Indian Statistical Institute
Kolkata

M.Tech. (Computer Science) Dissertation

Facility Location on Spherical Surfaces

A dissertation submitted in partial fulfillment of the requirements
for the award of Master of Technology
in
Computer Science

Author:
Subhadeep R Dev
Roll No: MTC1325

Supervisor:
Prof. Sandip Das
ACMU

M.TECH(CS) DISSERTATION THESIS COMPLETION CERTIFICATE

Student: Subhadeep R Dev (MTC1325)
Topic: Facility Location on Spherical Surfaces
Supervisor: Prof. Sandip Das

This is to certify that the thesis titled "Facility Location on Spherical Surfaces" submitted by Subhadeep R Dev in partial fulfillment for the award of the degree of Master of Technology is a bona fide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other university for the award of any degree or diploma.

Prof. Sandip Das

Date : 13^{th} $July$**,** 2015

Acknowledgements

At the end of my dissertation and my M.Tech training at the Indian Statistical Institute, Kolkata, I want to thank and give credit to all individuals who have provided me with invaluable assistance. Whether be it gentle guidance or access to materials or services that helped me a lot in my research work, it is greatly appreciated.

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Sandip Das, who has supported me throughout my thesis with his patience and knowledge. It was a memorable learning experience. For his patience, for all his advice and encouragement and for the way he helped me to think about problems with a broader perspective, I will always be grateful. One simply could not wish for a better or friendlier supervisor.

I would like to thank all the professors at the Indian Statistical Institute Kolkata who have made my educational life exciting and helped me to gain a better outlook on Computer Science. I would like to express my gratitude to Harmender, Archan, and Sanjana for interesting discussions.

I would like to thank everybody at Indian Statistical Institute, Kolkata for providing a wonderful atmosphere for pursuing my studies. I thank all my classmates, seniors and juniors who have made the academic and non-academic experiences very delightful.

My most important acknowledgement goes to my family and friends who have filled my life with happiness.

Abstract

The facility location problem in Computational Geometry deals with the placement of facilities with respect to some optimizing criteria. One of the most common facility location problems is the k-center problem or the minimax problem. The problem is defined as, given n sites, we need to find k facilities such that the maximum distance of any site to its nearest facility is the minimum over all placements of the k facilities. More formally, let C be the set of sites, then the problem is to find a set of facilities F such that the value $p = max_{f \in F}(min_{c \in C}(d(f, c)))$ is minimized; $d(f, c), f \in F$ and $c \in C$ is the distance metric. This problem is NP-hard if the value of k is a part of the input.

The 1-center problem, also know as the minimum enclosing circle problem involves with placing only 1 facility. In Eucledian plane this problem can be solved in $O(n)$ time, using the famous Megiddo prune and search technique. Here we consider a different version of the 1-center problem. The sites are located on the surface of a sphere, and the problem is to find a facility (on the surface of the sphere) with the same criteria and the distance metric set as $d(\hat{p}, \hat{q}) = \arccos(\hat{p}.\hat{q})$ where $\hat{p}$ and $\hat{q}$ are points on the surface of the sphere. The solution uses the Voronoi Diagram construction of points on spherical surfaces, which is an extension of the fortune's algorithm.[Sweeping the Sphere, Denis and Memede] The time complexity of our algorithm is $O(n \log n)$ and the space complexity is $O(n)$.

We also look into the problem of 2-disk cover, which is, given a set of sites on the spherical surface, to find two circles whose union can cover all the sites whose radius are not more than a given value r. This may then be extended to solve the 2-center problem for spherical surfaces.

Contents

1 Introduction **6**

2 Voronoi Diagram of a Sphere **7**
2.1 Introduction 7
2.2 Preliminaries and Definitions 7
2.3 Algorithm Overview 9
2.4 Voronoi Diagram Algorithm 11
2.5 Running Time and Storage 13

3 Spherical Minimum Enclosing Circle **14**
3.1 Introduction 14
3.2 Definition and Characteristics 14
3.3 Observations 16
3.4 Algorithm for Computing Spherical MEC 17
3.5 Complexity 18
3.6 Restricted Spherical MEC 18
3.6.1 Finding the Hemisphere 18
3.6.2 Restricted Spherical MEC with Center on a Query Line 19
3.6.3 Algorithm 20

4 The Spherical Two Disc Cover problem **22**
4.1 Introduction 22
4.2 Dynamic Maintenance of the Intersection of Congruent Disks . 22
4.3 Solving the 2DC Problem: the Case Where the Centers Are Well-Separated 24
4.4 Solving the 2DC Problem: the Case Where the Centers Are Close to Each Other 26

5 Conclusion and Future Work **28**

List of Figures

1 Voronoi diagram of some random sites 9
2 Parabola created by a point $\hat{p}$ 10
3 $K(P) = \cap_{p \in P} B_r(p)$ 23
4 The case $r < |c_1 c_2| \leq 3r$ 24
5 The case where $|c_1 c_2| < r$ 26

1 Introduction

One of the important problems of facility location is the k-center problem. The problem is defined as, given n sites, we need to find k facilities such that the maximum distance of any site to its nearest facility is the minimum over all placements of the k facilities. More formally, let C be the set of sites, then the problem is to find a set of facilities F such that the value $p = max_{f \in F}(min_{c \in C}(d(f, c)))$ is minimized; $d(f, c), f \in F$ and $c \in C$ is the distance metric. This problem is NP-hard if the value of k is a part of the input. A result [9] shows that their is an $n^{O(\sqrt{k})}$ algorithm for any k-center problem.

The 1-center problem and the 2-center problem are special cases of the k-center problem where the value of $k = 1$ and 2 respectively. In euclidean plane the best algorithm to solve the 1-center problem had been given by [8] and can be computed in $O(n)$ time. The planar 2-center problem has been solved in $O(n^2 \log^3 n)$ time by [6]. More recent algorithm by [10] and [11] solve the planar version in $O(n \log^9 n)$ and expected $O(n \log^2 n)$ time.

Here we consider a different version of the k-center problem. The sites are located on the surface of a sphere, and the problem is to find k facility (on the surface of the sphere) with the same criteria and the distance metric set as $d(\hat{p}, \hat{q}) = \arccos(\hat{p}.\hat{q})$ where $\hat{p}$ and $\hat{q}$ are points on the surface of the sphere. We give an algorithm for the spherical version of the 1-center problem. The solution uses the voronoi diagram construction of points on spherical surfaces, which is an extension of the Fortune's algorithm on spherical surfaces[4] and [5]. We give a detailed description of the voronoi diagram construction in Section 2. In the subsequent section we explain the construction of the spherical minimum enclsoing circle. The time complexity of our algorithm is $O(n \log n)$ and the space complexity is $O(n)$.

