CHAPTER 2

Spherical Minimax Location Problems

1.1 Problem Formulation

Without loss of generality assume that the sphere has a unit radius, since the arc length is directly proportional to the radius of the sphere. Let (ϕ_i, θ_i) denote the location on the surface of a unit sphere of the i-th demand point where ϕ_i and θ_i represent its latitude and longitude, respectively, and S denote the set of all demand points. The geodesic distance, α_i, between a facility location (ϕ, θ) and the i-th demand point is given by (see, e.g., [45] and [54]).

$$\cos\alpha_i = \sin\phi\sin\phi_i + \cos\phi\cos\phi_i\cos(\theta - \theta_i) \tag{1}$$

Thus the problem on the surface of the sphere can be stated as

$$\min_{(\phi,\theta)} \max_i \{\alpha_i\} \tag{2}$$

Patel [45] has shown that minimizing the maximum of the spherical arc distances between the facility point and the demand points on a sphere is equivalent to minimizing the maximum of the corresponding Euclidean distances. We use this concept to obtain the optimum solution of the spherical and hemispherical minimax location problems.

Given the latitude and longitude of a demand point one can use the following coordinate transformations to obtain the corresponding cartesian coordinates of the point:

$$x = \cos\phi\cos\theta,\ y = \cos\phi\sin\theta,\ z = \sin\phi, \tag{T}$$

where $0 < \theta < \pi$ for east longitude and $-\pi < \theta < 0$ for west longitude, and $0 < \phi < \pi/2$ for north latitude and $-\pi/2 < \phi < 0$ for south latitude.

After obtaining the optimum solution of the problem one can apply the inverse of the transformation (T) to get the latitude and longitude of the facility point. It follows immediately from the above discussion that the inverse of the transformation T is given by:

(i) The latitude $\phi^\circ = \frac{180^\circ}{\pi}\arctan\left|\frac{z}{\sqrt{x^2+y^2}}\right|$ in magnitude . The latitude is north or south according as $z > 0$ or < 0. If $z = 0$ then the point is situated on the equator.

Assume $x \neq 0$. Let $\theta^\circ = \frac{180^\circ}{\pi}\arctan\left|\frac{y}{x}\right|$

If $x > 0$ and $y \geq 0$ then longitude is θ° east.

If $x > 0$ and $y < 0$ then longitude is θ° west.

If $x < 0$ and $y \geq 0$ then longitude is $90^\circ + \theta^\circ$ east.

If $x < 0$ and $y < 0$ then longitude is $90^\circ + \theta^\circ$ west.

If $x = 0$ and $y > 0$ then longitude is 90° east, if $x = 0$ and $y < 0$ then longitude is 90° west.

Prior to establishing the main results we are going to introduce some preliminary concepts on spherical trigonometry.

1.2 Basic Spherical Trigonometry

Basic definitions and results from [54] are restated here for convenience. Formal definitions of the concepts especially connected with the theoretical foundations of the method are also given.

Definition 1.2.1. Every plane section of a sphere is a circle. The largest circle which can be drawn on the surface of a sphere is a circle passing through the centre of the sphere. Such a circle is called a *great circle*. All other circles on the surface of the sphere are called *small circles*.

The above definition leads to the following proposition:

Proposition 1.2.1. *Through any two points on the surface of a sphere, which are not diametrically opposite, one and only one great circle can be drawn.*

Definition 1.2.2. The *poles* of a great circle are the extremities of a diameter of the sphere that is perpendicular to the plane of the great circle. This diameter is also known as the *axis* of the great circle.

Note that the two poles for great circles are equidistant from its plane and the centre of the sphere. The poles and axes of small circles are similarly defined. However, since the plane of a small circle does not contain the centre of the sphere, its two poles are at a different distance from the plane of the small circle, one is nearer and the other is more distant. For convenience, we refer to them as the nearer and distant poles of a small circle.

Definition 1.2.3. Let A and B be two points, not diametrically opposite, on the surface of a sphere. Then, there is a unique great circle that passes through the two points. Moreover, A and B divide the great circle into two arcs. The length of the shorter arc is the *distance* between A and B or the *length of arc joining* A *and* B.

Since the sphere has a unit radius, the length of arc AB is simply the angle (measured in, e.g., radians) between two rays emanating from the centre of the sphere, O, one passing through A and the other through B.

Definition 1.2.4. The surface area of a sphere that is bounded by arc segments of three great

circles is called a *spherical triangle.*

If the spherical triangle has zero area, i.e., the three arcs intersect at a single point, then it is a degenerate spherical triangle, a case not considered in this research. Assume as in Article 22 in [54] that each side of a spherical triangle is less than π. This assumption yields the following proposition:

Proposition 1.2.2. *The sum of any two sides of a spherical triangle is greater than the third.*

1.3 Preliminary Concepts

The following notation will be used throughout the remainder of this chapter.

$A\hat{B}C \equiv$ the *spherical angle* subtended from a point B (on the surface of a sphere) by the (shorter) arc AC.

$\Delta ABC \equiv$ the plane triangle with vertices at points A, B, and C. Denote the angles of ΔABC as $<A, <B, <C$ or, $<BAC, <ABC, <ACB$.

