Approximation Algorithms for the Unit Disk Cover Problem in 2D and 3D

Ahmad Biniaz* Paul Liu† Anil Maheshwari* Michiel Smid*

February 12, 2016

Abstract

Given a set P of n points in the plane, we consider the problem of covering P with a minimum number of unit disks. This problem is known to be NP-hard. We present a simple 4-approximation algorithm for this problem which runs in $O(n \log n)$-time. We also show how to extend this algorithm to other metrics, and to three dimensions.

1 Introduction

In this paper we consider the *unit disk cover* (UDC) problem. Given a set P of n points in the plane, the UDC problem asks for the minimum number of disks of prescribed radius r (or simply unit disks of radius 1), which cover all points of P. Unless otherwise specified, we assume that the disks are in the L_2-norm.

The UDC problem is known to be NP-hard in the L_1, L_2, and L_∞ norms [8]. For points in $\mathbb{R}^d$ and any integer $l \geq 1$, it is possible to approximate the UDC problem in the L_2-norm within a factor of $\left(1 + \frac{1}{l}\right)^d$ with running time $O((dl)^{O(d)} n^{O((dl)^d)})$ [12] and within a factor of $2\left(1 + \frac{1}{l}\right)^{d\text{-}1}$ with running time $O((dl)^{O(d)} n^{O(d^d)})$ [11]. For points under the L_1 and L_∞ norms, similar ideas lead to a $\left(1 + \frac{1}{l}\right)^d$ approximation algorithm with running time $O(l^d n^{2l^d+1})$ [12] and a $\left(1 + \frac{1}{l}\right)^{d\text{-}1}$ approximation algorithm with running time $O(dl^{O(d\text{-}1)} n^{O(dl^{d\text{-}1})})$ [11]. However, these algorithms are mainly of theoretical interest, and are impractical for large data sets.

Gonzalez [11] presented a 2-approximation algorithm for the UDC problem in the L_1 and L_∞ norms and an 8-approximation in the L_2-norm. These algorithms run in $O(n \log S)$-time, where $S \leq n$ is the number of disks in an optimal solution. A constant approximation algorithm running in $O(n^3 \log n)$-time is also presented in [4]. The algorithm uses the fact that the UDC problem is equivalent to a set cover in a range space of finite VC dimension. However, no efforts were made to optimize or determine the exact value of the approximation factor. By constraining the disk centers to lie on a grid, Franceschetti et al. [9] developed, for any $l \geq 1$, an $O(Kn)$ time algorithm with approximation factor $3(1 + \frac{1}{l})^2$, where K is a function of l and the size of the approximation grid. A 2.8334-approximation algorithm which runs in $O(n(\log n \log\log n)^2)$ time is presented in [10]. We note that this algorithm is quite difficult to implement, and has a high constant factor in the running time. Using a different approach of dividing the input into vertical strips, Liu and Lu [13] presented a $\frac{25}{6}$-approximation algorithm for this problem running in $O(n \log n)$ time. A listing of all the algorithms as well as their approximation factors is given in Table 1.

*School of Computer Science, Carleton University, Ottawa, Canada. Research supported by NSERC.
†Department of Computer Science, University of British Columbia.

Reference	Approximation	Running Time
[11]	$2\left(1+\frac{1}{l}\right)$	$O(l^2 n^7)$
[11]	8	$O(n \log S)$
[4]	$O(1)$	$O(n^3 \log n)$
[9]	$3\left(1+\frac{1}{l}\right)^2$	$O(Kn)$
[10]	2.8334	$O(n(\log n \log\log n)^2)$
[13]	25/6	$O(n \log n)$
This paper	4	$O(n \log n)$

Table 1: A history of approximation algorithms for the unit disk cover problem in L_2.

There are numerous variants of the UDC problem. If the disk centers are constrained to an arbitrary point set Q, the UDC problem becomes the discrete unit disk cover problem (DUDC), which is also NP-hard. Many approximation algorithms are proposed for the DUDC problem, where the best known approximation factor is $9+\epsilon$ for any $0 < \epsilon \leq 6$ [2]. An instance of the UDC problem can be reduced to an instance of the DUDC problem as follows. Any solution for the UDC problem can be transformed (by moving the circles in the solution) to another solution in which each unit disk D has at least 2 input points on its boundary or an input point on its center; in the former case the center of D can be computed easily. Since each disk has unit radius, any pair of input points defines at most two possible centers for disks in our cover. Hence by choosing Q to be the union of P and these $O(n^2)$ centers, an instance of the DUDC problem is obtained. Thus, any approximation algorithm for the DUDC problem gives a solution for the UDC problem with the same approximation factor.

In the L_∞-norm, the UDC problem further reduces to the minimum clique cover problem [7]. The reduction uses the L_t *unit disk graph* on P. Each point in P corresponds to a vertex in the graph, and every edge (u, v) in the graph corresponds to intersecting L_t unit discs centered at u and v. Any family F of unit squares (L_∞ unit disks) satisfies Helly's property: if each pair of squares in F has a non-empty intersection, then the intersection of all squares in F is non-empty. Hence any clique in the L_∞ unit disc graph can be covered by a single L_∞ unit disc. Unfortunately, this reduction does not hold in the L_2-norm. The minimum clique problem on both the L_∞ and L_2 unit disk graphs has a large body of work, see [7] and the references contained therein.

We present an $O(n \log n)$-time constant-ratio approximation algorithm for the UDC problem in L_t-norms. In Section 2, we present a 4-approximation algorithm for this problem in the Euclidean norm (L_2-norm). By using the plane sweep technique, we show in Section 3 that this algorithm can be implemented to run in $O(n \log n)$ time. We emphasize that this algorithm is usable in practical settings and simple to implement. The most costly step is sorting of the points with respect to some dimension. In Section 4, we extend this algorithm to other L_t-norms. It is a 2-approximation for $t \in \{1, \infty\}$, a 6-approximation for $t > 2$, and a 5-approximation for $1 < t < 2$. We also extend this algorithm for points in three dimensions in Section 5; an instance of the UDC problem in three dimensions is known as unit ball covering. As a result, we obtain an $O(n \log n)$-time 12-approximation algorithm for the unit ball covering problem. Concluding remarks and open problems are given in Section 6.