In Section 4 we look into the problem of 2-disk cover, which is, given a set of sites on the spherical surface, to find two circles whose union can cover all the sites whose radius are not more than a given value r. This is an important sub problem for the computation of the spherical 2-center problem. Our algorithm follows from the work of [10] done on planar 2-center problem. We provide a simple data structure to keep track of whether a set of sites on the surface of a sphere can be covered by a circle of given radius r. We then use this to solve the 2-disc cover problem in $O(n^2 \log n)$ time.

2 Voronoi Diagram on a Spherical Surface

2.1 Introduction

The Voronoi tessellation associates a discrete set of given points, called sites, with regions of space, called cells, such that all points in the cell associated with a given site are closer to that site than to any other. In addition to generating grids, the Voronoi tessellation can be used to find facilities, such as hospitals or fuel depots, nearest to a given location, to identify the nearest neighbours of a site, as well as for many other applications in biology, physics and computer science. In $2D$, Voronoi cells are polygons in the plane, each containing the associated site. Edges of Voronoi cells are bisectors of the line joining neighbouring sites and consist of points equidistant from them. Vertices of Voronoi cells are intersections of edges and are equidistant from three or more sites. The Voronoi diagram shows the sites and the boundaries of the cells.

In $2D$, the simple algorithm which finds the Voronoi tessellation by computing the common intersections of bisectors formed by the site i and all other sites $j \neq i$; requires $O(N^2 \log N)$ time. Algorithms of complexity $O(N \log N)$ exist; these include incremental construction with randomization [1], divide and conquer algorithm [2], and Fortune's sweep line algorithm [3]. The Fortune algorithm uses a sweep line, and generates parabolas defined by the sites and the sweep line. The intersections of neighbouring parabolas are used to determine the cell edges.

The sweep line algorithm for constructing the Voronoi diagram can be extended to manifolds other than the $2D$ plane; in our case, to the surface of the sphere. The structure of this algorithm presented by [4] and [5] is similar to that of Fortune for the tessellation of the $2D$ plane, but with a different definition of distance, and with interesting new features due to the topology of the sphere. One simplification also arises, since tessellation of the sphere is necessarily on a closed domain. We explain their algorithm in the following sections.

2.2 Preliminaries and Definitions

Since any sphere can be scaled and shifted only the tessellation of the unit sphere centred at the origin is considered. The unit sphere is denoted as

$$S^2 = \{\hat{p} \in R^3 : ||\hat{p}||_2 = 1\}$$

where $||.||_2$ is the L^2 norm. The (geodesic) distance between any two points $\hat{p}$ and $\hat{q}$ on sphere is denoted as $d(\hat{p}, \hat{q})$, and

$$d(\hat{p}, \hat{q}) = \arccos(\hat{p}.\hat{q})$$

This is the arclength of the shorter segment of the great circle through $\hat{p}$ and $\hat{q}$, with $\hat{p}$ and $\hat{q}$ as endpoints. A region $H \subset S^2$ is convex if a geodesic joining any two points in H is contained in H.

The site is a set of N distinct points $P := \{\hat{p}\}_{t=1}^{N}$ on the sphere. The Voronoi diagram of P is denoted by $Vor(P)$. And $V(\hat{p}_i)$, the Voronoi cell for the site $\hat{p}_i$, the set of points on the sphere that are closer to $\hat{p}_i$ than to any other site:

$$V(\hat{p}_i) = \{\hat{r} \in S^2 : d(\hat{r}, \hat{p}_i) < d(\hat{r}, \hat{p}_j), \forall j \neq i\}$$

The bisector B_{ij} is the set of points equidistant to sites $\hat{p}_i$ and $\hat{p}_j$,

$$B_{ij} = \{\hat{r} \in S^2 : d(\hat{r}, \hat{p}_i) = d(\hat{r}, \hat{p}_j)\}$$

The bisector is a great circle with normal along $\hat{p}_j - \hat{p}_i$. Each Voronoi cell $V(\hat{p}_i)$ is a closed convex region on sphere, since it is formed by $N-1$ intersection of bisectors. A Voronoi edge E_{ij} between sites $\hat{p}_i$ and $\hat{p}_j$ is the intersection of B_{ij} with the boundary of the Voronoi cell,

$$E_{ij} = \delta V(\hat{p}_i) \cap B_{ij} = \delta V(\hat{p}_j) \cap B_{ij}$$

All edges in the Voronoi diagram on the sphere are circular arcs; segments of great circles. A Voronoi vertex $\hat{v}_{ij\ldots k}$ shared by sites $\hat{p}_i, \hat{p}_j, \ldots, \hat{p}_k$ is the point on the boundary of Voronoi cell and equidistant to those sites,

$$\hat{v}_{ij\ldots k} = \{\hat{r} \in \delta V(\hat{p}_i) : d(\hat{r}, \hat{p}_i) = d(\hat{r}, \hat{p}_j) = \ldots = d(\hat{r}, \hat{p}_k)\}$$

A circumcircle on a sphere which passes through the three sites $\hat{p}_i, \hat{p}_j, \ldots, \hat{p}_k$ is the intersection of the plane determined by these sites and the sphere. The circumcenter of the circle is defined to be on the sphere, and is equidistant to these three sites, thus it is the intersection of any two of the three possible bisectors. The intersection $\hat{v}_{ij}^{kj}$ between two bisectors B_{ij} and B_{kj}, is on both great circles, thus orthogonal to both normals, and is given by the cross product of the normals of the respective great circles,

$$\hat{v}_{ij}^{kj} = \pm \frac{(\hat{p}_i - \hat{p}_j) x (\hat{p}_k - \hat{p}_j)}{|\hat{p}_i - \hat{p}_j) x (\hat{p}_k - \hat{p}_j|}$$

There are two such intersections which lie on the two ends of the diameter of the sphere. The radius of the circumcircle is given by

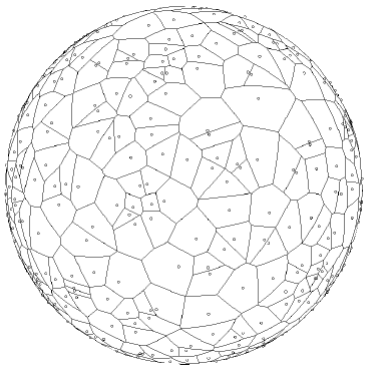

Figure 1: Voronoi diagram of some random sites

$$r = d(\hat{v}_{ij}^{kj}, \hat{p}_i) = \arccos(\hat{v}_{ij}^{kj}.\hat{p}_i)$$

Note that the radius depends on the choice of the circumcenter $\hat{v}_{ij}^{kj}$, and may be greater than $\frac{\pi}{2}$. For the Voronoi diagram $Vor(P)$, a circumcenter $\hat{v}_{ij}^{kj}$ is a vertex of $Vor(P)$ if and only if the circumcircle is empty, that is, if it does not contain any sites in the interior. The interior of the circumcircle is defined to be the spherical cap of the sphere cut by the plane determined by the three sites, containing the circumcenter $\hat{v}_{ij}^{kj}$.