As in [54], the spherical angle ABC is measured as the angle between two lines tangential at point B to the two great circles, one passing through A & B and the other through B & C.

The following definitions are necessary for the development of the algorithm.

Definition 1.3.1. Given three distinct points, A, B, and C, on the surface of a sphere, Π(A, B, C) denotes the unique plane passing through the three points and bisecting the sphere.

Definition 1.3.2. $\Gamma 3$(A, B, C) denotes the circle traced by the plane Π(A, B, C) cutting through the sphere.

Generally, $\Gamma 3$(A, B, C) is expected to be a small circle. However, it is possible that $\Gamma 3$(A, B, C) is a great circle.

Definition 1.3.3. Let A and B be two distinct points on the surface of a sphere that are not diametrically opposite. Denote the mid point of the (shorter) arc AB as the point P. Then, $\Gamma 2$(A,B) represents the small circle that goes through points A and B and has its nearer pole located at point P.

Definition 1.3.4. The length of the great circle arc from any point on the circumference of a small circle to its nearer pole is called the spherical radius of the small circle.

Definition 1.3.5. $N\Gamma 2$(A, B) and $N\Gamma 3$(A, B, C) denote the surface area of a sphere that contains the nearer pole of and is bounded by $\Gamma 2$(A, B) and $\Gamma 3$(A, B, C), respectively.

Definition 1.3.6. $R\Gamma 2$(A, B) and $R\Gamma 3$(A, B, C) denote the surface area of a sphere that contains the distant pole of and is bounded by $\Gamma 2$(A, B) and $\Gamma 3$(A, B, C), respectively.

Global Minimax Location Problem

2.1 Lemmas Related to the Algorithm

Before establishing the validity of the algorithm, presented in section 2.2, of the problem mentioned in section 1.1, we prove several lemmas.

Lemma 2.1.1. Let P be the nearer pole of $\Gamma 3(A, B, C)$, where ΔABC is an acute triangle. Let $Q(\neq P)$ be any point on $N\Gamma 3(A, B, C)$ and within the spherical triangle ABC. Then, the spherical radius of $\Gamma 3(A, B, C)$ is greater than minimum {QA, QB, QC}, where QA, QB, QC denote the geodesic distances between Q and A, Q and B, and Q and C, respectively.

Proof. In Figure 1, O denotes the centre of the circle $\Gamma 3(A, B, C)$. Then A_1, B_1, and C_1 are the points on the circumference of the circle that are diametrically opposite of A, B, and C, respectively. Since ΔABC is acute, points B and C cannot lie on the same side of the line joining A and A_1. The same is true for points C and A and the line joining B and B_1, and points A and B and the line joining C and C_1. Let D be any point of the circumference of $\Gamma 3(A, B, C)$, then obviously

$$\text{minimum } \{ \angle AOD, \angle BOD, \angle COD \} < \frac{\pi}{2}.$$

Extend the arc from P passing through Q to meet the circumference of $\Gamma 3(A, B, C)$ at point D. Without any loss of generality, assume

$$\angle AOD < \frac{\pi}{2}, \text{ i.e., } Q\hat{P}A < \frac{\pi}{2} \text{ and } \angle AOD \leq \angle BOD.$$

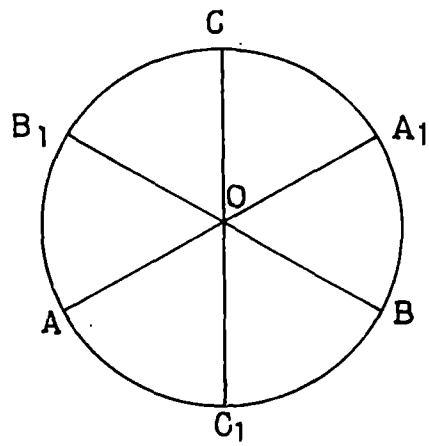


Figure - 1.

If Q lies on arc PA, then the proof is complete. When Q does not lie on arc PA, let M be the midpoint of the shorter arc segment between points A and B on the circumference of $\Gamma 3(A, B, C)$ (see Figure 2). Construct two great circle arcs, one joining points P and M and the other joining points A and Q. Extend arc AQ to meet arc PM at point T. By construction, the lengths of arcs BM and AM are the same. Since P is the nearer pole of $\Gamma 3(A, B, C)$, arcs PA and PB are of the same length also. Thus, the spherical triangles AMP and BMP are congruent and $A\hat{M}P = \frac{\pi}{2}$. From Article 42 in [54], we have

$$\cos(PA) = \cos(PM)\cos(AM) \tag{3}$$

$$\cos(TA) = \cos(TM)\cos(AM) \tag{4}$$

Now using the result PM > TM, we get from (3) and (4),

$$PA > TA \geq QA$$

Since PA is the spherical radius of $\Gamma 3(A, B, C)$, the proof is complete. □

