suppose that the plane P is already packed by infinitely many unit spheres, with each lattice point of L_U holding the center of one sphere. To obtain LSP_R^{Max} from the packing of P, we need to find a position and orientation of R on P such that R contains the maximum number of spheres from the packing of P. We discuss below two types of algorithms for computing the optimal position of R on P: translational algorithms that allow R to be translated, and rotational algorithms that allow R to be both translated and rotated.

3.1 2-D Lattice Packing with Translation

To produce efficiently the densest translational lattice sphere packing of R (denoted by $TLSP_R^{Max}$) in a given lattice L_U, we need to first identify a finite set S of spheres from the packing of P such that S contains at least one optimal solution for R. Once S is determined, we then seek an optimal position of R with respect to S. We solve this problem by reducing it to an interesting *thickest point problem*, and give two algorithms for the problem, one for the general case and the other for a special case.

A frequently arising decision is whether a sphere $s \in S$ is completely contained in R. To simplify this decision, we "shrink" the domain R by computing the Minkowski sum of a unit sphere s with the complement region $\overline{R}$ of R. The resulted region is denoted by R_-, i.e., $R_- = \overline{\overline{R} \oplus s}$. Note that the boundary of R_- may consist of both line segments and arcs (each arc is associated with a reflex vertex of R). After the shrinking, a sphere s is contained in R if and only if the center c_s of s is inside R_-.

The shrunk domain R_- need not be topologically equivalent to the original domain R. When this is the case, R_- is broken into multiple disconnected components, each corresponding to a subdomain of R. We pack on each component independently of other components. Note that it is possible that the spheres in two different components interfere with each other. However, such interferences can be handled with some care. Hence, we assume in this section that the shrunk domain R_- is one connected component with n vertices. The next lemma shows that shrinking R can be done efficiently.

LEMMA 1. *An n-vertex polygonal domain can be shrunk in $O(n \log n)$ time.*

Now we consider the problem of identifying the finite set S of spheres from the lattice L_U. Since the domain is already shrunk, we only need to identify a finite set of lattice points from L_U that gives rise to at least one optimal packing $TLSP_R^{Max}$. Let $v_1, v_2, \ldots, v_n$ be the n vertices of R_-. Let $R_- + (u, v)$ be a translation of R_- with offset point (u, v). Since the plane P can be tiled by translating infinite copies of the basic parallelogram of L_U, the following lemma holds.

LEMMA 2. *To obtain $TLSP_R^{Max}$, it is sufficient to translate R_- around inside the basic parallelogram.*

The above lemma suggests that in order to find $TLSP_R^{Max}$, it is sufficient for us to consider the set of spheres (actually the set of lattice points) which are away from R_- only by a basic parallelogram. As we move the offset point $f = (u, v)$ for R_- inside the basic parallelogram BP, lattice points may go in and out of R_-, causing the number of spheres contained in R to change. Thus, we need to identify the set S of lattice points which may cross the boundary of R_- while f is moving around BP.

To compute S, we consider a segment $\overline{ab}$ of R_- (an arc can be handled similarly). The lattice points which may cross or touch $\overline{ab}$ can be determined as follows. Let (x_a, y_a) and (x_b, y_b) be the coordinates of points a and b, respectively. We first determine to which parallelogram, BP_a, a belongs. (This can be easily done in $O(1)$ time by decomposing $\vec{a}$ into two vectors $\vec{a_1}$ and $\vec{a_2}$ along the directions of the basis vectors $\vec{u_1}$ and $\vec{u_2}$ and dividing $\vec{a_i}, i = 1, 2$, by $\vec{u_i}$ to obtain the integer vector $\vec{x_0} = (i_0, j_0)^T$.) Then we check the four lattice points of BP_a to see whether they cross $\overline{ab}$ while moving f in BP, and add each crossing point into a set $S_{\overline{ab}}$. For each lattice point $s_i = M\vec{x_i}$ in $S_{\overline{ab}}$, we further check the following three lattice points: $M(\vec{x_i} + (1, 0)^T)$, $M(\vec{x_i} + (0, 1)^T)$, and $M(\vec{x_i} + (1, 1)^T)$, and add the crossing points into $S_{\overline{ab}}$. By repeating this procedure, all lattice points which may cross $\overline{ab}$ are put into $S_{\overline{ab}}$.

Repeating the above procedure for each segment and arc of R_-, we obtain a collection of point sets. The union of them forms the sought lattice point set S. The size of S is roughly $O(m)$, where m is the total number of lattice points along the boundary of R_- in $TLSP_R^{Max}$.

Once S is obtained, we need to determine an optimal translational position for R_-. That is, we need to determine a point f_o in BP for the offset point f such that $R_- + f_o$ contains the maximum number of points in S. For this purpose, we compute, for each lattice point $s \in S$, a region s_f inside BP such that when f moves inside s_f, s is contained in $R_- + f$. The region s_f is called the *containing region* of s. The optimal offset point f_o is thus a point that is covered by the maximum number of containing regions. Below we discuss how to compute the containing region of each lattice point of S.

Assume that $\vec{s} = M\vec{x_s}$ is a member of $S_{\overline{ab}}$. (The case in which $\overline{ab}$ is an arc can be handled similarly.) We first compute the locus of the offset point f in BP when the boundary of $R_- + f$ touches s. Each point w on $\overline{ab} + f$ can be represented by its corresponding vector $\vec{w} = \vec{a} + \vec{f} + t(\vec{b} - \vec{a})$ for some $0 \le t \le 1$. In order to make $\overline{ab} + f$ touch s, we must have some point $\vec{w} = \vec{s}$. This means that there is a value $t \in [0,1]$ such that $\vec{w} = \vec{a} + \vec{f} + t(\vec{b} - \vec{a}) = \vec{s}$. Thus, the locus of f can be determined by the equality $\vec{f} = \vec{s} - \vec{a} + t(\vec{a} - \vec{b}), t \in [0,1]$. (Note that the slope of the locus is the same as that of $\overline{ab}$.) The locus of f is trimmed by the boundary of BP if it goes outside of BP. For each segment $\overline{uv}$ whose $S_{\overline{uv}}$ contains s, we compute a locus of f. All these loci, possibly together with the boundary of BP, form the containing region s_f of s.