2.3 Algorithm Overview

The Voronoi edge has the property that the points it contains are equidistant from two sites. If a parabola β_1 is defined as the locus of points that are equidistant from a point p_1 and a circumcircle, which we denote as the line L; and parabola β_2 as the locus of points equidistant from a second point p_2 and the same line L, then the intersection between β_1 and β_2 is equidistant from both p_1 and p_2, assuming both sites are on the same side of the line L. Making use of this fact is the central point of the Fortune algorithm.

The sweep line L on the sphere is defined as the intersection of a plane - the sweep-plane - with the sphere. The normal of the sweep-plane is $\hat{n}$, the point defined by $\hat{n}$ is called the north pole. The line L is characterized by the normal $\hat{n}$; and a parameter ξ, so that

$$L(\hat{n}, \xi) = \{\hat{r} : \hat{r}.\hat{n} = \cos\xi\}$$

In the algorithm, the line sweeps the sphere as follows. First from top ($\xi = 0$) to bottom ($\xi = \pi$), and then from bottom ($\xi = \pi$) towards top

($\xi = 2\pi$). It is convenient to imagine that, for $\xi > \pi$, the line is on the inside surface of the sphere. The line starts to sweep from the north pole towards the south pole, with increasing ξ.

Each site p_i above (north of) the sweep line, together with the sweep line, determines a parabola β_i. The boundary of the union of all such parabolas is the front, called the beach line, situated above the sweep line for $\eta \leq \pi$. It is easy to show that all points which lie above the beach line are closer to some site $p_i \in P$ above the beach line than to the line L, thus their closest site will not be below the sweep line. Hence points which lie above the beach line have already been incorporated into the diagram, leaving the region below the beach line to be considered. Two neighbouring arcs on the beach line intersect, and such intersections are called breakpoints. Each breakpoint is equidistant from two sites, which define the arcs of the parabolas, and hence must lie on some Voronoi edge. The sweep plane algorithm maintains the beach line as the line L sweeps, and traces the paths of the breakpoints as they move along the edges of the Voronoi diagram. The beach line is a line of latitude, any great circle passing through the north pole $\hat{n}$ intersects with the beach line exactly twice. The azimuthal angles of the intersections differ by π.

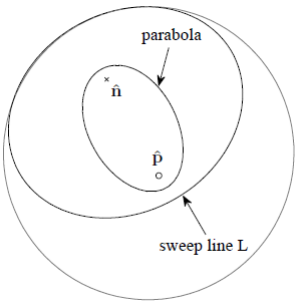

Figure 2: Parabola created by a point $\hat{p}$

A key difference between this scheme and the planar case is that the beach line on the sphere is a closed curve, while in the planar case it is a curve with open ends. In the planar case, the domain is usually infinite, and the line can sweep indefinitely to finish the diagram. However, the surface of a sphere is a closed domain. When sweep line reaches the south pole, the diagram is not finished. Even though all sites are located above the beach line, the region below the beach line has not yet been processed. Therefore the sweep line continues to sweep the line upward, figuratively on the inside of the sphere, for values $\xi > \pi$, until the tessellation is complete.

2.4 Voronoi Diagram Algorithm

The sweep line algorithm consists of simulating the growth of the beach line as the sweep line moves first down, outside then up, inside, the sphere. The beach line formed by the parabolas changes its shape continuously during the line sweep. As with in the case of the plane-sweep algorithm, only discrete "events" are of interest when there is change in the topology of the Voronoi diagram and structure of the beach line, all intermediate steps are skipped.

Input: A set $P := \{\hat{p}_i\}_{i=1}^{N}$ of N distinct points on the sphere.
Output: The Voronoi diagram $Vor(P)$ given in a doubly-connected edge list D.

1. Sort the sites in P and put them in the event queue Q_{site}; Initialize the empty event queue Q_{circle}; initialize the empty skiplist, and a doubly connected edge list D.

2. While Q_{site} or Q_{cirlce} is not empty

3. if Q_{site} is not empty, and the site $\hat{p}$ is before the events in the Q_{circle}

4. then HandleSiteEvent($\hat{p}_i$), and remove the node from Q_{site}

5. else HandleCircleEvent(γ), where γ is a node in the skiplist that represents the arc that will disappear

6. Traverse the half-edges of the doubly-connected edge list to add cell records and pointers to and from them.

The Procedures to handle the events are defined below.

HandleSiteEvent($\hat{p}_i$)

1. If the skiplist is empty, insert $\hat{p}_i$ into it, and set the previous and next pointers to point to itself. If the skiplist only contains one node, then simply append $\hat{p}_i$ to it, and reset all pointers correspondingly. Otherwise, continue with steps 2-4.

2. Search the skiplist from the reference position for the arc α which will intersect with the great circle through $\hat{p}_i$ and north pole $\hat{n}$. Assume that the two end points of α are (θ_1, ϕ_1) and (θ_2, ϕ_2) such that points on α with azimuthal angle which lies between ϕ_1 and ϕ_2. If

$$\phi_1 \leq \phi_i \leq \phi_2,\ for\ \phi_1 < \phi_2$$

or

$$\phi_i \leq min(\phi 1, \phi 2)\ or\ \phi_i \geq max(\phi_1, \phi_2),\ for\ \phi_1 > \phi_2$$

then the arc is found. If the arc α has a pointer to a circle event in the Q_{circle}, then the circle event is a false alarm, and is removed from Q_{circle}.

3. Duplicate arc α and insert the new arc for $\hat{p}_i$ between them in the skiplist. Set the previous and next pointers appropriately. Suppose that prior to the insertion, the beach line sequence was

$$\ldots \hat{p}_1, \hat{p}_2, \hat{p}_3, \hat{p}_4, \ldots$$

The insertion of $\hat{p}_i$ splits the arc associating $\hat{p}_j$ into two, denoted $\hat{p}_i^{'}$ and $\hat{p}_i^{''}$, involving the same site, $\hat{p}_j$, the new sequence will be

$$\ldots, \hat{p}_1, \hat{p}_2, \hat{p}_j^{'}, \hat{p}_i, \hat{p}_j^{''}, \hat{p}_3, \hat{p}_4, \ldots$$

4. Check the triple of consecutive arcs $\hat{p}_2, \hat{p}_j^{'}, \hat{p}_i$ and $\hat{p}_i, \hat{p}_j^{''}, \hat{p}_3$ for valid circle events. If found, insert the circle event(s) into Q_{circle} and add pointers between the node(s) in the skiplist and in Q_{circle}.