2 A 4-Approximation Algorithm in L_2

In this section we consider the UDC problem in the Euclidean norm. Given a point set P in the plane, let C_{opt} be an optimal unit disk cover for P. Recall that the unit disks have radius

1. The *unit disk intersection graph*, UDIG(P), is defined to have the points of P as its vertices and has a straight-line edge between two points $p, q \in P$ if and only if $|pq| \leq 2$, where $|pq|$ is the Euclidean distance between p and q. We begin with the following observation:

Observation 1. *For two points $p, q \in P$, if $(p, q) \notin$ UDIG(P), then p and q cannot be covered by a unit disk.*

An *independent set* in UDIG(P) is a subset I of P such that there is no edge between any pair of points in I. I is said to be a *maximal independent set* if for all $p \in P \setminus I$, $I \cup \{p\}$ is not an independent set in UDIG(P).

Assume I is a maximal independent set in UDIG(P). By Observation 1, the size of any independent set in UDIG(P) is a lower bound for the number of disks needed to cover P. Therefore,

$$|I| \leq |C_{opt}|. \tag{1}$$

It is known that to cover a disk of radius 2, seven unit disks of radius 1 are necessary and sufficient; see Figure 1. Moreover, to cover a ball of radius 2 in three dimension, 21 unit balls are necessary and sufficient [1]. Based on that, a 7-approximation algorithm for the UDC problem is obtained as follows. Let I be any maximal independent set in UDIG(P). For a point $p \in I$, let $D(p, 2)$ be the disk of radius 2 which is centered at p. Let $d(p)$ be a disk in any unit disk cover which covers p. By Observation 1, none of the points of P which are at distance greater than 2 from p can be covered by $d(p)$. Therefore, all points of P which are not in $D(p, 2)$ must be covered by disks different from $d(p)$. Moreover, all points of P which are covered by $d(p)$ are in $D(p, 2)$. Therefore, by covering $D(p, 2)$ with seven unit disks (Figure 1), for all $p \in I$, a 7-approximation algorithm is obtained. Note that UDIG(P) may have up to $O(n^2)$ edges, and hence the time complexity of computing UDIG(P) is quadratic in the worst case.

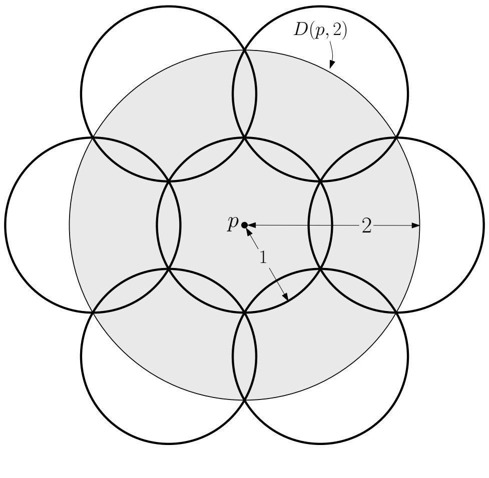


Figure 1: $D(p, 2)$ can be covered by 7 unit disks.

Now we show how to reduce the approximation ratio to 4. Let p be the leftmost point in P. In case of degeneracy, we consider the leftmost point with the smallest y-coordinate. Let ℓ be the vertical line passing through p. Let $R(p)$ be the intersection of $D(p, 2)$ with the half-plane to the right of ℓ, i.e., $R(p)$ is the right half-disk of $D(p, 2)$ (see Figure 2(a)). As discussed earlier, all points of P which are covered by $d(p)$ are in $D(p, 2)$ and consequently in $R(p)$. As shown in Figure 2(a), $R(p)$ can be covered by 4 unit disks. Figure 2(b) shows a configuration of seven points in $R(p)$ such that at least four unit disks are needed to cover all these seven

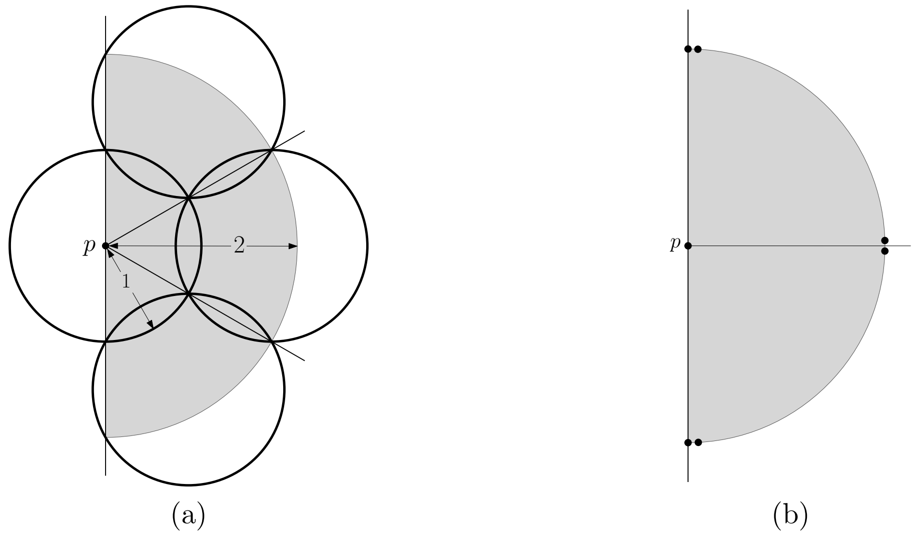


Figure 2: (a) Any half-disk of radius 2 can be covered by four unit disks. (b) Seven points in a half-disk of radius 2 which cannot be covered by less than four unit disks.

points: in any unit disk cover, the disk which covers p can cover at most one of the points on the boundary. The remaining five points need at least three unit disks to be covered.

For a point p and a given point set I, the *distance*, $\mathrm{d}(p, I)$, between p and I is defined as the minimum Euclidean distance between p and any point in I, i.e., $\mathrm{d}(p, I) = \min\{|pq| : q \in I\}$. If $I = \emptyset$, then $\mathrm{d}(p, I) = \infty$. Our 4-approximation algorithm is given in Algorithm 1. The output of this algorithm is a set C of unit disks that cover P. The algorithm starts by creating a sorted list of points from left to right. Then it repeatedly selects and deletes the first element in the list, say p. If $\mathrm{d}(p, I) \leq 2$, then p is already covered by some disk in C. Otherwise, i.e., if $\mathrm{d}(p, I) > 2$, the algorithm covers $R(p)$ by four unit disks, and adds them to C. Finally it returns the set C of unit disks.