The containing region s_f has some interesting properties.

PROPERTY 1. *If R_- is convex, then all containing regions are convex. If R_- contains a convex hole, then the containing region of a lattice point generated by an edge of this hole is the complement of a convex region in BP.*

The number of edges in each containing region is less than $n + 4$. If all edges of R_- are "long" (i.e., $\ge |\vec{u_1} + \vec{u_2}|$), then each lattice point can touch or cross only a constant number of edges of R_-, and thus each containing region has a constant number of edges.

The above approach for generating containing regions can be extended to the case in which R_- is bounded by a set of algebraic curves.

PROPERTY 2. *If R_- is bounded by a set of algebraic curves, then each containing region is also bounded by a set of algebraic curves. Further, each curve of R_- has the same degree as its corresponding curve of the containing region.*

Since the shrinking procedure does not change the degrees of the curves of R_-, based on Property 2, we can now consider the translational lattice packing problem in a domain R bounded by algebraic curves. Once the set of containing regions is generated, the packing problem is reduced to the following *thickest point problem*, where the *thickness* of a point is defined as the number of regions covering it.

PROBLEM 1. *Given a set F of connected regions on the plane, with each region bounded by a set of algebraic curves, find the thickest point on the plane.*

To solve this problem, let Γ be the set of algebraic curves bounding the regions in F, and $N = |\Gamma|$. Obviously, Γ forms an arrangement A on the plane. If the degree of each curve in Γ is bounded by a constant, then any two such curves intersect only a constant number of times. Thus, we can use known arrangement algorithms [21, 3] to first construct A in $O(N \log N + K)$ time, where K is the total number of intersections in A. In the worst case, $K = O(N^2)$. The optimal point can then be found by traversing the cells of A and computing the thickness of each cell.

LEMMA 3. *If every edge curve in Γ is an algebraic curve with a constant degree, then the thickest point problem on F can be solved in $O(N \log N + K)$ time.*

Next, we present a faster algorithm for a special case of the thickest point problem (this case does arise in our trimming and packing process, as to be shown later). In this special case, we assume that every region of F is a polygon possibly with holes, and further, the total number of different orientations among all edges in Γ is a constant c. To exploit the properties of this special case, we first perform a decomposition on each polygon P of F. It partitions P into a set of convex polygons such that the total number of edge orientations of the resulted convex polygons is the same as that of P, and the total number of vertices is $O(|P|)$.

LEMMA 4. *Each polygon P can be partitioned by a trapezoidal decomposition in $O(|P| \log |P|)$ time.*

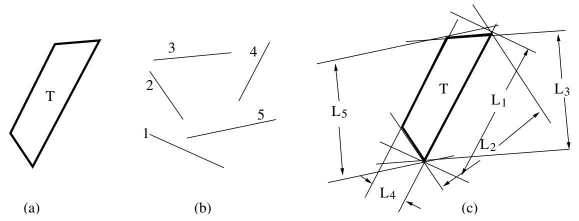

Figure 1: Converting a trapezoid T to a box in E^5: (a) the trapezoid T; (b) the five directions for E^5; (c) the distances for the corresponding five dimensions of E^5.

Lemma 4 generates $O(N)$ trapezoids. Once F is decomposed in this way, we use an orthogonalization procedure to convert every such trapezoid T into a box B_T in the c-D space E^c, where c is the number of edge orientations of Γ. Each orientation represents a dimension in E^c. In every orientation, we draw two (closest) parallel lines to "enclose" the trapezoid T (see Figure 1 for an example). The distance between the two parallel lines is mapped to the length of B_T in the corresponding dimension. After the conversion, we have a set $\mathcal{B}$ of $O(N)$ boxes in E^c.

Note that the above conversion does not change the thickness of any point in E^c. This is because the conversion is a one-to-one mapping from the 2-D plane to the c-D space E^c. To efficiently solve the thickest point problem in E^c, we use a k-D tree to represent the $O(N)$ boxes in $\mathcal{B}$. First, we build a k-D tree KD based on the $O(N)$ vertices of all boxes in $\mathcal{B}$. KD has $O(N)$ nodes. Each node of KD corresponds to a range of a box shape in E^c. We let every node v of KD maintain a thickest point, together with its thickness value t_v, in the corresponding range. Each box in $\mathcal{B}$ can be decomposed into $O(N^{1-\frac{1}{c}})$ sub-boxes, with each sub-box being exactly the same as the range of a certain node in KD. We say that a sub-box b matches a node v of KD if they have the same range. Thus, we can repeatedly insert each of the $O(N)$ boxes of $\mathcal{B}$ into KD. Every time when a sub-box b matches a node v of KD, we increase the thickness t_v of v by one.

The thickness of the $O(N)$ boxes of $\mathcal{B}$ can be obtained by performing a depth-first search in KD. Each time when a node v of KD is visited, the sum of the thickness of all the ancestors of v in KD is propagated to v. Thus, the overall thickest point is obtained at a leaf node of KD.

Each box of $\mathcal{B}$ can be inserted into KD in $O(N^{1-\frac{1}{c}})$ time. Thus, the total insertion time is $O(N^{2-\frac{1}{c}})$. Clearly, the construction and depth-first search of KD can also be done in the same amount of time. The trapezoidal decomposition and the orthogonalization procedure can be done in $O(N \log N)$ time. Hence we have the following lemma.

LEMMA 5. *The thickest point problem on a set F of polygons (possibly with holes) with a constant c of different edge orientations can be solved in $O(N^{2-\frac{1}{c}})$ time and $O(N)$ space, where N is the total number of vertices of F.*

Remark: If $c = 2$, an $O(N \log N)$ time and space solution can be obtained by using a range-tree based data structure. (The details are left to the full paper.)