HandleCircleEvent(γ)

1. Let $\hat{p}_i, \hat{p}_j, \hat{p}_{ik}$ be the three sites that generate this circle event. Delete the node for $\hat{p}_j$ representing the disappearing arc from the skiplist and remove the circle event from the Q_{circle}; Suppose that prior to the deletion, beach line sequence was

$$\ldots \hat{p}_1, \hat{p}_i, \hat{p}_j, \hat{p}_k, \hat{p}_2, \ldots$$

after the event, the sequence this becomes

$$\ldots \hat{p}_1, \hat{p}_i, \hat{p}_k, \hat{p}_2, \ldots$$

2. Add the circumcenter of the circumcircle causing the circle event as a vertex record to the doubly-connected edge list D. Create three half-edge records starting with the new vertex in three cells corresponding to $\hat{p}_i, \hat{p}_j, \hat{p}_k$.

3. Remove all circle events associated with the triples $\hat{p}_1, \hat{p}_i, \hat{p}_j$ and $\hat{p}_j, \hat{p}_k, \hat{p}_2$ from Q_{cirlce}, this can be done by using pointers from the predecessor and the successor of $\hat{p}_j$ in the skiplist. Check two new triples of consecutive arcs involving both former left and right neighbor arcs of α, that is, $\hat{p}_1, \hat{p}_i, \hat{p}_j$ and $\hat{p}_j, \hat{p}_k, \hat{p}_2$ for valid circle events. If found, insert the circle event(s) into Q_{cirlce}.

2.5 Running Time and Storage

The total running time is $O(N \log N)$, and the required total storage space is $O(N)$.

Handling each site event requires searching for the arc where the new parabolic arc can be inserted; this requires time $O(\log N)$. Inserting the arc into the beach line requires constant time $O(1)$. Handling each circle event requires constant time $O(1)$. Checking for the circle events requires constant time $O(1)$ as well. There are N site events and 2N-4 circle events, thus the total running time is $O(N \log N) + O(N)$.

All data structures require the storage space $O(N)$. The beach line has at most $2N-2$ arcs. Since all circle events in the queue are associated with sites in the beach line, the maximum length of the circle event queue is $2N - 2$. The site event queue has N nodes. The expected number of edges in the Voronoi diagram is $3N - 6$.

3 Spherical Minimum Enclosing Circle

3.1 Introduction

The smallest-circle problem or minimum covering circle problem is a mathematical problem of computing the smallest circle that contains all of a given set of points in the Euclidean plane. The smallest-circle problem was initially proposed by the English mathematician James Joseph Sylvester in 1857 [7]. The smallest-circle problem in the plane is an example of a facility location problem (the 1-center problem) in which the location of a new facility must be chosen to provide service to a number of customers, minimizing the farthest distance that any customer must travel to reach the new facility.

The minimal enclosing circle is used in planning the location of a shared facility. For example, a shared facility is a hospital servicing a community. If we consider each home in a community as points in the plane, finding minimal enclosing circle gives a good place to put the hospital i.e, the center of the minimal circle. Placing the hospital at the center of minimal circle minimizes the distance between the hospital and the farthest home (point) in the community. This problem is also useful to examine the point that lie on the boundary of the minimal enclosing circle.

The simplest algorithm considers every circle defined by two or three of the n points, and finds the smallest of these circles that contains every point. There exits $O(n^3)$ such circles, and each takes $O(n)$ time to check, for a total running time of $O(n^4)$. Elzinga and Hearn gave an $O(n^2)$ algorithm in 1972, and Shamos and Hoey (1975), Preparata (1977), and Shamos (1978) discovered the first $O(n \log n)$ algorithms. Finally, in 1983 Nimrod Megiddo showed that the minimal enclosing circle problem can be solve in $O(n)$ time using the prune-and-search techniques for linear programming.

We considered here the spherical version of the minimal enclosing circle problem. The set of sites P is located on the surface of a sphere S and we need to find a 1-center i.e. the point with the least maximum distance from all points in the set P.

3.2 Definition and Characteristics

Since any sphere can be scaled and shifted, we only consider the spherical minimum enclosing circle of points on a unit sphere. The unit sphere is denoted as

$$S^2 = \{\hat{p} \in R^3 : ||\hat{p}||_2 = 1\}$$

where $||.||_2$ is the L^2 norm. The (geodesic) distance between any two points $\hat{p}$ and $\hat{q}$ on a sphere is denoted as $d(\hat{p};\hat{q})$, and

$$d(\hat{p};\hat{q}) = \arccos(\hat{p}.\hat{q})$$

This is the arclength of the shorter segment of the great circle through $\hat{p}$ and $\hat{q}$, with $\hat{p}$ and $\hat{q}$ as endpoints.

Two points p_1 and p_2 on the surface of a sphere S are said to be antipodal to each other if the line joining them passes through the center of the sphere S.

A circle C on the spherical surface is defined as the intersection of a the boundary of the sphere S with a plane L. Its normal $\hat{n}$ is defined as the normal to the plane L which passes through the center of the sphere S. The center of C is the intersection of its normal $\hat{n}$ with the boundary of the sphere S. Note that a circle has two centres which are an antipodal pair.

The spherical minimum enclosing circle M of a set of points P on the surface of the sphere, is defined as the circle with the minimum radius which covers the set P. By covering we mean that all the points in P lie on one side of the plane L whose intersection with the sphere S forms the circle M. The center of the spherical minimum enclosing circle M is the center of the circle M on that side of L on which all of P lies. The radius is defined as the geodesic distance between the boundary of the spherical minimum enclosing circle M and its center.

Alternatively, the spherical minimum enclosing circle problem can also be formulated as the 1-centre problem. Given a set of points P on the surface of a sphere S, to find a point c (also on the surface of the sphere), such that the maximum distance between c and any other point $p \in P$ is the minimum over all point $c \in \delta S$. δS is defined as the boundary of the sphere S.

Some characteristics of the spherical minimum enclosing circle are as follows:

- The spherical MEC must have at least two points on its boundary.
- If the spherical MEC passes through two points then the great circle joining these two points passes through the center of the spherical MEC.
- If the spherical MEC passes through three points or more than the three points together will form an acute angled triangle.
- Unlike the MEC in aplane the spherical MEC is not unique.

3.3 Observations

The observations that can be made about the MEC and the vornoi diagram are as follows

Observation 1: *For a point set $P = p_1, p_2, p_3, \ldots$ on the surface of a sphere the nearest neighbour voronoi diagram V is also the farthest pair voronoi diagram of the set of points $P^{'} = p_1^{'}, p_2^{'}, p_3^{'}, \ldots$ where $p_i^{'}$ is the antipodal point with respect to p_i.*

Proof. For points $x, y \in S$ we have $d(x, y) = \pi - d(x, y^{'})$. It follows that for a site $u \subset S$, we have $d(x, u) = \pi - d(x, u^{'})$. Therefore, x belongs to the region of $Vor(P)$ where u belongs if and only if x belongs to the reqion of farthest-$Vor(P)$ where $u^{'}$ belongs.