Algorithm 1 UNITDISKCOVER(P)

Input: A point set P in the plane.
Output: A set C of unit disks that cover P.

```
 1: $C \leftarrow \emptyset$
 2: $I \leftarrow \emptyset$
 3: $L \leftarrow$ list of points in $P$ sorted from left to right
 4: while $L$ is not empty do
 5:     $p \leftarrow$ first element of $L$
 6:     if $\mathrm{d}(p, I) > 2$ then
 7:         Cover $R(p)$ by four unit disks $c_1, c_2, c_3, c_4$
 8:         $C \leftarrow C \cup \{c_1, c_2, c_3, c_4\}$
 9:         $I \leftarrow I \cup \{p\}$
10:     $L \leftarrow L - \{p\}$
11: return $C$
```

In each iteration, Algorithm 1, adds p to I if and only if $\mathrm{d}(p, I) > 2$. Thus, in UDIG(P), p is not connected to any point in I. Therefore, I is an independent set in UDIG(P). In addition, the while loop iterates over all points. Thus, after Algorithm 1 terminates, I is a maximal independent set in UDIG(P).

Theorem 1. *Algorithm 1 is a 4-approximation for the unit disk cover problem.*

Proof. Consider the set I of points and the set C of unit disks after the termination of Algorithm 1. Since I is a maximal independent set in UDIG(P), by Inequality (1) we have $|I| \leq |C_{opt}|$. Each point $q \in P$ is in a half-disk $R(p)$, for some $p \in I$ (possibly $q = p$). Since for each $p \in I$, we cover $R(p)$ with four unit disks, C covers P. Moreover, $|C| \leq 4|I| \leq 4|C_{opt}|$. This proves the statement of the theorem. $\square$

The running time of Algorithm 1, can be expressed as $O(n \log n + n \cdot t(\mathrm{d}))$, where $t(\mathrm{d})$ is the time for computing $\mathrm{d}(p, I)$. Any nearest-neighbor data structure is sufficient here, and only insertions and queries are needed. As the nearest-neighbor problem is a decomposable search problem, the general technique of Bentley and Saxe [3] gives an $O(\log^2 n)$-amortized time bound for both insertions and queries, and uses only $O(n)$-space. Using this data structure, $\mathrm{d}(p, I)$ can be computed in $O(\log^2 n)$-amortized time, and hence Algorithm 1 can be implemented to run in $O(n \log^2 n)$-time.

3 Improving the Time Complexity

Instead of computing $\mathrm{d}(p, I)$ dynamically, we can speed up Algorithm 1 by taking advantage of the fact that we only need to check if $\mathrm{d}(p, I)$ is greater than 2. Every time we add a new point p to I in Algorithm 1, we are essentially removing every point in P lying in $R(p)$. We can do this in $O(n \log n)$-time with a simple sweep-line algorithm.

We sweep a vertical line from left to right and maintain a binary search tree (BST) storing the centers of all the half-disks intersecting the sweep line. The points in BST are sorted in non-decreasing order of their y-coordinates. In case of ties, we sort them in increasing order of their x-coordinates. Since all half-disks have radius 2, they are uniquely defined by their centers which are stored in BST. Initially BST is empty.

We also keep an event queue that stores two types of events: *site events* and *deletion events.* A site event is a point of P. Each deletion event is associated with a site event; for each point $p \in P$ its deletion event is the rightmost point of $R(p)$. Thus, for every point $p = (p_x, p_y)$ in P, we have a deletion event $p' = (p_x + 2, p_y)$. The event queue is kept as a priority queue sorted by the x-coordinates of the events. Initially we add to the event queue each point $p \in P$ as a site event and p' as a deletion event. At each step of the sweep algorithm, we pop the event with the smallest x-coordinate from the queue, and "move" the sweep-line to that point.

Deletion events are straight-forward to handle, as we remove the center of the half-disk —which corresponds to this event—from BST.

Now we describe how to handle the site events. Let p be the current site event which is encountered by the sweep-line SL. If p is covered by a half-disk in BST, then we proceed to the next event. If p is not covered by any half-disk in BST, then we insert a new half-disk (its center) into BST. Since the half-disks in BST have radius 2, we have the following observation:

Observation 2. *The distance between any two points in BST is more than 2.*

Note that the half-disks corresponding to the points of P that are to the left of SL and are not in BST do not intersect SL. Therefore, these points have distance bigger than 2 from SL, and p cannot be covered by their half-disks.

In order to check if p is covered by any half-disk intersecting the sweep-line we do the following. We search for p in BST by its y-coordinate. Let p^- and p^+ be the predecessor and the successor of p in BST, respectively. In other words, p^- is the point in BST with the largest y-coordinate and p^+ is the point in BST with the smallest y-coordinate such that $p^-_y < p_y < p^+_y$.

If $|pp^-| \leq 2$ (or $|pp^+| \leq 2$), then p is covered by $R(p^-)$ (or $R(p^+)$). However, this may not be the only case to decide if p is covered by a half-disk in BST. As shown in Figure 3(a), p is covered by a half-disk which is neither $R(p^-)$ nor $R(p^+)$.

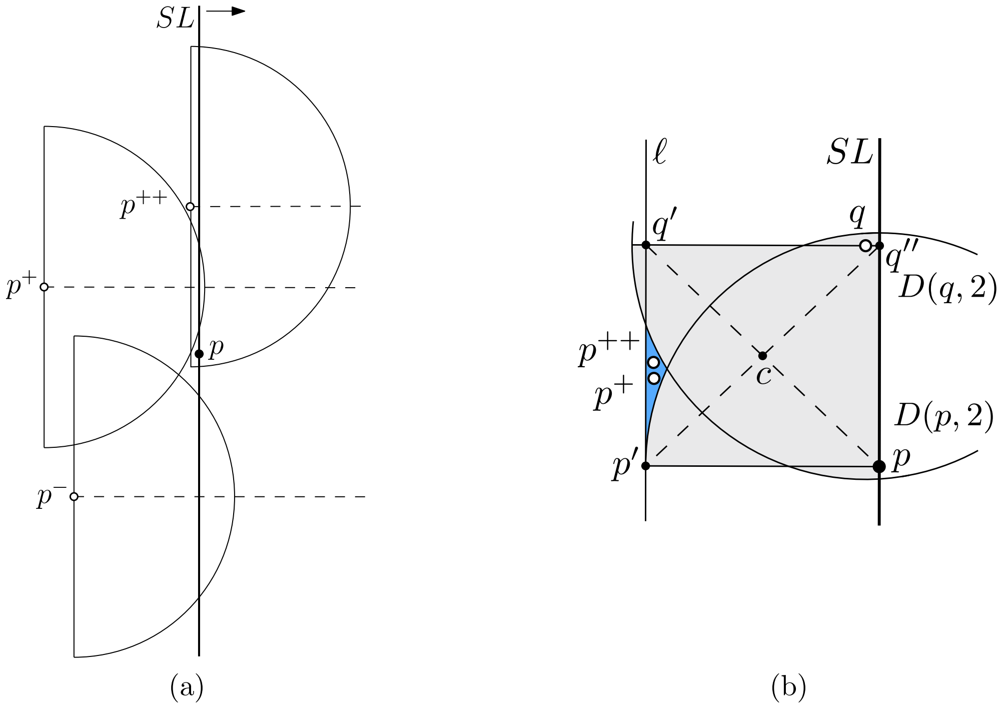


Figure 3: (a) p is covered by a half-disk other than $R(p^-)$ and $R(p^+)$. (b) Proof of Lemma 1

Let p^{--} be the predecessor of p^- and p^{++} be the successor of p^+ in BST.