Now, we go back to the sphere packing problem. As we have shown, a containing region is computed from R_- and a lattice point in S. Each edge (except for the boundary of BP) of a containing region has the same orientation as its corresponding edge of R_-. Thus, if R (or equivalently R_-) is convex and has only a constant number c of edge orientations, then the total number of edge orientations in Γ is $c + 2$, where 2 is the number of edge orientations of BP. (Note that since an optimal point can always be found inside BP, we can actually ignore the boundary of BP when computing the thickest point.) Hence, such a special case of the $TLSP_R^{MAX}$ problem can be solved efficiently.

LEMMA 6. *For a convex polygonal domain R with a constant number c of different edge orientations, the $TLSP_R^{MAX}$ problem can be solved in $O(N^{2-\frac{1}{c}})$ time and $O(N)$ space, where N is the total number of edges of the generated containing regions. Particularly, if $c = 2$, the problem can be solved in $O(N \log N)$ time and space.*

For the general case (i.e., R is bounded by a set of degree-bounded algebraic curves), we have the following lemma.

LEMMA 7. *Given a domain R with its boundary edges being algebraic curves of constant degrees, the $TLSP_R^{Max}$ problem can be solved in $O(N \log N + K)$ time, where N is the number of edges of the generated containing regions, and K is the size of the arrangement formed by the containing regions.*

In the above two lemmas, $N = n \times m$ in the worst case, where n is the size of R and m is the total number of spheres along the boundary of R in $TLSP_R^{Max}$. Note that, in practice, N may be much smaller than $n \times m$. For example, for the cases in which all bounding edges of R are "long" edges (i.e., $\geq |\vec{u_1} + \vec{u_2}|$), or "short" edges are separated by long edges, the containing region of each lattice point has only a constant number of edges, and thus N is roughly $O(n + m)$.

3.2 2-D Lattice Packing with Translation and Rotation

For a domain R, the translational lattice packing may not yield the densest lattice packing due to the given orientation of R. Thus, we like to study the rotational lattice packing problem that allows R to be translated and rotated. We denote the densest rotational lattice sphere packing of R by $RLSP_R^{Max}$.

To compute $RLSP_R^{Max}$, similar to the translational lattice packing, we first shrink R to obtain R_-. We then identify a set S of lattice points which may cross the boundary of R_- while translating and rotating R_-. Finally, we determine the optimal position and orientation of R.

To compute S, we consider all possible lattice points which may cross an edge of R_- while translating and rotating R_-. The rotation of R_- may cause a large set of lattice points to cross the boundary of R_-. For each boundary edge $\overline{ab}$ of R_-, the number of crossing lattice points of $\overline{ab}$ is roughly proportional to the area "swept" by $\overline{ab}$ during the rotation, which clearly depends on the location of the rotation center. Thus, the problem of minimizing the set S is reduced to finding an optimal rotation center that minimizes the total area swept by all $O(n)$ edges of R_-. We call this the *annuli minimization problem.*

To find an optimal rotation center, we consider an edge $\overline{ab}$ of R_-. Suppose that the rotation center is $r_o = (x, y)$. Then the rotation of $\overline{ab}$ sweeps the plane and forms an annulus $nu_{\overline{ab}}$. The area of $nu_{\overline{ab}}$ is equal to $\pi(r_{out}^2 - r_{in}^2)$, where r_{out} and r_{in} are the radii of its inner and outer circles. Precisely, r_{out} is the distance from r_o to the furthest point of a and b, and r_{in} is the distance from r_o either to the supporting line l_{ab} of $\overline{ab}$ or to the nearest of a and b. There are four different cases. To distinguish these cases, we draw three lines: one is the bisector bs_{ab} of $\overline{ab}$, and the other two, l_a and l_b, are parallel to bs_{ab} and passing through a and b respectively (see Figure 2 for an example). In each of the four regions partitioned by these three lines, the area of the annulus is either a linear or quadratic function of x and y. Thus, the optimal rotation center can be computed by first constructing an arrangement A' from the $O(n)$ partitioning lines obtained from all edges of R_-, and then in each convex cell of A', finding an optimal rotation center point by solving the associated quadratic minimization problem on that cell. The objective function for each cell of A' can be updated in an online fashion.

LEMMA 8. *For a domain R_- of n edges, the annuli minimization problem can be solved by reducing it to solving $O(n^2)$ quadratic minimization problems. Furthermore, these $O(n^2)$ optimization problems (including both the objective functions and constraints) can be generated in altogether $O(n^2)$ time.*

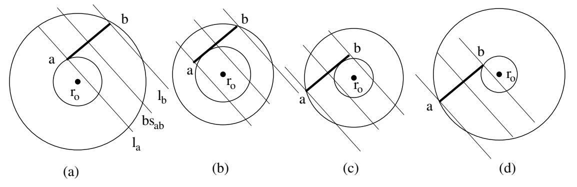

Figure 2: Four types of annuli.

Note that the above approach for computing the rotation center is also applicable to the case when the rotation angle is $< 2\pi$ (e.g., for a hexagonal lattice, it is sufficient to rotate R_- between 0 and $\frac{\pi}{3}$).