Observation 2: *The dual of the spherical minimal enclosing circle is the spherical maximum empty circle i.e. both the circles have the same boundary and their center forms an antipodal pair.*

Proof: The maximum empty circle is defined as the circle with the maximum radius which does have any site inside it. The proof is trivial in the sense that if the radius of the minimal enclosing circle is r then the radius of the maximal empty circle is $\pi - r$.

Observation 3: *The complexity of the nearest neighbour voronoi diagram with respect to the number of points is linear.*

Proof. Considering the general case when the sites do not all lie on a great circle. The Euler characteristic for a convex polyhedron with v vertices, e edges and f faces is

$$v - e + f = 2$$

This can be applied here, only the number of faces needs to be replaced by number of cells. If N_n is the number of n-sided cells, then, for a nondegenerate Voronoi diagram, where each edge is shared by two, and each vertex by three sites, we have

$$\sum_n (N_n \frac{n}{3} - N_n \frac{n}{2} + N_n) = 2$$

$$\sum_n (6 - n) N_n = 12$$

The average number of edges of the Voronoi cell is

$$\frac{\sum_n nN_n}{N} = 6 - \frac{12}{N}$$

Thus the average number of edges of Voronoi cells is less than 6, and approaches 6 as the number of sites becomes large. The total number of vertices is $2N - 4$, and the total number of edges is $3N - 6$.

3.4 Algorithm for Computing Spherical MEC

Based on the above observations we now proceed to give an algorithm for computing the spherical minimum enclosing circle for a set of points P on the surface of the sphere. The algorithm takes help of the computation of the nearest neighbour voronoi diagram explained in the previous section. We first compute the nearest point voronoi diagram of P. Then for each of the edges and vertices of $Vor(P)$ we compute the maximum empty circle with center on it. The largest of all these maximum empty circles is the global maximum empty circle and its complement is the minimum enclosing circle.

Input: S, the sphere; $P = \{p_1, p_2, p_3, \ldots\}$, the set of points on the surface of the sphere.
Output: The spherical MEC of P.

1. Compute the voronoi diagram of P by the algorithm for voronoi diagrams in a sphere. For each vertex of the voronoi diagram maintain any one of at least three points of P which forms that vertex. Also for every edge of the voronoi diagram maintain the pair of points of P which are associated with it.

2. For each vertex of the voronoi diagram compute the radius of the maximum empty circle that can be drawn with center on the vertex. This is nothing but the distance between the vertex and the point associated with it. For an edge of the voronoi diagram, compute the intersection of that edge with the line joining its associated pair of site points. If there is no intersection than this is not the candidate for the global maximum empty circle and so can be skipped. Else the radius of the maximum empty circle with center as the intersection point is computed.

3. Out of all the maximum empty circle considered, find the one with the maximum radius. We call it the global maximum empty circle.

4. The complement of the global maximum empty circle is the spherical minimum enclosing circle for the points set P. If the center and radius of the global maximum empty circle are c and r respectively then, the spherical minimum enclosing circle has radius $R = \pi - r$ and center $C = c^{'}$, where $c^{'}$ is the antipodal point of c.

3.5 Complexity

The total running time is $O(N \log N)$, and the required total storage space is $O(N)$.

As already has been discussed computing the voronoi diagram in a sphere takes $O(N \log N)$ time. At creation of any vertex or edge of the voronoi diagram its associated site points can be inserted in $O(1)$ time. Computation of each maximum empty circle takes $O(1)$ time. And since the complexity of the voronoi diagram is $O(N)$ the total time to compute all valid maximum empty circles and finding the global maximum among them takes $O(N)$ time.

All data structures need only $O(N)$ space to store the voronoi diagram and the associated site points for each of vertices and edges. This is because the total number of vertices and edges is $O(N)$. The maximum empty circle can be defined by just the radius and center and needs $O(1)$ storage per circle.

3.6 Restricted Spherical MEC

In this section we look at a restricted version of the 1-center problem. Let C be a set of sites on the unit sphere S such that their exists a plane passing through the center of the sphere and all points in C lie on one side of the plane. The problem is to find a circle which encloses all the points, and whose radius is the minimum among all possible such circles. We call it the restricted spherical minimal enclosing circle problem.

3.6.1 Finding the Hemisphere

The first problem is to find a hemisphere such that all points in C are on that hemisphere. We give an $O(n)$ algorithm for it as follows:

Input: S, the sphere; $P = \{p_1, p_2, p_3, \ldots\}$, the set of points on the surface of the sphere.
Output: A hemisphere H such that H encloses p, $\forall p \in P$.

1. Let p_i be any point in P.

2. For each point $p_j \in P$ we compute $d(p_i, p_j)$.

3. Let p_f be the point which is farthest from p_i, i.e. $d(p_f, p_i) \geq d(p_j, p_i), \forall p_j \in P$. Note that p_f lies on the convex hull of P and $\exists$ a hemisphere h which covers all the points in P and which passes through p_f.

4. We consider the great circle passing through p_i and p_f as the 0° meridian and p_f as the north pole.

5. $\forall p_j \in P$, find the points p_{max} and p_{min} with the maximum and minimum azimuthal angel. We say that both these points lie on the convex hull of the point set P.

6. Therefore the great circle passing through p_f and p_{max}(or p_{min}) is the required hemisphere H.

3.6.2 Restricted Spherical MEC with Center on a Query Line

Here we describe the sub problem required by Megiddo's prune and search technique to find the 1-center for a set of a set of points. We change some of the definitions to fit the algorithm for the restricted spherical MEC. For any pair of orthogonal great circles, consider their intersection with the computed hemisphere H. There will be 2 semi great circle segments; we call one of them as the NS axis and the other as the EW axis, with their point of intersection with H being N, S and E, W respectively. We then try to find the spherical 1-center of the set of sites C, with the restriction that the 1-center lies on the EW great circle segment.

Input: H, the hemisphere; $P = \{p_1, p_2, p_3, \ldots\}$, the set of points on the surface of the sphere; the 2 axes NS and EW.
Output: The 1-center for P restricted on the semi great circle segment EW.

1. If P has no more than 2 points we compute the restricted 1-center by brute force.

2. Form disjoint pair of points of P as $(p_1, p_2), (p_3, p_4), \ldots$. If there are odd number of points in P then set the last pair as (p_n, p_1).

3. For each pair of points (p_i, p_{i+1}) find the point x_i on EW such that $d(x_i, p_i) = d(x_i, p_{i+1})$. Note that their will always be 1 such points for any pair of points in the EW semi great circle segment.

4. Find the median of the $\frac{n}{2}$ x_i's. Let it be called x_m.

5. Computer the set I of points in P which are farthest from x_m.

6. Let m be a meridian passing through N, S and x_m. If points in I lie on both sides of m, we have found a solution.

7. Else if all points in I are to the left of m, then for each x_i to the right of m discard the point farthest from x_m in the pair (p_i, p_{i+1}). The other case is symmetric.