Lemma 1. *If p is covered by any half-disk intersecting the sweep line, then $p \in R(p^{--}) \cup R(p^-) \cup R(p^+) \cup R(p^{++})$.*

Proof. The proof is by contradiction. Assume p is covered by a half-disk $R(q)$ which is centered at a point q in BST while $p \notin R(p^{--}) \cup R(p^-) \cup R(p^+) \cup R(p^{++})$. Without loss of generality assume $q_y \geq p_y$. Since p^+ is the successor of p and p^{++} is the successor of p^+ in BST, we have $q_y \geq p_y^{++}$. Let l be the vertical line which is at distance 2 from p and to the left of the sweep line SL; see Figure 3(b). All points in BST (including p^+, p^{++}, and q) lie between (or on) l and SL.

Let p' be the intersection point of l and the horizontal line passing through p. Let q' (resp. q'') be the intersection point of l (resp. SL) and the horizontal line passing through q. See Figure 3(b). Let R be the rectangle having its corners on p, p', q' and q''. Observe that the maximum side length for R is 2.

Since $p_y \leq p_y^+ \leq p_y^{++} \leq q_y$, p^+ and p^{++} lie in R. Consider $D(p,2)$ and $D(q,2)$. Since $p \in R(q)$, $|pq| \leq 2$; this implies that $p, q \in D(p,2) \cap D(q,2)$. By Observation 2, both p^+ and p^{++} are outside $D(q,2)$. In addition, p is to the right of p^+ and to the right of p^{++} and $p \notin R(p^+) \cup R(p^{++})$, which implies that both p^+ and p^{++} are outside $D(p,2)$. Therefore p^+ and p^{++} lie in region $Q = R - (D(p,2) \cup D(q,2))$; the blue region in Figure 3(b). Let c be the intersection point of the two diagonals of R. The triangle $\triangle pq'q''$ is a subset of $D(q,2)$ and the triangle $\triangle pp'q''$ is a subset of $D(p,2)$. Thus, Q is a subset of the triangle $\triangle cp'q'$. $\triangle cp'q'$ has diameter at most 2. Thus, the distance between any two points in Q is at most 2. Therefore, $|p^+p^{++}| \leq 2$; which contradicts Observation 2. $\square$

Given a site event p, in $O(\log n)$-time we can find p^{--}, p^{-}, p^{+}, and p^{++} in BST. In order to check if p is in the coverage of any point in BST, by Lemma 1, it is enough to check if the distance of p to p^{--}, p^{-}, p^{+}, or p^{++} is at most 2. Therefore, each site event can be handled in $O(\log n)$-time; each deletion event can be handled in $O(\log n)$-time as well. Since we have $2n$ events, we conclude that Algorithm 1 can be implemented to run in $O(n \log n)$-time and $O(n)$-space.

4 Extensions to other Metrics

In this section we extend the algorithm presented in Section 2 to other metrics. We consider the unit disk cover problem for a point set P in the L_t-norm, for $t \geq 1$. In the L_t-norm, a unit circle which is centered at the origin is expressed by the equation

$$|x|^t + |y|^t = 1.$$

Figure 4 shows the unit circles in different L_t-norms. We refer to the union of a unit circle in the L_t-norm and its interior as an L_t-unit disk.

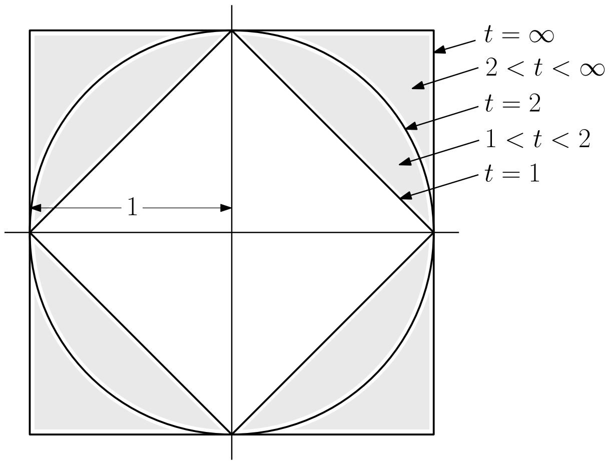


Figure 4: Illustration of unit circles in different L_t-norms.

Observation 3. *For any t and t', with $1 \leq t < t' \leq \infty$, the L_t-unit disk which is centered at the origin is contained in the $L_{t'}$-unit disk which is centered at the origin.*

Let $D_t(p, 2)$ be the L_t-unit disk which is centered at point p and scaled by a factor of 2. Observe that any L_t-unit disk which covers p, does not cover any point outside $D_t(p, 2)$. Let $R_t(p)$ be the right half-disk of $D_t(p, 2)$. By Observation 3, $R_t(p)$ is contained in $R_\infty(p)$.

4.1 L_t for $t \geq 2$

Assume $t \geq 2$. As shown in Figure 5(a), $R_\infty(p)$ can be covered by six L_2-unit disks. Since $R_t(p) \subseteq R_\infty(p)$, $R_t(p)$ can also be covered by six L_2-unit disks. By Observation 3, any L_2-unit disk is contained in an L_t-unit disk. Thus, $R_t(p)$ also can be covered by six L_t-unit disks. Therefore, a modified version of Algorithm 1 gives an L_t-unit disk cover C for P such that $|C| \leq 6|C_{opt}|$.