Lemma 2.1.2. Let P_1 be the distant pole of $\Gamma 3(A, B, C)$ where ΔABC is an acute triangle. Let Q_1 be a point on $R\Gamma 3(A, B, C)$ and $Q_1 \neq P_1$. If Q_1 is sufficiently close to P_1 then

$$\text{maximum } \{AQ_1, BQ_1, CQ_1\} > AP_1$$

Proof. Let P be the nearer pole of $\Gamma 3(A, B, C)$. Let Q be the diametrically opposite point to Q_1. Obviously, Q is on $N\Gamma 3(A, B, C)$ and $P \neq Q$. Since Q_1 is sufficiently close to P_1 and P is in the spherical triangle ABC, Q must be in the spherical triangle as well. Assume that

$$QA = \text{minimum } \{QA, QB, QC\}.$$

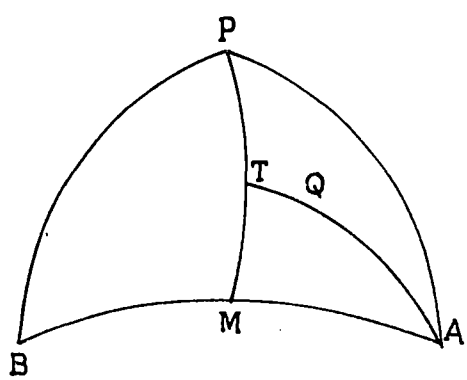


Figure - 2.

Then from Lemma 2.1.1 it follows that QA < PA. Construct two great circle arcs, one joining A to P_1 and the other joining A to Q_1. Since P and Q are diametrically opposite of P_1 and Q_1, respectively, we have

$$AP + AP_1 = \pi = AQ + AQ_1$$

Now $AQ < AP \Rightarrow AP_1 < AQ_1$. This proves lemma 2.1.2 . □

Lemma 2.1.3. Let A, B, and C be three points on a unit sphere with $<C> \frac{\pi}{2}$. Let P and P_1 be respectively the nearer and distant poles of Γ3(A, B, C). Then there exist points, Q say, close to P_1, such that arc CQ < arc AQ < arc AP_1.

Proof. To establish lemma 2.1.3 we make use of the following properties of a spherical triangle (see Articles 35 and 37 in [54]).

(a) The angles at the base of an isosceles spherical triangle are equal.

(b) If one angle of a spherical triangle is greater than another, the side opposite the greater angle is greater than the side opposite the smaller angle.

In figure 3, M denotes the mid point of the great circle arc AB. Since P_1 is the distant pole of Γ3(A, B, C), we have

$$\text{arc } AP_1 = \text{arc } BP_1 = \text{arc } CP_1 \tag{5}$$

Hence spherical triangles AMP_1 and BMP_1 are congruent (see Article 34 in [54]) and

$$B\hat{M}P_1 = \frac{\pi}{2} \tag{6}$$

Again $<C> \frac{\pi}{2}$ implies that C lies on the arc of the small circle AB. Without loss of generality, assume B and C to lie on the same side of the great circle arc PMP_1. Take a point Q on the great circle arc P_1M. Construct two great circle arcs, one joining Q to B and the other joining Q to C.

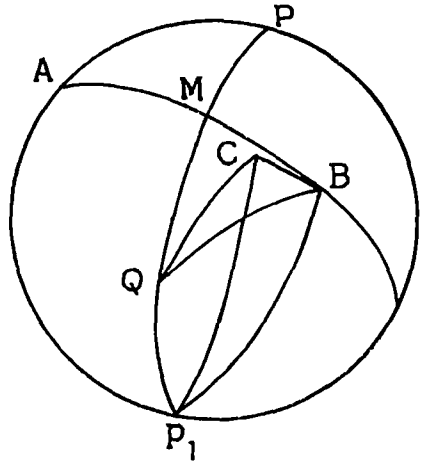


Figure - 3

Since arc BP_1 = arc CP_1 it follows from property (a) that

$$C\hat{B}P_1 = B\hat{P}C_1 \tag{7}$$

Since Q lies on arc MP_1 (see figure 3), we obtain

$$C\hat{B}Q < C\hat{B}P_1 \tag{8}$$

and $$B\hat{C}Q > B\hat{C}P_1 \tag{9}$$

It follows from property (b) and results (7) through (9),

$$\text{arc } BQ > \text{arc } CQ \tag{10}$$

Using (7), we get from the spherical triangles BMQ and BMP_1, (see Article 42 in [54])

$$\cos(BQ) = \cos(BM)\cos(MQ) \tag{11}$$

and $$\cos(BP_1) = \cos(BM)\cos(MP_1) \tag{12}$$

Since arc MQ < arc MP_1, we obtain from (11) and (12)

$$\text{arc } BQ < \text{arc } BP_1 \tag{13}$$

Again since spherical triangles AMQ and BMQ are congruent (see Article 34 in [54]), we get

$$\text{arc } AQ = \text{arc } BQ \tag{14}$$

Combining (14), (13), (10), and (5) we get the required result. □

It is to be noted that if $<C> \frac{\pi}{2}$ then from Lemma 2.1.3 it follows that the distant pole of $\Gamma 3(A, B, C)$ cannot be a solution of the spherical minimax location problem. Moreover, Lemma 2 of [50] implies that for $<C> \frac{\pi}{2}$, the nearer pole of $\Gamma 3(A, B, C)$ cannot yield the optimum solution of the hemispherical minimax location problem.