8. Repeat steps 1 - 7.

At each step we prune $\frac{1}{4}^{th}$ of the points. Therefore the running time of this algorithm is given by the recursion

$$\begin{aligned} T(n) &= T(\frac{3n}{4}) + O(n) \\ &= O(n) \end{aligned}$$

3.6.3 Algorithm

After finding out the hemisphere H we can compute the minimal enclosing circle by megiddo's prune and search technique.

Input: H, the hemisphere; $P = \{p_1, p_2, p_3, \ldots\}$, the set of points on the surface of the sphere.
Output: The 1-center for P.

1. If the number of points in P is ≤ 16, then we find the restricted 1-center problem using brute force. We compute all possible pair of sets which are a cover of P and check for which pair we get the circles with the minimum radius.

2. Form disjoint pair of points of P as $(p_1, p_2), (p_3, p_4), \ldots$. If there are odd number of points in P then set the last pair as (p_n, p_1).

3. For each pair of points (p_i, p_j) we compute the perpendicular bisector of $p_i\bar{p}_j$. This is a great circle which intersects EW segment at atleast 1 point. Let them be denoted by L_i's.

4. We compute the slopes of all the bisectors with the horizontal axis and find the median. Then we rotate the coordinate axis such that exactly half of the slopes are positive and the rest of them negative. Note that the points N, S, E, W also gets rotated accordingly.

5. Construct a set of disjoint pair L_i, L_{i+1} such that L_i has a positive slope and L_{i+1} has a negative slope with the horizontal axis. Calculate their point of intersection and let them be denoted by x_i's.

6. We find the median of the x_i's first in the horizontal direction and find a great circle v passing through N, S such that half the x_i's are to its left and half the points to its right. We then apply the query line restricted 1-center algorithm in the previous section on the segment EW to check if the optimal 1-center lies to the left or to the right of v or on v.

7. If the optimal 1-center lies to the left of v then for all x_is to the right of v we compute thier median in the vertical direction. We find a horizontal lattitute h passing thorough E, W such that half of these x_i's are above and the other half below. We then apply the query line restricted 1-center algorithm on the query line v to determine which direction of h the optimal point lies. This is similar for the case when optimal point lies to the right of v.

8. If we have found the optimal point we stop. Else if say the optimal point is to the left of v and below h. Then for each x_i to the right of v and above h, let L_i be the bisector with the negative slope associated with x_i. We prune away that point in the pair of L_i which is closer to the intersection of v and h. The case is symmetric for other directions of the 1-center with respect to v and h.

9. Continue from step 1.

At each step we do $O(n)$ computations. The number of x_i's is $\frac{n}{4}$ and for $\frac{1}{4}^{th}$ of these x_i's we prune away one point of P associated with it. Therefore the time complexity of this algorithm can be given by the recursion

$$\begin{aligned} T(n) &= T(\frac{15n}{16}) + O(n) \\ &= O(n) \end{aligned}$$

4 The Spherical Two Disc Cover problem

4.1 Introduction

Let S be a set of n points on the surface of the sphere. The 2-center problem for S is to cover S by (the union of) two congruent closed disks whose radius is as small as possible. This is a special case of the general p-center problem, where we wish to cover S by p congruent disks whose radius is as small as possible. When p is part of the input, the problem is known to be NP-complete [8]. A result [9] shows that their is an $n^{O(\sqrt{P})}$ algorithm for any p-center problem.

A major component of the algorithm for computing the 2-center problem is a procedure for solving the fixed-size problem: Given a radius r, we want to determine whether S can be covered by two closed disks of radius r. This problem is refereed to as the $2DC$ (2-disk cover) problem. The strategy is to assume that such a pair of disks exist, call them D_1, D_2, and to conduct a search for their centres. Let c_i denote the center of D_i , and let C_i denote the circle bounding D_i , for $i = 1, 2$. We may assume, with no loss of generality, that $|c_1c_2|$ is as small as possible.

We describe an algorithm for the $2DC$ problem in the subsequent section. The algorithm uses a routine which we explain in the next section. Section 4.3 and 4.4 then describe the algorithm for two different cases. Both of these cases are related to the work done by [10] for euclidean plane.

4.2 Dynamic Maintenance of the Intersection of Congruent Disks

We first describe a procedure, which the algorithm will use repeatedly, for solving the $2DC$ problem. We want to maintain dynamically a set P of points in the plane, under insertions and deletions. After each update, we wish to determine whether the intersection $K(P) = \cap_{p \in P} B_r(p)$ is nonempty, where $B_r(p)$ is the closed disk of radius r centered at p. This condition is equivalent to the condition that P can be covered by a disk of radius r.

At every insertion and deletion of $B_r(p)$ we keep track of the boundary region of $K(P)$. $K(P)$ is maintained as an alternating sequence of points and circles. For e.g. for the Figure 3 we maintain the region $K(P)$ as

$$q_1, B_r(p_1), q_2, B_r(p_2), q_3, B_r(p_3).$$

At every insertion of a new circle $B_r(p)$ we find in $O(n)$ time the intersection of $B_r(p)$ with $K(P)$. We then update the region of $K(P)$ as will be

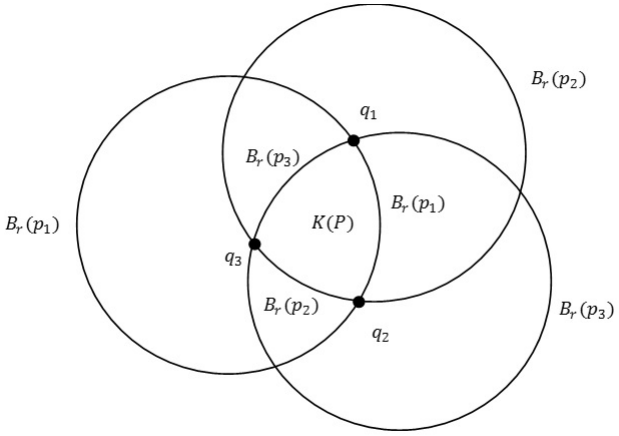

Figure 3: $K(P) = \cap_{p \in P} B_r(p)$

described. Suppose the present $K(P)$ be given by the sequence

$$q_1, B_r(p_1), q_2, B_r(p_2), q_3, B_r(p_3), q_4, B_r(p_4), q_5, B_r(p_5), q_6, B_r(p_6)$$

and let the newly inserted circle $B_r(p_i)$ intersect $K(P)$ at the curves $B_r(p_2)$ and $B_r(p_5)$ at points p_j and p_k respectively. Then the $K(P)_{new} = K(P)_{old} \cap B_r(p_i)$ is given by the sequence

$$q_1, B_r(p_1), q_2, B_r(p_2), q_j, B_r(p_i), q_k, B_r(p_5), q_6, B_r(p_6)$$

and we store the sequence $q_3, B_r(p_3), q_4, B_r(p_4), q_5$ at some place index by $B_r(p_i)$. If the inserted circle $B_r(p_i)$ doesn't intersect $K(P)$ at all, then we insert the circle into a queue T.