Since an L_t-unit disk contains an L_2-unit disk, Lemma 1 can be extended to the L_t-norm:

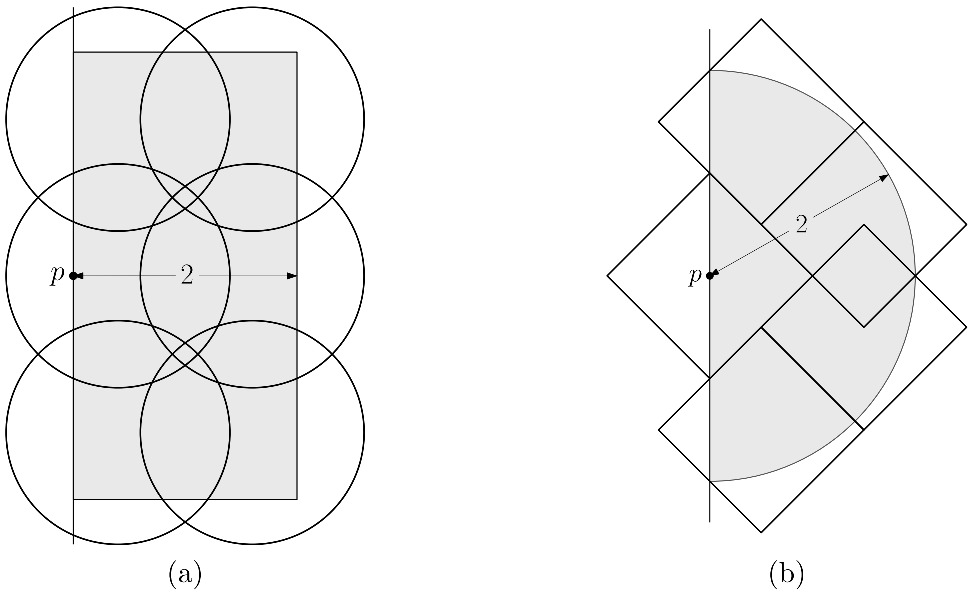


Figure 5: (a) $R_\infty(p)$ which is covered by six L_2-unit disks. (b) $R_2(p)$ which is covered by five L_1 unit disks.

Lemma 2. *If p is covered by any L_t-half disk intersecting the sweep line, then $p \in R_t(p^{--}) \cup R_t(p^-) \cup R_t(p^+) \cup R_t(p^{++})$.*

Therefore, an $O(n \log n)$-time 6-approximation algorithm for the UDC problem in the L_t-norm, where $t \geq 2$, is obtained.

4.2 L_t for $1 \leq t \leq 2$

Assume $1 \leq t \leq 2$. As shown in Figure 5(b), $R_2(p)$ can be covered by five L_1-unit disks. By Observation 3, $R_t(p)$ is contained in $R_2(p)$. In addition, an L_1-unit disk is contained in an L_t-unit disk. Thus, $R_t(p)$ can also be covered by five L_t-unit disks. Therefore, a modified version of Algorithm 1 gives an L_t-unit disk cover C for P such that $|C| \leq 5|C_{opt}|$. Lemma 1 can be extended to the L_1-norm as follows.

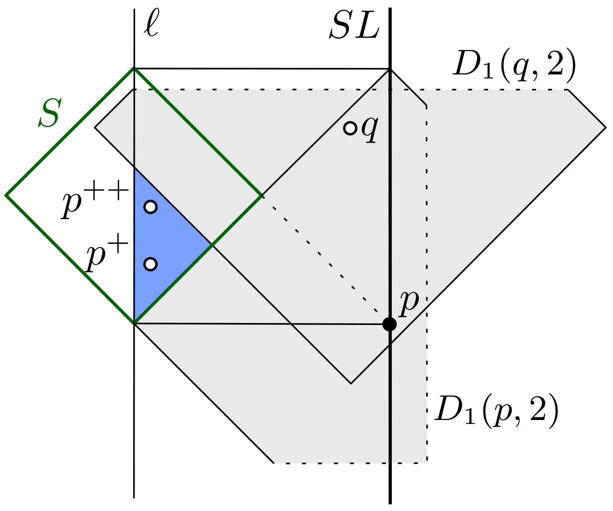
Lemma 3. *In L_1-norm, if p is covered by any half-disk intersecting the sweep line, then $p \in R_1(p^{--}) \cup R_1(p^-) \cup R_1(p^+) \cup R_1(p^{++})$.*

Proof. The proof is by contradiction; and similar to the proof of Lemma 1. We skip the details. Consider $D_1(p, 2)$ and $D_1(q, 2)$. Note that both p^+ and p^{++} are outside $D_1(p, 2) \cup D_1(q, 2)$. See Figure 6(a). Therefore p^+ and p^{++} lie in region $Q = R - (D_1(p, 2) \cup D_1(q, 2))$, where R is a unit square which has its bottom-right corner on p. As shown in Figure 6(a), Q (the blue region) can be covered by the L_1-unit disk S. Therefore, the L_1-distance between p^+ and p^{++} is at most 2; which contradicts Observation 2. $\square$

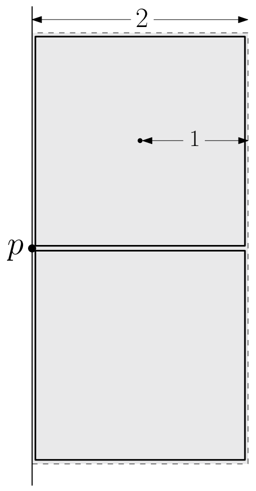
Since an L_t-unit disk contains an L_1-unit disk, Lemma 3 can be extended to the L_t-norm. Therefore, an $O(n \log n)$-time 5-approximation algorithm for the UDC problem in the L_t-norm, where $1 \leq t \leq 2$, is obtained.

4.3 L_∞ and L_1

Assume $t = \infty$. An L_∞-unit disk is an axis-aligned square of side length 2. As shown in Figure 6(b), $R_\infty(p)$ can be covered by two L_∞-unit disks. Therefore, a modified version of



(a)



(b)

Figure 6: (a) Illustration of Lemma 3. (b) $R_\infty(p)$ which is covered by two L_∞ unit disks.

Algorithm 1 gives an L_∞-unit disk cover C for points in P such that $|C| \leq 2|C_{opt}|$. In addition, we have the following Lemma, which is stronger than Lemma 1.

Lemma 4. *If p is covered by any L_∞-half disk intersecting the sweep line, then $p \in R_\infty(p^-) \cup R_\infty(p^+)$.*

Therefore, a simple $O(n \log n)$-time 2-approximation algorithm for the UDC problem in the L_∞-norm is obtained. Gonzalez [11] presented a faster $O(n \log S)$-time 2-approximation algorithm for this problem, where S is the size of an optimal solution.