If one of the stated conditions in Theorem 2.1.1 below is true then we obtain the optimum solution of the hemispherical minimax location problem. The proof of Theorem 2.1.1 follows from lemmas 4 and 2 of [50].

Theorem 2.1.1. (i) If ΔABC is an acute triangle and $N\Gamma 3(A, B, C)$ contains all demand points then the nearer pole of $\Gamma 3(A, B, C)$ is the unique facility point.

(ii) If $N\Gamma 2(A, B)$ contains all demand points, then nearer pole of $\Gamma 2(A, B)$ is the required facility. □

We now state and prove a theorem regarding optimal solution of the global minimax problem.

Theorem 2.1.2. If there exists a triplet (A, B, C) of demand points such that

(i) ΔABC is acute,

(ii) The centre of the sphere and all demand points lie on the same side of $\Pi(A, B, C)$, and

(iii) (A, B, C) generates the plane closest to the centre of the sphere, then the distant pole of $\Gamma 3$(A, B, C) is the required facility point.

Proof. Lemma 2.1.3 implies that the triplet of points forming an obtuse triangle cannot yield an optimal solution. Furthermore, it follows from lemma 2.1.2 that the distant pole of the small circle defined by a triplet satisfying (i) and (ii) is a local minimum and (iii) implies optimality. □

It follows from lemma 2.1.3 that for a global minimax problem we have to consider only acute triangles, and there are at most n(n-1)(n-2)/6 acute triangles. Consequently, from Theorem 2.1.2 after a finite number of arithmetic operations we obtain the solution of the spherical minimax location problem. Our algorithm for global optimization is based on theorems 2.1.1 and 2.1.2. We are now going to present our algorithm.

2.2 Algorithm

A few remarks are in order before discussing the algorithm:

Denote the set of demand points by $S = \{A_i : i = 1, 2, ..., n\}$. For convenience, from now on we will refer to the existing demand points as points. Throughout the algorithm the set {A, B, C} is the most recent triplet of points defining Π(A, B, C). The parameter d_1 denotes the most recent Euclidean distance between Π(A, B, C) and the centre of the sphere (d_1 being simply the distance between the centres of the circle and the sphere) when ΔABC is an acute triangle, and d refers to the corresponding quantity in the previous iteration. Take the initial value of d equal to the radius of the unit sphere. Let (X, Y) denote the initial pair of demand points. The set S1 will contain the triplet of demand points corresponding to the local minimum. Initially S1 = ⌀. If a local minimum obtained at a particular iteration is better than the one calculated with the current contents of S1, then we update S1. The set S2 contains the pair of demand points (X, Y). During the course of a search as soon as a triplet of points forming a non-acute triangle is encountered, we update S2 with the ends of the largest side of the triangle provided the Euclidean distance between the new pair of points is greater than that between the points stored in S2.

If i, j, and k are three positive integers such that $i < j < k$, and n(>2) is an integer. Then to update the triplet (i, j, k) we mean one of the following:

(i) If $k < n$ then

$i \leftarrow i, j \leftarrow j, k \leftarrow k + 1$

(ii) If $k = n$ and $j < n - 1$ then

$i \leftarrow i, j \leftarrow j + 1, k \leftarrow j + 1$

(iii) If $k = n, j = n - 1$ and $i < n - 2$ then

$i \leftarrow i + 1, j \leftarrow i + 1, k \leftarrow j + 1$

Global Minimax Algorithm

Initial Step. $i \leftarrow 1, j \leftarrow 2, k \leftarrow 3, d \leftarrow 1, S1 \leftarrow \varnothing, A \leftarrow A_i, B \leftarrow A_j, C \leftarrow A_k, X \leftarrow A, Y \leftarrow B,$ $S2 = \{ X, Y \}$, ρ = Euclidean distance between X and Y. Go to Step 1.

Step 1. Find d_1. If ΔABC is acute and $d_1 \leq d$ then go to Step 2. Otherwise, go to Step 4.

Step 2. If all points of $S - \{A, B, C\}$ lie on $N\Gamma 3(A, B, C)$ then the nearer pole of $\Gamma 3(A, B, C)$ is the required facility point of a hemispherical location problem; stop.

Else go to Step 3.

Step 3. If all points of $S - \{A, B, C\}$ lie on $R\Gamma 3(A, B, C)$ then the distant pole of $\Gamma 3(A, B, C)$ is a local minimum; $d \leftarrow d_1$, update S1. Go to Step 5.

Step 4. If ΔABC is not an acute triangle then find d_2, the length of the largest side of ΔABC. If $d_2 > \rho$ then $\rho \leftarrow d_2$ and update S2. Go to Step 5.

Step 5. If $i = n - 2, j = n - 1$, and $k = n$ then go to Step 6.

Else update the triplet (i, j, k). $A \leftarrow A_i, B \leftarrow A_j, C \leftarrow A_k$ and repeat Step 1.

Step 6. If $d_2 = 2$, the diameter of the sphere, then if there exists a great circle passing through a pair (X, Y) of S2 and containing a point of $S - \{X, Y\}$ such that all other points of S lie on one side of this great circle then the poles of the great circle are the required facility points; stop.