Once the rotation center is located, we translate the origin of the coordinate system to this center. (This changes the coordinates of the vertices of R_-.) The set S of crossing lattice points can then be computed from those lattice points which are either contained by any of the n annuli (or a portion of an annulus if the rotation is $< 2\pi$) or within a distance of a basic parallelogram away from the lattice points contained by the annuli (for translation). The total time for generating S is $O(|S|)$. Similar to our translational lattice packing approach, the rotational lattice packing algorithm needs to identify the containing regions for all lattice points in S with respect to R_-. With the motions of both translation and rotation, the space for the containing regions in this case becomes 3-D (the 3-rd dimension, called the α-axis, represents the rotation angles). To generate the set of 3-D containing regions, we first compute the set of the 2-D initial containing regions (i.e., with a rotation angle $\alpha = 0$). Due to the rotation, each edge of a 2-D containing region becomes a 3-D *spiral* surface patch. To see that, we consider a segment $\overline{ab}$ of R_- and a lattice point $s \in S$. Each point p of $\overline{ab}$ can be represented as $\vec{p} = \vec{a} + \vec{f} + tW(\vec{b} - \vec{a})$, $0 \le t \le 1$, where f is the offset point, W is the rotation matrix with

$$W = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

and θ is the rotation angle. The locus of the 3-D point (f, θ) forms a surface SF_{ab} when $\overline{ab}$ is translated and rotated as it remains touching s. The surface SF_{ab} can be expressed by the equality $\vec{f} = \vec{s} - \vec{a} + tW(\vec{a} - \vec{b})$, $0 \le t \le 1$.

In 3-D, the basic parallelogram BP becomes a polygonal cylinder C. Surface patches which go outside of C are truncated at the boundary of C. Denote the set of (truncated) surface patches and the boundary of C as Γ. The following lemma shows some nice property of the surface patches in Γ.

LEMMA 9. *The surface patches in Γ are all pseudo-planes. That is, any two surface patches in Γ intersect at a continuous curve (if they intersect each other), and any three surface patches intersect at no more than one point.*

With the above lemma, we solve the thickest point problem in 3-D by using a space-sweeping algorithm along the α-axis. Note that each containing region in 3-D is a polygonal cylinder spiraling up along the α-axis. The thickest point is covered by the maximum number of such cylinders. The space-sweeping algorithm first constructs the planar arrangement A_0 corresponding to the rotation angle $\alpha = 0$, and computes the thickness map for each cell of A_0. To determine the next event point, we consider an edge $e_i \in A_0$. Let e_{i-1} and e_{i+1} be the neighboring edges of e_i along a cell of A_0. Denote the three surface patches generated by the rotation of the three edges as SF_i, SF_{i-1}, and SF_{i+1}. We compute the intersection point $v_{i,i-1}$ (resp., $v_{i,i+1}$) between the bounding curves of SF_i and SF_{i-1} (resp., SF_{i+1}) with the other surface patch, and the intersection point v of the three surface patches. All these intersection points are inserted into a priority queue based on their α-coordinates. The next event point is the intersection point in the priority queue with the smallest α-coordinate. At each event point, the algorithm may generate new event points, and it identifies whether a new cell is created or an old cell disappears. For these two cell changing cases, the algorithm needs to update the thickness map. The thickness of a new cell can be computed from its neighboring cells in $O(1)$ time. In this way, the sweeping algorithm spends only $O(1)$ time to compute each vertex of the whole 3-D arrangement A of Γ, and the total computation cost can be charged to the vertices of A with a logarithmic cost per vertex. Thus, we have the following lemma.

LEMMA 10. *The space-sweeping algorithm finds the thickest point in 3-D in $O((K + N)\log N)$ time, where K is the size of the 3-D arrangement A and $N = |\Gamma|$.*

In the worst case, $K = O(N^3)$. Based on our experiments, K is normally much smaller than $O(N^3)$.

Remark: The above approach for finding rotational lattice sphere packing can be extended to the case in which R is bounded by a set of algebraic curves with constant degrees.

3.3 Shaking a Lattice Packing

The lattice packings produced in Subsections 3.1 and 3.2 have an interesting property. That is, the resulted spheres cluster in the middle of the domain R. For certain lattice structures, e.g., the 2-D hexagonal lattice whose basis is $\{(2,0)^T, (1, \sqrt{3})^T\}$, the packings are very tight at the middle. Locally, the packing may even be optimal in the middle. But globally, there may still be some small "unpackable" areas scattered along the boundary of R. To make use of these small areas, we apply a "shake" procedure which tries to "blow" the spheres from the "center" of R to its boundary and gather these unused small areas to the center for packing more spheres.

From the experiments, we observed that unused yet unpackable areas frequently occur around the corners (vertices) of the domain. Thus, we first push a sphere to each unoccupied corner. Next, we repeatedly apply two "move" and "drop" procedures. These two procedures could be viewed as making the movements of spheres in a special field of forces. The directions of the moving spheres can change and may be different for different spheres. Unlike some physical simulation such as [48], we can let a sphere "pass through" some baffled spheres to reach a stable point under the effect of the field. Several different shake procedures are implemented and tested, based on different ways to choose the moving directions and different criteria for the movements. These procedures can also be combined together to perform the shaking task. The details are left to the full paper.

We continue the shaking until no more sphere can be added. During the shaking process, whenever we detect that an empty space large enough for a sphere shows up, we put a new sphere to take that space. As a result, the execution time of the shaking is quite fast (often in several to tens of seconds).

4. TRIMMING AND PACKING

In the last section, we use the domain R as a whole to pack spheres. The packings so generated are very dense for "fat" or convex domains, since the initial lattice packings already occupy most area in the middle of such domains and leave only very small unused areas along their boundaries.

However, this strategy does not always work well, especially for irregular-shaped domains. Due to its local structures, different parts of a domain R may desire different translation and rotation lattice packings. Treating R as a single cell may make the overall lattice packing density very low. In Figure 3(a), for example, the domain R consists of multiple "strips" connected together, and each strip can pack a row of spheres; it is possible that no matter which translation or rotation lattice packing is used, the packing density remains very low if the whole R is treated as one cell. A better approach is to partition R into multiple "nice" cells, and pack each cell somewhat independently. Figure 3(b) shows an optimal packing based on a partition of R.

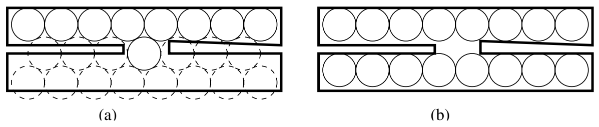

Figure 3: (a) Lattice packing on R as a whole may not yield a dense packing. (b) An optimal packing obtained by using lattice packing in two subdomains.