The UDC problem in the L_1-norm can easily be reduced to a UDC problem in the L_∞-norm by simply rotating the x and y axes by 45° around the origin, followed by scaling with $\sqrt{2}/2$. Therefore, a simple $O(n \log n)$-time 2-approximation algorithm for the UDC problem in L_1 is obtained.

5 Unit Ball Cover Problem in $\mathbb{R}^3$

In this section we consider the *unit ball cover* (UBC) problem. Given a set P of n points in $\mathbb{R}^3$, the UBC problem asks for the minimum number of unit balls (balls of radius 1) which cover all points of P. We show how to extend Algorithm 1 to cover the points in P. We sweep the space by a plane; instead of covering half-disks, here we cover half-balls.

To cover a ball of radius 2 in three dimension, twenty-one unit balls are necessary and sufficient [1]. Therefore, we can obtain a 21-approximation algorithm for the UBC problem by using Observation 1. However, instead of covering a ball of radius 2, as we have seen in Algorithm 1, it is sufficient to cover a half-ball of radius 2. Let $B(p,2)$ be the ball of radius 2 which is centered at a point p. Any plane which passes through p, divides $B(p,2)$ into two half-balls, say $L(p)$ and $R(p)$. Consider a covering of $B(p,2)$ with 21 unit balls, as in [1]. We find a plane H which passes through p such that one of the half-spaces, on each side of H, contains 7 unit balls. Assume $L(p)$ is in the same half-space as these 7 unit balls. Thus, $R(p)$ is covered by the remaining 14 unit balls. We can use this covering of $R(p)$ to cover any half-ball of radius 2. Therefore, a half ball of radius 2 in $\mathbb{R}^3$ can be covered by 14 unit balls. See Figure 7 for such a covering: the half-ball is centered at the origin and has its base on xy-plane; the coordinates of the centers of the unit balls are given in Appendix A.

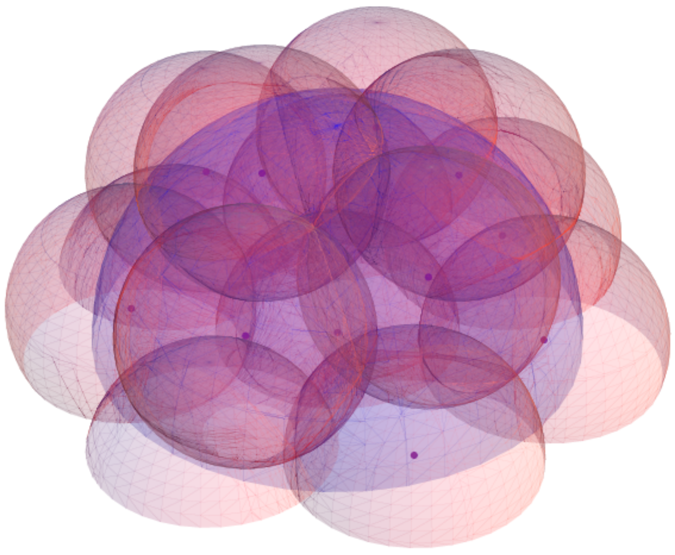


Figure 7: Covering a half-ball of radius 2 by 14 unit balls.

Let p be the current site event. In Algorithm 1, in order to check $\mathrm{d}(p, I) > 2$, it is sufficient to check $B(p, 2)$ does not contain any point of I; this is a ball emptiness query. A ball emptiness query in $\mathbb{R}^3$ can be transformed to a half-space emptiness query in $\mathbb{R}^4$ by projecting the points of P to the paraboloid $x_4 = x_1^2 + x_2^2 + x_3^2$. Chan [5] presented a linear-size data structure which can be constructed in $O(n \log n)$-time that answers half-space emptiness queries in $\mathbb{R}^4$ in $O(\sqrt{n})$-time. Based on the techniques of Bentley and Saxe [3], this gives an insertion-only dynamic data structure which supports insertions and half-space emptiness queries in $\mathbb{R}^4$ in $O(\sqrt{n} \log n)$-amortized time. Let I' be the set of points obtained by projecting the points of I to $\mathbb{R}^4$, and let H be the half-space obtained by projecting $B(p, 2)$ to $\mathbb{R}^4$. We store the points of I' in an insertion-only dynamic data structure. In order to check $B(p, 2)$ is empty, we check the emptiness of H. Therefore, an $O(n\sqrt{n} \log n)$-time 14-approximation algorithm for the UBC problem is obtained.

5.1 Improving the Approximation Factor

Through a combination of both manual and automated searching, we have found a covering of the half-ball with 12 unit balls. See Figure 8; the coordinates of the centers of the unit balls are given in Appendix A. We give a basic description of the search and the verification below.

Constructing the covering The boundary of a half-ball consists of a hemisphere *surface* and a *base* disk. Let H, H', and h be the half-balls of radius 2, $\sqrt{3}$, and 1, respectively, which are centered at the origin and have their base on the xy-plane. Note that $h \subset H' \subset H$. Let S, S', and s be the surfaces of H, H', and h, respectively. We show how to cover H with a set $\mathcal{B}$ of eleven balls that are centered on S' and a ball b that is centered at the origin. Note that h is covered by b, and we have to make sure $H - h$ is covered by $\mathcal{B}$. Each ball in $\mathcal{B}$ creates a cap on H with contact angle of 60°. They also create caps of the same contact angle on h. Hence, if S is covered by the balls in $\mathcal{B}$, then s is covered, and because of the convexity, $H - h$ is also covered. Thus, in order to verify the covering, it suffices to check if $\mathcal{B}$ covers S (or s). To find the cover itself, we adjust the centers of the balls in $\mathcal{B}$ manually until they cover S. The

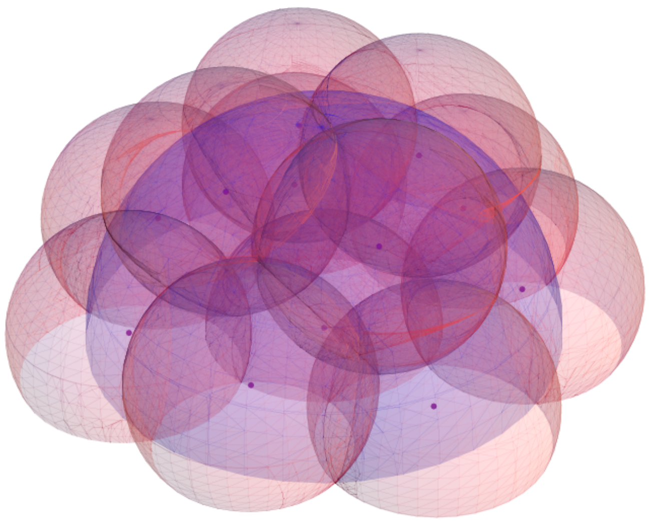


Figure 8: Covering a half-ball of radius 2 by 12 unit balls.

covering is verified by the following procedure.