If $d_2 < 2$, and $N\Gamma 2(X, Y)$ contains all points then the nearer pole of $\Gamma 2(X, Y)$ is the facility point of the hemispherical location problem; stop.

Else, the distant pole of the triplet in S1 is a facility point.

Remarks

1. If we consider all possible triplets of points taken from the set S of n points and we verify whether the remaining n-3 points not considered in a particular triplet lie on the same side of the plane passing through this triplet then the algorithm is $O(n^4)$.

2. It is to be noted that the plane of $\Gamma 2(A, B)$ divides the sphere into two disjoint surfaces. For any point $C \in R\Gamma 2(A, B)$, $\angle ACB$ is acute. On the other hand if $D \in N\Gamma 2(A, B)$ then $\angle ADB$ is obtuse. Hence for two given points A and B, the triplet (A, B, C) yields an optimum solution

provided C $\in$ RΓ2(A, B), ΔABC is acute and all points of the set S - {A, B, C} lie on one side of Π(A, B, C). Now RΓ2(A, B) may contain at most n - 2 points. Consequently, there are no more than n - 2 planes through A, B and one of the points of RΓ2(A, B) at a time. Among these planes only two planes contain points of RΓ2(A, B) on one side. These two planes have the characteristic property that they have the maximum and minimum inclinations with the plane of Γ2(A, B). Now O(n) arithmetic operations will be needed to determine these maximum and minimum inclinations, and O(n) additional arithmetic operations are required to know whether the remaining points lie on one side of the plane and if so whether the distance, d_1, of the plane from the centre of the sphere is less than d, the previous distance. Therefore for a given pair (A, B), O(n) arithmetic operations will be needed to get an optimum provided such a solution exists. Since A and B are any two points of S, for different A and B we can construct n(n-1)/2 different Γ2(A, B) each requiring O(n) arithmetic operations for testing optimality. Taking the above facts into account, the algorithm turns out to be $O(n^3)$ complex.

3. The algorithm presented here determines all global minimax facility points and also finds the hemispherical minimax location point. It also determines whether all points lie on a hemisphere or not.

2.3 Numerical Results

In this section we are going to obtain the solutions of two minimax location problems. These problems are taken from Patel's paper [45]. Problem 1 finds the solution of the global minimax problem and problem 2 obtains the solution of the hemispherical minimax problem.

Problem 1. The Cartesian coordinates of 14 points are given in Table 1.

The coordinates of the optimal point are

(0.967, -0.230, -0.108)

The Euclidean distance between the facility point and the farthest demand point is 1.6745 and the corresponding geodesic distance is 1.985. The number of local optima encountered during the execution of the algorithm is 5. The distant pole of the small circle, defined by the 5-*th* and 13-*th* and 14-*th* points, is the required minimax point.

It is to be noted that Patel [45] predicted the coordinates of the optimal facility as (0.223, 0.077, 0.972), and the Euclidean distance between the farthest demand point and the optimal facility as 1.701. But this result is not correct. Such inaccuracies are usual, because generating

all local solutions may, at times, be very difficult to implement in practice.

Table 1: Coordinates of 14 points on a unit sphere

Points	x	y	z
1	0.5105	0.2209	0.8310
2	0.8949	-0.1433	0.4226
3	0.7235	0.6794	-0.1219
4	0.6895	-0.6895	0.2215
5	-0.1822	0.9832	0.0000
6	0.0854	-0.8869	0.4540
7	0.3422	-0.9250	-0.1650
8	0.0441	0.8422	-0.5373
9	0.4330	-0.7500	-0.5000
10	0.2500	-0.4330	0.8660
11	0.1830	0.6830	0.7071
12	0.0872	0.0000	0.9962
13	-0.6209	-0.7399	-0.2588
14	-0.2113	0.4532	0.8660

Problem 2. Table 2 below shows the latitudes and longitudes as also the corresponding cartesian coordinates of these 15 points situated in the northern hemisphere.

The solution of this problem is the nearer pole of $\Gamma 2(A, B)$, where A and B denote the points Paris and Manila, respectively. The cartesian coordinates of the facility point are (0.1191, 0.6435, 0.7561) and the corresponding latitude and longitude are 49.12°N and 79.51°E, respectively

To solve Problems 1 and 2 we have developed the PASCAL Code (Borland's Turbo Pascal Version 6) of the above problems and used a DX2 66 MHz PC AT to implement the programs. The CPU times to get the optimal solutions for both the problems is 0.06 sec. Sarkar and Chaudhuri [50] solved Problem 2 using the same Computer and Compiler and the CPU time

involved was 0.16 sec. Patel [45] solved both the problems in about 5 secs. of CPU time. Patel [45] derived the result in a Convex 220 main frame computer.

Table 2. Latitudes and Longitudes, and corresponding cartesian coordinates of 15 points.