Another reason for partitioning a domain into multiple cells is to achieve a better time efficiency. In Section 3, the packing algorithms all take super-linear time in terms of the number N of edges or surfaces of the containing regions. N is between $O(n+m)$ and $O(nm)$, where n is the number of edges of R_- and m is the number of involved crossing lattice points. Partitioning R into smaller "fat" and convex cells makes the values of n and m for packing each cell much smaller than those for R, and thus speeds up the overall running time significantly. Besides, such a partitioning of R enables us to make use of the efficient algorithm in Section 3 for the special case of the 2-D thickest point problem (for a convex domain with a constant number of edge orientations).

To achieve a faster and denser packing, we use a procedure called *trimming and packing*. This procedure first partitions the domain R into a set of triangles or trapezoids. Then the neighboring triangles or trapezoids are merged to form larger convex cells. The set of cells define a dual graph G_D (each vertex v of G_D is for a cell $C(v)$, and an edge connects two vertices if their d-D cells share a $(d-1)$-D face). To pack spheres in all cells, the procedure repeatedly removes the lowest degree vertex v from G_D and packs $C(v)$ by using the algorithms in Section 3. The details of this procedure are in the full paper.

An important feature of our trimming and packing procedure is that the remaining "packable" subdomain is always a connected component. This prevents our algorithm from producing many "unpackable" small areas, a problem occurring in some commonly used approaches [10, 43, 61, 63].

5. EXTENSIONS AND APPLICATIONS

Our sphere packing algorithms in previous sections can be extended in several directions, which we sketch in this section. These include extending the lattice packing algorithms to 3-D and high dimensions, and applying the sphere packing algorithms to treatment planning of radiosurgery.

To pack spheres in a 3-D domain, we also use the trimming and packing procedure to partition the domain into convex cells. For each cell, we use lattice packing algorithms to find a good initial packing, and shaking procedures to improve the packing. Since the trimming and packing, shaking, and shrinking algorithms are similar to those for 2-D, we focus on the lattice packing step. Further, our experiments suggest that for convex domains, the quality of translation packings is very close to that of rotation packings. Thus, we only discuss the 3-D translational lattice packing in a convex cell.

Suppose after shrinking, the convex domain R consists of n vertices. Further, assume that the boundary of R_- is triangulated. Since a lemma similar to Lemma 2 holds in 3-D (which can be proved similarly), the set S of crossing lattice points can be easily computed. Thus, to compute the containing regions, we only need to consider a triangle Δ_{abc} on the boundary of R_- against a lattice point $s \in S$.

Let w be any point of $\Delta_{abc} + f$, where f is the offset point of R_- inside the basic block of L_U. Then w can be expressed as $\vec{w} = \vec{a} + \vec{f} + t_1(\vec{b} - \vec{a}) + t_2(\vec{c} - \vec{a})$ with $t_1 \in [0,1]$, $t_2 \in [0,1]$, and $t_1 + t_2 \in [0,1]$. The locus of f, when Δ_{abc} touches s, is determined by $\vec{w} = \vec{s}$, and can be expressed by $\vec{f} = \vec{s} - \vec{a} + t_1(\vec{a} - \vec{b}) + t_2(\vec{a} - \vec{c})$. Clearly, this is a triangle with the same normal as Δ_{abc}.

The optimal 3-D translational lattice packing $TLSP_R^{Max}$ can be obtained by finding the thickest point inside the basic block. Using a space-sweeping algorithm, we have the following lemma.

LEMMA 11. *For a 3-D convex polyhedral domain with n vertices, the thickest point problem can be solved in $O((K+N)\log N)$ time, where N is the number of faces of the set of containing regions and K is the size of the 3-D arrangement generated by the set of N faces.*

For the case in which the 2-D faces of R have only a constant number c of different orientations, we can obtain a set CR of containing regions with a constant number of different face orientations. Thus, we can decompose CR into a set of L convex regions with $c+3$ different face orientations, where 3 is for the orientations of the boundary faces of the basic block. In the worst case, $L = O(N^2)$. By using an orthogonalization technique similar to that used in Section 3 and ignoring the boundary of the BP during computing the thickest point, we have the following lemma.

LEMMA 12. *For a 3-D convex domain R with a constant number c of different 2-D face orientations, $TLSP_R^{Max}$ can be computed in $O(L^{2-\frac{1}{c}})$ time and $O(L)$ space.*

It is also possible to extend our approach for translational lattice sphere packing to higher dimensional spaces.

Our congruent sphere packing algorithms can be used as a key procedure for solving the sphere packing problem arising in Gamma Knife surgical treatment planning. Based on a common approach used in practice, we can first put congruent balls of the largest available size into the domain, and then repeat for the next size balls, until no more balls can be put into the domain. Note that our algorithms for domains bounded by algebraic curves are especially useful in this setting (since packing larger balls leaves subdomains bounded by arcs). This common approach is intended for reducing the total number of balls used and thus shortening the treatment time.

After the balls are placed into the domain, another problem in Gamma Knife surgical treatment planning is to determine how much radiation (i.e., the weight) each ball should deliver. Note that the radiation in a ball also affects the surrounding uncovered parts of the tumor domain. To completely destroy a tumor, every point p of the tumor domain should receive a prescribed amount of radiation. In particular, if p is not in a ball, then it must receive the required radiation from the "cross-firing" of radiation by nearby balls. This is a weight assignment problem for balls after a packing is done. We can solve this problem by computing an arrangement of balls and using linear programming.

6. IMPLEMENTATION AND EXPERIMENT

In this section, we present some experimental results on our 2-D pack-and-shake sphere packing algorithms. All our implementations use the hexagonal lattice as an example.