In the case of deleting a circle $B_r(p_i)$, this data structure only deletes circles in the order in which they were inserted. We check $K(P)$ and find the position of $B_r(p_i)$ in the sequence. We delete $B_r(p_i)$ and its immediate predecessor and successor and insert in its place the sequence of points stored at the location refereed to by $B_r(p_i)$. If T is non-empty then we insert the circles into the now modified $K(P)$; we stop inserting at the very first circle which doesn't intersect with $K(P)$.

Each circle is inserted into $K(p)$ at most once, so if there are n insertions and m deletions, the total amortized complexity is $O(n^2)$ as $m < n$.

4.3 Solving the 2DC Problem: the Case Where the Centers Are Well-Separated

Suppose first that $|c_1c_2| > r$. Let $\delta > 0$ be some sufficiently small constant angle. Now let us imagine 2 orthogonal axes one in the north-south direction and the other in the east-west direction. We rotate the sphere by $j\delta$, for $j = 0, 1, \ldots, \lfloor \frac{2\pi}{\delta} \rfloor$ on both the axes. In one of the orientations, the plane passing through the great circle passing through c_1 and c_2 will be almost horizontal.

Assume further that $|c_1c_2| > 3r$, say. Then a vertical meridian separating D_1 and D_2 exists. To detect whether this case arises, sort the points of S by their azimuthal coordinates, and scan them from left to right. Let S_L denote the set of points on the left hemisphere of the meridian which passes through the currently scanned point q, including q, and let S_R denote the complementary set. We maintain the sets S_L and S_R dynamically, repeatedly moving each scanned point from S_R to S_L, and checking, after each update, whether $K(S_L)$ and $K(S_R)$ are nonempty. If both are nonempty, we have found two disks of radius r whose union covers S. If the currently assumed configuration does exist and we are at the correct orientation, then, in exactly one of these steps, both intersections must be nonempty, so the above procedure will detect the existence of a 2DC of this kind. Using the technique described in Section 4.2, the cost of handling this case is $O(n^2)$.

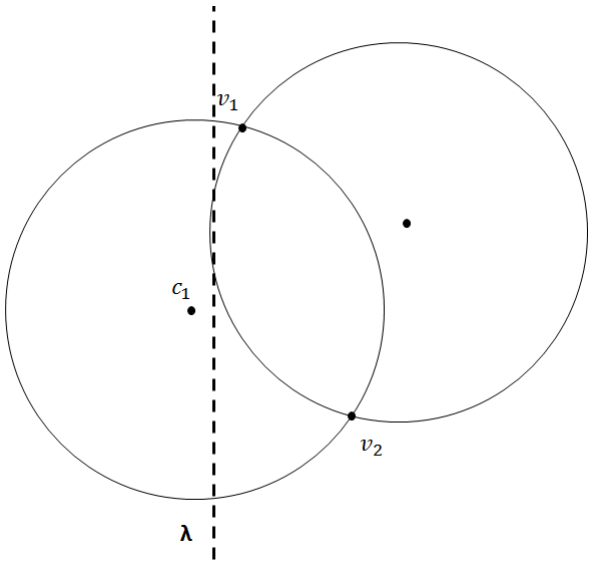

Figure 4: The case $r < |c_1c_2| \leq 3r$

Next assume that $r < |c_1c_2| \leq 3r$. Let v_1 and v_2 denote the points of intersection of C_1 and C_2, with v_1 lying to the left of v_2. See Figure 4. If D_1 and D_2 are disjoint, we define v_1 to be the leftmost point of D_2 and define v_2 to be the rightmost point of D_1; if v_1 lies to the right of v_2, we can proceed as in the previous case, because D_1 and D_2 are then separated by a vertical line; so we still assume that v_1 lies to the left of v_2. Since we assume that the orientation of c_1c_2 is at most some δ in absolute value and that $\theta(c_2) - \theta(c_1) > 0.99r$, it is easily seen that $\theta(v_1) - \theta(c_1) > 0.4r$. Note that the left semicircle of C_1 must pass through at least one point q of S (or else we could have brought D_1 and D_2 closer together, by moving D_1 to the right). Let λ be any vertical meridian separating c_1 from v_1, and let S_L denote the subset of points of S lying to the left hemisphere of λ. Then S_L contains q and is fully contained in D_1. Note that the difference between the largest and smallest azimuthal coordinates of points of S is at most $5r$, so we can draw a constant number of vertical meridians λ, say with horizontal separation $0.3r$ between adjacent meridians, so that at least one of them will separate c_1 and v_1. Assume that λ is the correct meridian. We then have the set S_L available, and we compute the region $K(S_L) = \cap_{p \in S_L} B_r(p)$, in $O(n^2)$ time. The above arguments imply that c_1 must lie on the (right) boundary of $K(S_L)$. For each $p \in S_R = S \setminus S_L$, we intersect $\delta B_r(p)$ with $\delta K(S_L)$. As is well known, each such intersection consists of at most two points. We obtain all these points, and sort them, including the vertices of $K(S_L)$ along $\delta K(S_L$, into a list Γ. This can easily be done in $O(n^2)$ time.

We now iterate over each point v in Γ, place the center c_1 of D_1 at v, or on the subarc of $\delta K(S_L)$ between v and the next point in Γ, and update the set $S^{'}(c_1)$ of points of S not covered by D_1. We note that when c_1 moves from a point in Γ to an adjacent subarc, or from a subarc to an adjacent point, either a single point is added to $S^{'}(c_1)$, or a single point is removed from that set, or the set remains unchanged. At each point c_1 that we visit, we test whether $S^{'}(c_1)$ can be covered by a disk of radius r , and stop as soon as this happens, for we have obtained an affirmative solution to the 2DC problem (with radius r). Otherwise, we continue until Γ is exhausted, and conclude that λ cannot be the desired line. If this procedure fails for all of the $O(1)$ lines λ and for all the $O(\frac{1}{\delta^2})$ orientations, we conclude that there is no solution to the 2DC problem (with radius r) with the currently assumed configuration. Here we can not use the technique of Section 4.2 as the deletions are not in the order necessary. Therefore using the solution of the spherical 1-center problem, the cost of handling this case is $O(n^2 \log n)$.