City/Point	ϕ	θ	x	y	z
London, England	51.5N	0.4E	0.6225	0.0043	0.7826
Paris, France	48.9N	2.3E	0.6568	0.0264	0.7536
Zurich, Switzerland	47.5N	8.5E	0.6694	0.1000	0.7361
Rome, Italy	41.9N	12.5E	0.7267	0.1611	0.6678
Copenhagan, Denmark	55.7N	12.6E	0.5500	0.1229	0.8261
Berlin, Germany	52.5N	13.4E	0.5922	0.1411	0.7934
Stockholm, Sweden	59.3N	18.9E	0.4830	0.1654	0.8600
Athens, Greece	38.0N	23.7E	0.7216	0.3167	0.6157
Ankara, Turkey	39.9N	32.8E	0.6449	0.4156	0.6415
Telaviv, Israel	32.1N	34.8E	0.6956	0.4835	0.5314
Moscow, Russia	55.7N	37.7E	0.4459	0.3446	0.8261
Teheran. Iran	35.4N	51.4E	0.5058	0.6370	0.5793
Bombay, India	18.9N	72.8E	0.2798	0.9038	0.3239
Manila, Philipines	14.6N	121.0E	-0.4984	0.8295	0.2521
Tokyo, Japan	35.6N	139.7E	-0.6201	0.5260	0.5820

Computational Results

We have considered 5 sets containing 10, 20, 50, 80 and 100 data points distributed at random over a unit sphere. Each of the above set was randomly generated a hundred times. Table 3 shows the results of computation.

Table: 3.

No. of points	Maximum No. of local minimum	Average CPU time in secs.
10	5	0.02
20	8	0.22
50	8	4.11
80	7	17.58
100	8	34.95

2.4 Conclusion

We have shown in this paper that solving the global minimax location problem with the geodesic norm when the points are spread over a sphere is to find the plane, closest to the centre of the sphere, containing three points forming an acute triangle. The process of optimization consists in finding a sequence of planes, containing acute triangles, of gradually diminishing distances from the centre of the sphere, provided such acute triangles exist.

At a particular iteration, if a triplet of points (A, B, C) forming an acute triangle is such that NΓ3(A, B, C) contains all points then the nearer pole of Γ3(A, B, C) is the required unique optimum location point; but if NΓ2(A, B) contains all points then the corresponding location point is the nearer pole of Γ2(A, B). On the contrary, if there exists neither NΓ3(A, B, C) nor NΓ2(A, B) having properties mentioned above then the distant pole of Γ3(A, B, C) is a required facility point provided Π(A, B, C) is closest to the centre of the sphere and ΔABC is acute.

In developing our algorithm we have made use of the distance between the centre of the sphere and the centre of the circle Γ3(A, B, C). The equation of the plane, containing three points A, B, and C, is

$$ax + by + cz + p = 0, \text{ where } a^2 + b^2 + c^2 = 1,$$

(a, b, c, p) being determined by the coordinates of A, B, and C. If Π(A, B, C) is closest to the centre of the sphere, then for a spherical minimax location problem, (a, b, c) is the facility point provided $p > 0$; for $p < 0$, (-a, -b, -c) is the facility point; when $p = 0$ either pole of the great circle containing A, B, C will represent the facility point.

For a hemispherical minimax location problem the coordinates of the facility point are (a, b, c) when $p < 0$ and (-a, -b, -c) when $p > 0$.

If the nearer pole of Γ2(A, B) is the facility point of a hemispherical minimax location problem then the coordinates of the facility point are (aλ, bλ, cλ) where (a, b, c) are the coordinates of the midpoint of the line segment AB and