Verifying the covering Let $\mathcal{B} = \{B_1, \ldots, B_{11}\}$. For each $B_i \in \mathcal{B}$, let C_i be the circle on S which is the intersection of the surface of B_i with S; here we assumed S is the surface of the whole ball of radius 2. Let $\mathcal{C} = \{C_1, \ldots, C_{11}\}$. We choose $\mathcal{B}$ such that each circle $C_i \in \mathcal{C}$ intersects some circle $C_j \in \mathcal{C}$, with $i \neq j$, in two distinct points. Let C be the boundary circle of the base of H. Let I be the set of intersection points between every two circles in $\mathcal{C} \cup \{C\}$ which are on S.

Lemma 5. *If S is not covered by $\mathcal{B}$, then there exists a point of S at distance ϵ from a point of I that is also not covered by $\mathcal{B}$, for some $\epsilon > 0$.*

Proof. Let p be a point on S that is not covered by $\mathcal{B}$. Let C_i be the circle in $\mathcal{C}$ which has the smallest spherical distance to p, where the spherical distance between C_i and p is the length of the shortest path between C_i and p on S. Let B_i be the ball in $\mathcal{B}$ which corresponds to C_i, and let c be the center of B_i. Let B'_i to be the ball of radius $1 + \epsilon$ centered at c. Let C'_i be the circle on S which is the intersection of the surface of B'_i with S. Let p' be the point of C'_i that has the smallest spherical distance to p. See Figure 9. Observe that no point of the arc pp' is covered by $\mathcal{B}$. Since C_i intersects some circles of $\mathcal{C}$, by picking ϵ to be small enough, we make sure that C'_i also intersects the same circles of $\mathcal{C}$. Let C_j be the first circle of $\mathcal{C}$ which is intersected by C'_i while walking on C'_i in counter-clockwise (or clockwise) from p'. Let p'' be the point on the boundary of C'_i just before intersecting C_j. Observe that no point of the arc $p'p''$ is covered by $\mathcal{B}$. Since p'' is not covered by $\mathcal{B}$ and it is at distance ϵ from an intersection point of C_i and C_j, the claim follows. $\square$

By Lemma 5, if S is not covered by $\mathcal{B}$, then there exists an open neighborhood of a point in I that is not covered by $\mathcal{B}$. Thus, in order to verify the covering, we check each point in I to be strictly within a ball of $\mathcal{B}$. The entire verification procedure is done in rational arithmetic, with no possible rounding errors. From the results of our search, we believe that 12 unit balls are necessary and sufficient to cover the half-ball of radius 2, but we do not yet have a proof of this. The lower bound is eleven as twenty-one unit balls are necessary to cover an entire ball of radius 2.

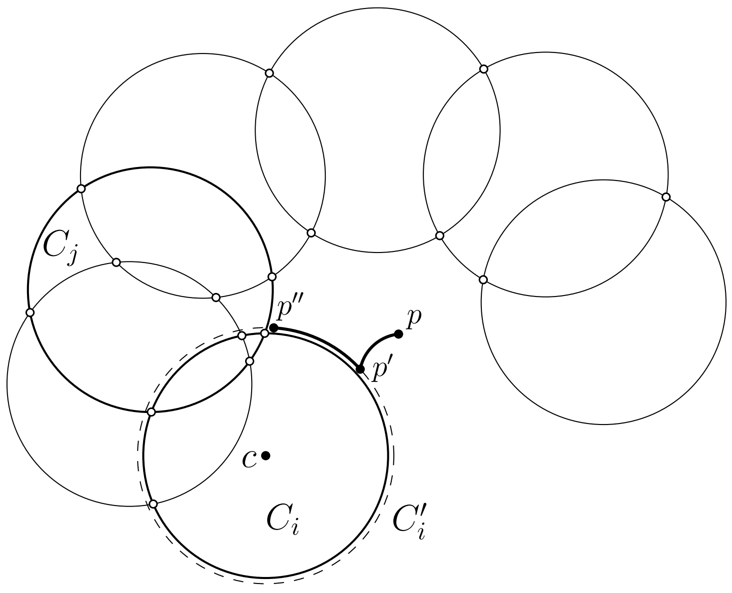


Figure 9: The intersection points between circles belong to I. No point of the line segment pp' and the arc $p'p''$ is covered by $\mathcal{B}$.

5.2 Improving the Time Complexity

In this section we show how to improve the running time by using a dynamic rectangular fixed windowing data structure (DRW). This data structure maintains a dynamic set, P, of n points in the plane to support fixed window-queries, while allowing points to be inserted into and deleted from P. Given an orthogonal rectangle W, a fixed window-query is to report the points in $W' \cap P$ where W' is an arbitrary translate of W. In [6] the authors presented an $O(n)$-space dynamic rectangular fixed windowing data structure which supports insert and delete operations in $O(\log n)$ time and search operations in $O(\log n + k)$ time, where k is the size of the output.

We sweep the space by a plane H—which is perpendicular to z-axis—from the point with the smallest z-coordinate to the point with the largest z-coordinate. We maintain a DRW storing the points whose half-balls intersect H. The points are stored in DRW by their x and y coordinates. Initially DRW is empty. As in Section 3, we keep track of an event queue which stores the site and deletion events. For each point $p \in P$ we have a site event at p and a deletion event at $p' = (p_x, p_y, p_z + 2)$. The event queue is sorted by the z-coordinates of the events. Deletion events are easy to handle. When H encounters a deletion event p', we remove its corresponding site event, p, from DRW; this takes $O(\log n)$ time. If p is not in DRW, then we proceed to the next event. Deletion events ensure that the points whose half-ball do not intersect H, are not in DRW. Based on that, the z-coordinate of the points in DRW is in the range $[p'_z - 2, p'_z)$.