We implemented our algorithms on Sun Ultra Sparc 30 workstations using the C++ based library LEDA 4.1. Experiments suggest that our algorithms produce reasonably dense packings for polygonal domains of almost all shapes.

We experimented four algorithms: *Translation*, *Translation and Shake*, *Rotation*, and *Rotation and Shake*. The first two algorithms ran very fast, normally in about one minute. However, the last two algorithms suffered very long execution time, and hence we simplified them in order to speed up their executions (but the simplifications may give rise to a less dense packing quality). Even after our simplifications, the last two algorithms still ran quite long. An explanation for such long execution time is that these two algorithms need to compute the 3-D arrangement of $O(N)$ complicated surfaces as discussed in Section 3; furthermore, computing such 3-D arrangements involves solving quite many nonlinear equation systems, which takes nontrivial numerical manipulations.

Tables 1 and 2 summarize some of our implementation results on arbitrary convex polygonal domains and non-convex polygonal domains, respectively. In these tables, T is for *Translation*, TS for *Translation and Shake*, R for *Rotation*, and RS for *Rotation and Shake*; an entry NA means that during the executions, our machines were rebooted by others since the program needed a very long time to run on the examples. Our data were obtained by taking the average of many examples that we ran. The numbers of spheres used in those packing examples range from 20 to 300.

Our results show that in most cases, *Rotation and Shake* gave a better packing quality, but suffered a long execution time. While *Translation and Shake* sometimes gave a little worse quality than *Rotation and Shake*, it ran much faster. Thus, there is a tradeoff between the execution time and packing quality.

We also compared our packings on domains of triangles and squares against some known results obtained by using optimization methods [55]. In particular, we packed spheres in the unit square by using the algorithm TS (Translation and Shake), and compared our results with those in [55]. Note that the problem studied in [55] is somewhat different: Given a domain and an integer k, pack k congruent spheres in the domain to maximize their radius. Some nonconvex programming methods are used in [55] for that problem, and their algorithms generally run in hours. We used the output radii of [55] as our input radii. This comparison thus may not be very fair. In Figure 4, the solid curve denotes the results for our TS algorithm, and the dashed curve denotes the results for methods in [55]; the x-axis is for the radii of the spheres, and the y-axis is for the number of spheres packed in the unit square. It shows that our packing qualities are no more than 10% worse than those in [55]. But our TS packing algorithm only ran in less than 5 seconds (comparing with the hours of [55]).

Two examples of actual packings produced by our algorithms are given in Figure 5 and 6.

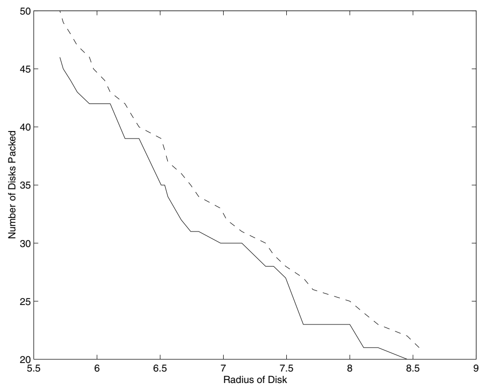

Figure 4: Comparison of packing results on square domains.

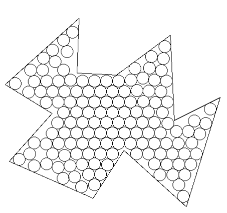

Figure 5: Example of sphere packing. Algorithm / Density / Time: Trans. + Shake / 0.743458 / 285.399s.

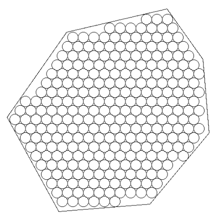

Figure 6: Example of sphere packing. Algorithm / Density / Time: Translation / 0.828477 / 81.2697s.

Acknowledgement

The authors are very grateful to Dr. Cedric Yu, Department of Radiation Oncology, School of Medicine, University of Maryland at Baltimore, for discussing the sphere packing problems and related Gamma Knife surgical treatment planning problems with us and for providing useful references.

7. REFERENCES

[1] P.K. Agarwal and M. Sharir, "Pipes, cigars, and kreplach: The union of Minkowski sums in three dimensions," *Proc. 15th ACM Symp. on Computational Geometry*, 1999, pp. 143-153.

[2] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C.K. Yap, "Finding minimal convex nested polygons," *Proc. 1st Annu. ACM Symp. Comput. Geom.*, 1985, pp. 296-304.

Table 1: Packing results on convex polygonal domains.

Radius	4	3.5	3	2.5	2	1.5
Density of T	0.721905	0.740449	0.758691	0.775426	0.789447	0.799397
Time of T	14.56s	18.32s	31.96s	51.49s	86.67s	168.56s
Density of TS	0.747356	0.747707	0.760555	0.788978	0.789969	0.800245
Time of TS	18.6563s	24.96s	37.23s	62.19s	116.94s	197.18s
Density of R	0.731554	0.741701	0.741118	0.771218	0.787096	NA
Time of R	31510s	52205s	61927s	68801s	73541s	NA
Density of RS	0.748134	0.74969	0.761018	0.785398	0.790388	NA
Time of RS	32338s	53096s	62364s	68891s	73702s	NA

Table 2: Packing results on nonconvex polygonal domains.

Radius	4	3.5	3	2.5	2	1.5
Density of T	0.61335	0.624044	0.677518	0.718292	0.741687	0.76385
Time of T	11.94s	13.12s	18.72s	29.26s	37.62s	47.77s
Density of TS	0.628908	0.62995	0.680645	0.735003	0.744466	0.765882
Time of TS	12.98s	17.34s	24.82s	29.91s	45.23s	86.00s
Density of R	0.61575	0.621292	0.673455	0.734508	0.742732	0.764343
Time of R	33313s	42606s	52997s	62486s	75311s	86355s
Density of RS	0.628787	0.62343	0.681582	0.751713	0.766709	NA
Time of RS	33417s	42714s	54005s	63493s	76534s	NA

[3] N.M. Amato, M.T. Goodrich, and E.A. Ramos, "Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling," *Proc. 11th Annual ACM-SIAM Symp. on Discrete Algorithms*, 2000, pp. 705–706.