$$\lambda = \frac{1}{\sqrt{a^2 + b^2 + c^2}}$$

2.5 Pascal Program of the Global Minimax Algorithm

In this section we develop the Pascal program corresponding to the global minimaxlocation algorithm given in section 2.2.

```
program spherical_location(input,output,infile);
{This program obtains exact solution, in finite number of steps, of the global minimax
location problem by using selective enumeration technique. In case of a global minimax
problem, it obtains all optimum solution when solution is not unique. This program also
determines the unique optimum solution of the hemispherical minimax problem.}
 uses crt,dos;
 type list=array[1..100] of real;
 var
  infile:text;
  x,y,z:list; {these three vectors denote the coordinates of the demand points }
  i,j,k,i1,i2,i3,j1,j2,j3,n,n1,n2,k1,k2,p1,p2:integer;
  a,b,c,d1,d2,e1,e2,e3,u,v,v1,dd:real;
  flag,aflag,count:boolean;
  hh,mm,ss,hs:word;
 {p1, p2 represent indices of the ends of the largest side of a non acute triangle. In a course of
a search as soon as a larger side is encountered, one has to update the values of p1 and p2. dd
is the length of the corresponding side. i1, i2, i3, denote the indices of vertices of an acute
triangle. We update these values when we get a better solution.
 u = the square of the diameter of the circle containing three points}
 procedure plane(p,q,r:integer);
{this procedure determines the equation to the plane through three demand points}
  begin        {constants a,b and c are the direction ratios of the normal to the plane}
   a:=(y[q]-y[p])*(z[r]-z[p])-(y[r]-y[p])*(z[q]-z[p]);
   b:=(z[q]-z[p])*(x[r]-x[p])-(z[r]-z[p])*(x[q]-x[p]);
   c:=(x[q]-x[p])*(y[r]-y[p])-(x[r]-x[p])*(y[q]-y[p]);
   d1:=a*x[p]+b*y[p]+c*z[p]
  end;{end of the procedure plane}
 procedure distance(var dist:real;r,s:integer);
{this procedure finds the square of the Euclidean distance between two points}
  begin
   dist:=sqr(x[r]-x[s])+sqr(y[r]-y[s])+sqr(z[r]-z[s])
  end;{end of the procedure distance}
```

```
procedure interchange(var e11,e22:real; var ii,jj:integer);
  var kk:integer;
{this procedure obtains the maximum of two numbers and the indices corresponding to these
numbers}
 begin
    if e11<e22 then
      begin
        v:=e11;kk:=ii;e11:=e22;e22:=v;ii:=jj;jj:=kk
      end
  end;{end of the procedure interchange}
 procedure  acute(s1,s2,s3:real); {this procedure tests whether three points form an acute
triangle}  begin
    j1:=i;j2:=j;j3:=k;aflag:=true;
{ when the value of the boolean variable aflag = false then we have to update the triplet (i, j,
k)}
   interchange(s1,s2,j1,j3);
    interchange(s1,s3,j2,j3);
    if s1>=s2+s3 then{this condition states that the triangle is not acute, consequently aflag =
false}
      begin
        aflag:=false;
        if s1>dd then
        { this condition implies that the largest side of the non-acute triangle is greater than the
previous value of the corresponding quantity, so one has to update the values of p1, p2 and
dd}
         begin
           p1:=j1;p2:=j2;dd:=s1;
         end
      end {end of s1>=s2+s3}
    else          {this condition states the triangle having vertices i, j, k is an acute triangle}
      begin
```

```
        v1:=(s1)/(1-sqr(s2+s3-s1)/(4*s2*s3));   {v1= square of the diameter of the small
circle}
        if v1<=u then aflag:=false
        {this condition implies that the radius of the circle through points i, j, and k is not
greater than the corresponding previous quantity  and one need updating the values of  i, j, k}
      end                                {v1>u implies better solution and aflag=true}
  end;{end of the procedure  acute}
 procedure test;
 {tests whether the points are on the same or opposite sides of the origin with respect to the
plane through points i, j, k}
   var l:integer;
   begin
    {k1>0 implies that there exists at least one point such that the centre of  the sphere and this
point lie on the opposite sides of the plane defined by the triplet (i, j, k)}
     k1:=0;k2:=0;l:=1;flag:=true;plane(i,j,k);
    {k2>0 implies that there exists at least one point such that the centre of  the sphere and this
point lie on the same side of the plane defined by the triplet (i, j, k)}
     while flag and (l<=n) do
      begin
       if (l<>i) and (l<>j) and (l<>k) then
        begin
         d2:=a*(x[l]-x[i])+b*(y[l]-y[i])+c*(z[l]-z[i]);
         if d1*d2>0 then k1:=k1+1 else k2:=k2+1
        end;          {end of test that points lie on the same side of the plane }
      {d1*d2>0 implies centre of the sphere and the point l lie on the opposite sides of the
plane defined by the triplet (i, j, k)}
       if (k1>0) and (k2>0) then flag:=false;
       l:=l+1;  {flag=false implies points do not lie on the same side of the plane defined by the
triplet (i, j, k)}
      end;{end of while}
     if flag=true then    {flag=true implies that the points lie on the same side of the plane
defined by the triplet (i, j, k)}
```

```
      begin
        u:=v1;i1:=i;i2:=j;i3:=k          {k1=0 and flag=true determines local optimum solution}
      end    {k2=0 or k1=n-3 implies optimal solution of the hemispherical problem defined by
three points}
  end;{end of procedure test}
 procedure optpoint;  {this procedure determines Cartesian coordinates of the optimum
point}
  begin
    plane(i1,i2,i3);
    d2:=1/sqrt(a*a+b*b+c*c);
    if d1>0 then
      begin
        a:=-a;b:=-b;c:=-c
      end;
    a:=a*d2;b:=b*d2;c:=c*d2
  end;{end of the procedure optpoint}
 procedure two_point;      {this procedure finds the coordinates of the facility point of the
hemispherical location problem when it is determined by two demand points}
  begin
   a:=(x[p1]+x[p2])/2;b:=(y[p1]+y[p2])/2;c:=(z[p1]+z[p2])/2;
   flag:=true;i:=1;
   writeln('p1, p2 ',p1,' ',p2);
   while (i<=n) and flag do
    begin
     if (i<>p1) and (i<>p2) then
       if  (x[i]-a)*a+(y[i]-b)*b+(z[i]-c)*c<0 then flag:=false;
       i:=i+1
    end
  end;{end of the procedure two_point}
 procedure latitude;     {this procedure finds latitude of the facility point}
  begin
    if c<0 then i:=-1;              {i=-1 implies south latitude}
```

```
    v:=d1*arctan(abs(c/sqrt(a*a+b*b)));
  end; {end of the procedure latitude}
procedure longitude;          {this procedure finds the longitude of the facility point}
  begin
   if a<>0 then v1:=d1*arctan(abs(b/a));
   if (a>0) and (b<0) then j:=-1;      {j=-1 implies longitude is west}
   if (a<0) and (b<0) then
     begin
      j:=-1;v1:=90+v1
     end;
   if (a<0) and (b>0) then v1:=90+v1;
  end; {end of the procedure longitude}
begin   {main action block}
  clrscr;
  assign(infile,'file1.hem');          {file1.hem contains coordinates of the demand points}
  reset(infile);
  writeln('supply the number, n, of demand points');
  readln(n);
  for i:=1 to n do
    readln(infile,x[i],y[i],z[i]);
  gettime(hh,mm,ss,hs);
  writeln(hh,':',mm,':',ss,'.',hs); {dd denotes the largest side of a non-acute triangle and p1, p2
represent the indices of the end point of  this side}
  u:=0;i1:=0;i2:=0;i3:=0;i:=1;n1:=n-2;n2:=n-1;count:=true;dd:=0;   {u gives the information
about the radius of the circle through the vertices, i, j and k of the acute angled triangle}
  while (i<= n1) and count do
{boolean variable count is used to determine whether there is an  acute angled triangle, having
vertices i, j, k , which yields the solution of the hemispherical minimax location problem}
    begin
     j:=i+1;
     while (j<=n2) and count do
       begin
```

```
        distance(e1,i,j);
        k:=j+1;
        while (k<=n) and count do
          begin
            distance(e2,j,k);
            distance(e3,k,i);
            acute(e1,e2,e3);
            if aflag=true then test;  {the value of the boolean variable aflag = true implies that
an acute triangle may have better solution}
            k:=k+1;
{the condition k1=n-3 implies optimum solution of a hemispherical minimax location problem}
            if k1=n-3 then count:=false
          end;{end of k loop}
        j:=j+1
      end;{end of loop j}
    i:=i+1;
  end; {end of i while}
  if count=false then  {count =false implies a triplet of the point (i, j, k) determines the
optimum solution of a hemispherical  minimax location problem}
    begin
      writeln('a hemispherical minimax problem');
      optpoint;
      a:=-a;b:=-b;c:=-c;
      writeln('Cartesian coordinates of the facility points are:');
      writeln(a,',',b,',',c)
    end
  else   {count=true implies either solution of a hemispherical minimax problem obtained by
two points or global problem}
    begin
      two_point;
      if flag=true  then
{flag = true implies solution a hemispherical problem generated by  two demand points}
```

```
      begin
        writeln('two point hemispherical problem');
        dd:=1/sqrt(a*a+b*b+c*c);
        a:=a*dd;b:=b*dd;c:=c*dd;
        writeln('Cartesian coordinates of the facility points are:');
        writeln('(',a:4:2,', ',b:4:2,', ',c:4:2,')')
      end
    else                    {flag=false yields solution of a global problem}
      begin
        writeln('a global minimax problem');
        optpoint;           {this procedure finds the coordinates of the optimum point}
        writeln('Cartesian coordinates of the facility points are:');
        writeln(a:4:2,',',b:4:2,',',c:4:2)
      end
  end;
d1:=45/arctan(1);
latitude;
longitude;
if i=-1 then {this condition implies that the latitude is south}
   writeln('latitude is ',v:4:2,'S')
else          {this condition implies that the latitude is north}
   writeln('latitude is ',v:4:2,'N');
 if j=-1 then {this condition implies that the longitude is west}
    writeln('longitude is ',v1:4:2,'W')
 else         {this condition implies that the longitude is east}
    writeln('longitude is ',v1:4:2,'E');
gettime(hh,mm,ss,hs);
writeln(hh,':',mm,':',ss,'.',hs);
close(infile)
end.{end of action block}
```