Now, we describe how to handle the site events. Assume H encounters a site event p. If p is covered by any half-ball intersecting H, then we proceed to the next event, otherwise, we insert p into DRW. This ensures that the distance between any two points in DRW is more than 2. In order to check if p is covered by any half-ball intersecting H, it is enough to check $B(p, 2)$ contains a point of DRW. Let $C(p)$ be the smallest axis-aligned cube that contains $B(p, 2)$. Observe that $C(p)$ is a $4 \times 4 \times 4$ cube. Let $S(p)$ be the projection of $C(p)$ to H. Observe that $S(p)$ is a 4×4 square. Now we query DRW by the window $S(p)$; let Q be the set of all points in DRW falling in $S(p)$. If $B(p, 2)$ contains a point q in DRW, then $q \in Q$. Observe that in $\mathbb{R}^3$, the points of Q fall in an axis-aligned $2 \times 4 \times 4$ cuboid one of its faces is $S(p)$. Based on that and since the mutual distances between the points in Q are more than 2, we argue that Q has a constant number of points. Thus, the window query $S(p)$ takes $O(\log n)$ time. In addition, in a constant time we can check if $B(p, 2)$ contains a point of Q, and subsequently, a point of DRW. Thus, each site event can be handled in $O(\log n)$ time. Therefore, we can modify Algorithm 1 to obtain a 12-approximation for the unit ball covering problem in $\mathbb{R}^3$ in $O(n \log n)$ time and $O(n)$ space.

6 Conclusion

We considered the NP-hard problem of covering n given points in the plane with the minimum number of unit disks. We presented an easily implementable 4-approximation algorithm which runs in $O(n \log n)$-time and $O(n)$-space. The presented algorithm is faster than previous algorithms having a similar approximation ratio. It is interesting that the most time consuming step of the algorithm is sorting and maintaining a BST.

We extended the algorithm to other L_t-norms. As a result we obtained $O(n \log n)$-time algorithms; a 2-approximation for $t \in \{1, \infty\}$, a 6-approximation for $t > 2$, and a 5-approximation for $1 < t < 2$. We also extended the presented algorithm to cover the points in three dimensions with unit balls. As a result, an $O(n)$-space and $O(n \log n)$-time 12-approximation algorithm for the unit ball covering problem is obtained.

The natural problem is to reduce the approximation ratio of the presented algorithms, while not increasing the running time. Another open problem is to prove that twelve unit balls are necessary and sufficient to cover a half-ball of radius two.

References

[1] J. O'Rourke. Covering a unit ball with balls half the radius. http://www.mathoverflow.net/questions/98007/covering-a-unit-ball-with-balls-half-the-radius.

[2] R. Acharyya, M. Basappa, and G. K. Das. Unit disk cover problem in 2D. In *Proceedings of 13th Int. Conf. in Comput. Sci. and its App.-ICCSA*, pages 73–85, 2013.

[3] J. L. Bentley and J. B. Saxe. Decomposable searching problems. I. Static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

[4] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

[5] T. M. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012.

[6] R. Klein, O. Nurmi, T. Ottmann, and D. Wood. A Dynamic Fixed Windowing Problem. *Algorithmica*, 4(4):535–550, 1989.

[7] M. De, G. K. Das, and S. C. Nandy. Approximation algorithms for the discrete piercing set problem for unit disks. In *Proceedings of the 23rd Annual Canadian Conf. on Comput. Geom.*, 2011.

[8] R. J. Fowler, M. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3), 1981.

[9] M. Franceschetti, M. Cook, and J. Bruck. A geometric theorem for approximate disk covering algorithms. Technical report, 2001.

[10] B. Fu, Z. Chen, and M. Abdelguerfi. An almost linear time 2.8334-approximation algorithm for the disc covering problem. In *Proceedings of 3rd International Conference of Algorithmic Aspects in Information and Management*, pages 317–326, 2007.

[11] T. F. Gonzalez. Covering a set of points in multidimensional space. *Inf. Process. Lett.*, 40(4):181–188, 1991.

[12] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

[13] P. Liu and D. Lu. A fast 25/6-approximation for the minimum unit disk cover problem. *CoRR*, abs/1406.3838, 2014.

A Appendix

The coordinates of the points for 14 and 12 covering is given in Tables 2 and 3, respectively. Here we assume the half-ball has radius 1 and the unit balls have radius 1/2. The half-ball is centered at the origin and has its base on xy-plane. Since the coordinates of the points in 14 covering are obtained from [1], some of z-coordinates are negative.

Table 2: The coordinates of the 14 cover, and the excluded balls of the 21 cover.

ball #	x	y	z
1	-0.7042476993699713	-0.1897384908139258	0.4669408193044991
2	-0.3292862314363903	-0.7679448770185321	0.22766449373236058
3	0.5967710354972143	0.024432485403975968	0.6271102565247735
4	-0.803599430801605	-0.3050361127732187	-0.10573681771048753
5	0.6442044964439396	0.5624162836729194	0.1367060371671741
6	-0.11704300843515354	-0.27387869213520005	0.8131980518986511
7	-0.03634386701799877	0.8543361223697555	0.13707210477113296
8	-0.39301629040814245	0.449001371543003	0.6276427110198327
9	0.3327167675185201	-0.7924581713220512	-0.10634783351783872
10	0.8423378833005855	-0.19947886977538137	0.025992391460308913
11	0.3483067871797808	-0.641230866192109	0.46637530683454576
12	0.1931042097097512	0.4492843966181599	0.7147408565360098
13	-0.725047439878574	0.472849050034495	0.02683527916757068
14	0	0	0
1	0.27777845357333375	0.64820988747711	-0.5026562053076835
2	-0.0002474206056868669	0.00050958641266698074	-0.8660248031983451
3	-0.31458497985994993	0.6332628300547775	-0.5000151029724296
4	0.42206181844876095	-0.46795691910398096	-0.5940369529342732
5	-0.2785742232734966	-0.6488076234832963	-0.5014436157480976
6	0.6752035014327228	0.20869474927712645	-0.500547429119508
7	-0.6304928571982162	-0.016465523803227772	-0.5934713797150755

Table 3: The coordinates of the centers of 12 unit balls.

ball #	x	y	z
1	-0.7638103193805045	0.2916707987226469	0.28552047418765747
2	-0.7361984406498542	-0.4102678330060574	0.19922891653883215
3	-0.15685273236392883	-0.7980104830689634	0.297618025767945
4	0.5429274132864615	-0.6551216417858168	0.16138605381484397
5	0.8231862798974567	0.027307901405844496	0.2676165673298175
6	0.5069450704690941	0.6715187748330166	0.2051078510780353
7	-0.2619940422412506	0.8087354742390338	0.1652454372666754
8	-0.3625112845808338	-0.1713273372287619	0.7676148201211697
9	-0.185959221348503	0.5374732209436377	0.6531016037064448
10	0.3644368797245205	0.22772543684531862	0.7518822288831304
11	0.28771813222122117	-0.3895392199254971	0.7179675985244508
12	0	0	0