4.4 Solving the 2DC Problem: the Case Where the Centers Are Close to Each Other

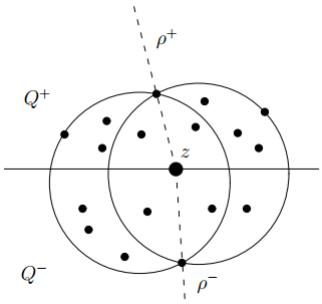

Figure 5: The case where $|c_1c_2| < r$

Finally, assume that $|c_1c_2| < r$. In this case the area of $D_1 \cap D_2$ is at least $O(r^2)$, whereas the entire S can be covered by a circle R of radius $O(r)$, which we can easily compute in $O(n \log n)$ time. It follows that we can construct $O(1)$ points within R, so that at least one of them will lie in $D_1 \cap D_2$ (and fairly close to both c_1 and c_2). Let z be such a point. We sort the points of S in angular order about z, and partition the sorted list into two sublists, Q^+, Q^-, by the horizontal great circle passing through z. Assume that Q^+ is sorted in clockwise direction about z and that Q^- is sorted in counterclockwise direction. See Figure 5

We now apply a technique that resembles standard searching in monotone matrices. Let M be the matrix whose rows correspond to points in Q^+ (in clockwise angular order), and whose columns correspond to points in Q (in counterclockwise order). For $a \in Q^+, b \in Q$, we define M_{ab} as follows. Let ρ^+ be a ray emanating from z and passing between a and the next point of Q^+, and let ρ^- be a ray emanating from z and passing between b and the next point of Q. Let S_L^+ be the prefix of Q^+ consisting of points that lie counterclockwise to ρ^+, and let S_L^- be the prefix of Q^- consisting of points that lie clockwise to ρ^-. Let $S_L = S_L^+ \cup S_L^-$ and let $S_R = S \setminus S_L$. Then

$$M_{ab} = \begin{cases} \text{'YY' if both } S_L \text{ and } S_R \text{ can be covered by disks of radius } r \\ \text{YN' if } S_L \text{ can be covered by a disk of radius } r \text{ but } S_R \text{ cannot} \\ \text{'NY' if } S_R \text{ can be covered by a disk of radius } r \text{ but } S_L \text{ cannot} \\ \text{'NN' if neither } S_L \text{ nor } S_R \text{ can be covered by a disk of radius } r. \end{cases}$$

(Note that the number of rows plus the number of columns of M is n.) Our goal is thus to determine whether M has an entry YY. We denote by $M^{(L)}$ (resp. $M^{(R)}$) the matrix containing the left (resp. right) characters of the entries of M. The matrices $M^{(L)}, M^{(R)}$ have the following monotonicity properties, whose proof is obvious:

(a) If $M^{(L)}_{ab} = \text{N}$ then $M^{(L)}_{a'b'} = \text{N}$ for any $a' \geq a, b' \geq b$.

(b) If $M^{(R)}_{ab} = \text{N}$ then $M^{(R)}_{a'b'} = \text{N}$ for any $a' \leq a, b' \leq b$.

We first compute all entries in the middle column of M. Using the technique of Section 4.2, this can be done in $O(n^2)$ time. If an entry YY has been detected then we are done. Suppose we have found an entry $M_{ab} = \text{NN}$. Then properties (a) and (b) imply that the top-left submatrix $\{M_{a'b'}\}_{a' \leq a, b' \leq b}$ and the bottom-right submatrix $\{M_{a'b'}\}_{a' \geq a, b' \geq b}$ of M can be discarded from further analysis, because they cannot contain a YY entry. We thus recurse with the remaining bottom-left and the top-right submatrices of M. If no NN entry is detected, then either all entries in the middle column are YN, or all are NY, or there is a single transition from a YN entry to a following NY entry. In the first case we can discard the left submatrix of M, and in the second case we can discard the right submatrix of M. In the third case we can discard, as above, the top-left and the bottom-right submatrices of M. (The difference from the previous case is that, if $M_{ab} = \text{YN}$ and $M_{a+1,b} = \text{NY}$, then now we discard $\{M_{a'b'}\}_{a' \leq a, b' \leq b}$ and $\{M_{a'b'}\}_{a' \geq a+1, b' \geq b+1}$.) In each case we thus recurse on subproblems whose total size is half the size of the original matrix, so the procedure will terminate after logarithmically many steps. The terminal stage is when the current submatrix has only a single column. We then scan this column, as above; if a YY entry has been found, we have an affirmative solution to the 2DC problem. Otherwise, if no YY entry is found in any subproblem, for all possible orientations, we conclude that the currently assumed configuration cannot arise, which implies a negative solution to the 2DC problem, because by now we have exhausted all possible cases. The total cost of this procedure is $O(n^2 \log n)$.

We thus conclude that the 2DC problem, for a set of n points in the plane and for any fixed radius r, can be solved in $O(n^2 \log n)$ time.

5 Conclusion and Future Work

The optimal time to compute the largest empty circle for a set of points in the euclidean plane is $O(n \log n)$ [12]. The restriction being that the center of the circle lies inside the convex hull. Since the spherical minimal enclosing circle is the dual of the maximum empty circle, we can fairly assume that the optimal spherical minimal enclosing circle construction is also $O(n \log n)$. Further work can be to design an incremental algorithms for the spherical 1-center algorithm. This could facilitate us to design solution for some restricted version of the 1-center problem.

We have given an algorithm for the 2-disk cover problem. Further work can be to improve the efficiency of the method in Section 4.2 and also to uplift the restriction on the order of deletion of points. The algorithm can also be used along with Megiddo's [8] parametric searching technique to design a solution for the spherical 2-center problem.

References

[1] L.J. Guibas, D.E. Knuth and M. Sharir, Randomized incremental construction of Delaunay and Voronoi diagrams, Algorithmica 7 (1-6), 381413, 1992.

[2] M.I. Shamos and D. Hoey. Closest-point problems. In Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci., 151-162, 1975.

[3] S.J. Fortune, A sweepline algorithm for Voronoi diagrams. Algorithmica, 2: 153174, 1987.

[4] Xiaoyu Zheng, Roland Ennis, Gregory P. Richards, and Peter Paly-Muhoray, A Plane Sweep Algorithm for the Voronoi Tessellation of the Sphere, electronic-Liquid Crystal Communications, 2011.

[5] Joao Dinis, Margarida Mamede, Sweeping the Sphere, International Symposium on Voronoi Diagrams in Science and Engineering, 2010.

[6] Pankaj K. Agarwal and Micha Sharir, Planar Geometric Location Problems, Algorithmica (1994) 11: 185-195

[7] J.J. Sylvester, A question in the geometry of situation, Quart. J. Math., (1857), p. 79.

[8] N. Megiddo, Linear-time algorithms for linear programming in R^3 and related problems, SIAM J. Comput. 12 (1983), 759776.

[9] J. Hershberger and S. Suri, Offline maintenance of planar configurations, Proc. 2^{nd} ACM-SIAM Sympos. Discrete Algorithms, 1991, 3241.

[10] Micha Sharir, A Near-Linear Algorithm for the Planar 2-Center Problem, 2006

[11] David Eppstein, Faster Construction of Planar Two-centers, Tech. Report 96-12, 1996

[12] G. T. Toussaint, "Computing largest empty circles with location constraints," International Journal of Computer and Information Sciences, vol. 12, No. 5, October, 1983, pp. 347-358.