[4] B.S. Baker, D.J. Brown and H.K. Katseff, "A 5/4 algorithm for two-dimensional packing," *Journal of Algorithms*, Vol. 2, 1981, pp. 348-368.

[5] B.S. Baker, E.G. Coffman, Jr., and R.L. Rivest, "Orthogonal packing in two dimensions," *SIAM J. Comput.*, Vol. 9, No. 4, 1980, pp. 846-855.

[6] B.S. Baker and J.S. Schwarz, "Shelf algorithms for two-dimensional packing problems," *SIAM J. Comput.*, Vol. 12, No. 3, 1983, pp. 508-525.

[7] G. Bär and C. Iturriaga, "Rectangle packing in polynomial time," *Proc. 5th Canad. Conf. Comput. Geom.*, 1993, pp. 455-460.

[8] C. Baur and S.P. Fekete, "Approximation of geometric dispersion problems," *Proc. APPROX'98*, 1998, pp. 63-75.

[9] M. Bern, S. Mitchell, and J. Ruppert, "Linear-size nonobtuse triangulation of polygons," *Proc. 10th Annu. ACM Symp. on Comput. Geom.*, 1994, pp. 221-230.

[10] J.D. Bourland and Q.R. Wu, "Use of shape for automated, optimized 3D radiosurgical treatment planning," *SPIE Proc. Int. Symp. on Medical Imaging*, 1996, pp. 553-558.

[11] S.-W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, and S.-H. Teng, "Silver exudation," *Proc. 15th Annu. ACM Symp. on CS88 Comput. Geom.*, 1999, pp. 1-10.

[12] L.P. Chew, "Guaranteed-quality Delaunay meshing in 3D (short version)," *Proc. 13th Annu. ACM Symp. on Comput. Geom.*, 1997, pp. 391-393.

[13] E.G. Coffman, Jr., M.R. Garey, D.S. Johnson, and R.E. Tarjan, "Performance bounds for level-oriented two-dimensional packing algorithms," *SIAM J. Comput.*, Vol. 9, 1980, pp. 808-826.

[14] E.G. Coffman, Jr., and J.C. Lagarias, "Algorithms for packing squares: a probabilistic analysis," *SIAM J. Comput.* , Vol. 18, No. 1, 1989, pp. 166-185.

[15] E.G. Coffman, Jr. and P.W. Shor, "Average-case analysis of cutting and packing in two dimensions," *European Journal of Operational Research*, Vol. 44, 1990, pp. 134-144.

[16] J.H. Conway and N.J.A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York, 1988.

[17] K.M. Daniels and V.J. Milenkovic, "Column-based strip packing using ordered and compliant containment," *Proc. of the First ACM Workshop on Applied Computational Geometry (WACG)*, 1996, pp. 33-38.

[18] K.M. Daniels and V.J. Milenkovic, "Multiple translational containment, part I: An approximation algorithm," *Algorithmica*, Vol. 19, 1997, pp. 148-182.

[19] K.M. Daniels, V.J. Milenkovic, and D. Roth, "Finding the maximum area axis-parallel rectangle in a simple polygon," *Computational Geometry: Theory and Applications*, Vol. 7, 1997, pp. 125-148.

[20] H. Dyckhoff, "A topology of cutting and packing problems," *European Journal of Operational Research*, Vol. 44, 1990, pp. 145-159.

[21] H. Edelsbrunner, L.J. Guibas, J. Pach, R. Pollack, R. Seidel, M. Sharir, "Arrangements of curves in the plane: Topology, combinatorics, and algorithms," *Theoretical Computer Science*, Vol. 92, 1992, pp. 319–336.

[22] S.P. Fekete and J. Schepers, "On more-dimensional packing I: Modeling," technical report, ZPR 97-288, Website http://www.zpr.uni-koeln.de/p̃aper.

[23] S.P. Fekete and J. Schepers, "On more-dimensional packing II: Bounds," technical report, ZPR 97-289, Website http://www.zpr.uni-koeln.de/p̃aper.

[24] S.P. Fekete and J. Schepers, "On more-dimensional packing III: Exact algorithm," technical report, ZPR 97-290, Website http://www.zpr.uni-koeln.de/p̃aper.

[25] M. Formann and F. Wagner, "A packing problem with applications to lettering of maps," *Proc. 7th Annu. ACM Symp. Comput. Geom.*, 1991, pp. 281-288.

[26] R.J. Fowler, M.S. Paterson, and S.L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Information Processing Letters*, Vol. 12, No. 3, 1981, pp. 133-137.

[27] E. Friedman, "Packing unit squares in squares: A survey and new results," *The Electronic Journal of Combinatorics*, Vol. 5, 1998, #DS7.

[28] Z. Füredi, "The densest packing of equal circles into a parallel strip," *Discrete & Computational geometry*, Vol. 6, 1991, 95-106.

[29] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.

[30] I. Golan, "Performance bounds for orthogonal oriented two-dimensional packing algorithms," *SIAM J. Comput.*, Vol. 10, No. 3, 1981, pp. 571-582.

[31] R.L. Graham and B.D. Lubachevsky, "Dense packing of equal disks in an equilateral triangle: From 22 to 34 and beyond," *The Electronic Journal of Combinatorics*, Vol. 2, 1995, #A1.

[32] R.L. Graham and B.D. Lubachevsky, "Repeated patterns of dense packings of equal disks in a square," *The Electronic Journal of Combinatorics*, Vol. 3, 1996, #R16.

[33] R.L. Graham and N.J.A. Sloane, "Penny-packing and two-dimensional codes," *Discrete & Computational Geometry*, Vol. 5, 1990, pp. 1-11.

[34] J. Heistermann and T. Lengauer, "Efficient automatic part nesting on irregular and inhomogeneous surfaces," *Proc. 4th ACM-SIAM Symp. Discrete Algorithms*, 1993, pp. 251-259.

[35] D. Hilbert, *Mathematische Probleme*, Archiv. Math. Phys. 1 (1901), 44-63 and 213-237 = Gesamm. Abh., III, 290-329. English translation in BAMS 8 (1902), 437-479 = *Proc. of Symposia in Pure Mathematics*, 28 (1976), 1-34.

[36] D.S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *Journal of ACM*, Vol. 32, No. 1, 1985, pp. 130-136.

[37] G.A. Kabatiansky and V.I. Levenshtein, "Bounds for packings on a sphere and in space," *Problemy Peredachi Informatsii*, Vol. 14, No. 1, 1978, pp. 3-25.

[38] R.M. Karp, M. Luby, and A. Marchetti-Spaccamela, "A probabilistic analysis of multidimensional bin-packing problems," *Proc. 16th Annu. ACM Symp. Theory Comput.*, 1984, pp. 289-299.

[39] M. Keil and J. Snoeyink, "On the time bound for convex decomposition of simple polygons," to appear in *International J. of Computational Geometry and Applications.*

[40] G. Kyperberg and W. Kuperberg, "Double-lattice packings of convex bodies in the plane," *Discrete & Computational Geometry*, Vol. 5, 1990, pp. 389-397.

[41] J. Leech, "Sphere packing and error-correcting codes," *Canadian Journal of Mathematics*, Vol. 23, 1971, pp. 718-745.

[42] K. Li and K.H. Cheng, "On three-dimensional packing," *SIAM J. Comput.*, Vol. 19, No. 5, 1990, pp. 847-867.

[43] X.-Y. Li, S.-H. Teng, and A. Üngör, "Biting: Advancing front meets sphere packing," to appear in a special issue of *International Journal of Numerical Methods in Engineering*, 1999.

[44] Z. Li and V.J. Milenkovic, "Compaction and separation algorithms for nonconvex polygons and their applications," *European Journal of Operational Research*, Vol. 84, 1995, pp. 539-561.

[45] C.D. Maranas, C.A. Floudas, and P.M. Pardalos, "New results in the packing of equal circles in a square," *Discrete Mathematics*, Vol. 142, 1995, pp. 287-293.

[46] M. Mckenna, J. O'Rourke, and S. Suri, "Finding maximal rectangles inscribed in an orthogonal polygon," *Proc. 23rd Allerton Conf. Commun. Control Comput.*, 1985, pp. 486-495.

[47] N. Megiddo and K.J. Supowit, "On the complexity of some common geometric location problems," *SIAM J. Comput.*, Vol. 13, No. 1, 1984, pp. 182-196.

[48] V.J. Milenkovic, "Position-based physics: Animating and packing spheres inside polyhedra," *Proc. 7th Canad. Conf. Comput. Geom.*, 1995, pp. 79-84.

[49] V.J. Milenkovic, "Translational polygon containment and minimal enclosure using linear programming based restriction," *Proc. of the 1996 ACM Symp. on the Theory of Computing (STOC)*, 1996, pp. 109-118.

[50] V.J. Milenkovic, "Multiple translational containment, part II: Exact algorithms," *Algorithmica*, Vol. 19, 1997, pp. 183-218.

[51] V.J. Milenkovic, "Densest translational lattice packing of non-convex polygons," *Proc. 16th ACM Annual Symp. on Computational Geometry (SCG)*, 2000, pp. 280-289.

[52] G.L. Miller, D. Talmor, S.-H. Teng, and N. Walkington, "A Delaunay based numerical method for three dimensions: Generation, formulation and partition," *Proc. 27th Annu. ACM Symp. Theory Comput.*, 1995, pp. 683-692.

[53] H. Minkowski, "Diskontinuitätsbereich für arithmetische Aequivalenz," *J. reine angew. Math.* 129, 1905, pp. 220-274.

[54] D.M. Mount and R. Silverman, "Packing and covering the plane with translates of a convex polygon," *Journal of Algorithms*, Vol. 11, 1990, pp. 564-580.

[55] K.J. Nurmela and P.R.J. Östergård, "Packing up to 50 equal circles in a square," *Discrete & Computational Geometry*, Vol. 18, 1997, pp. 111-120.

[56] C.A. Rogers, *Packing and Covering*, Cambridge Univ. Press, Cambridge, 1964.

[57] J. Ruppert, "A new and simple algorithm for quality 2-dimensional mesh generation," *Proc. 3rd Annu. ACM-SIAM Symp. on Discrete Algorithms*, 1992, pp. 83-92.

[58] J.R. Shewchuk, "Tetrahedral mesh generation by Delaunay refinement," *Proc. 14th Annu. ACM Symp. on Comput. Geom.*, 1998, pp. 86-95.

[59] A. Steinberg, "A strip-packing algorithm with absolute performance bound 2," *SIAM J. Comput.*, Vol. 26, No. 2, 1997, pp. 401-409.

[60] G.F. Tóth and W. Kuperberg, "A survey of recent results in the theory of packing and covering," Chapter in *New Trends in Discrete and Computational Geometry, Algorithms and Combinatorics*, Vol. 10, Springer-Verlag, 1993, pp. 251-279.

[61] J. Wang, "Packing of unequal spheres and automated radiosurgical treatment planning," *Journal of Combinatorial Optimization*, Vol. 3, 1999, pp. 453-463

[62] J. Wang, "Medial axis and optimal locations for min-max sphere packing," to appear in *Journal of Combinatorial Optimization.*

[63] Q.R. Wu, "Treatment planning optimization for Gamma unit radiosurgery," Ph.D. Thesis, The Mayo Graduate School, 1996.

[64] C.K. Yap, "An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments," *Discrete Comput. Geom.*, Vol. 2, 1987, pp. 365-393.