Solution of a Single Facility Location Problem on a Hemisphere using the Geodesic Norm

3.1 Lemmas and Theorems Related to the Algorithm

In this section we will present some basic tools which will be used to develop the algorithm, in section 3.2, of the problem given in section 1.1. The following theorems will be needed to develop our theory, the proofs of which are given in [54].

Theorem 3.1.1. *Any two sides of a spherical triangle are together greater than the third side.*

Theorem 3.1.2. *If one side of a spherical triangle is greater than another, the angle opposite the greater side is greater than the angle opposite the smaller side.*

Identical and symmetrical equality of triangles (ISET) [54]:

Two spherical triangles on the same sphere are either congruent or symmetrically equal, and have all their corresponding elements equal, when two sides and the included angle of one are equal to the corresponding quantities of the other.

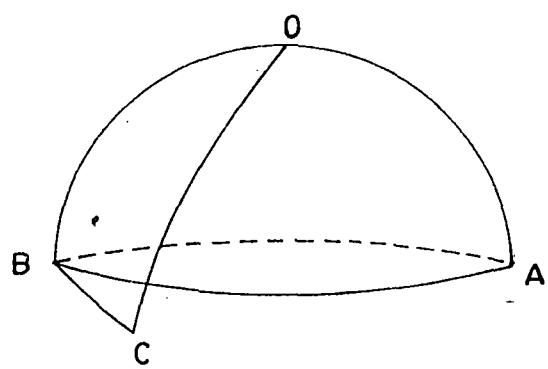
Lemma 3.1.1. *If* $C \in \Sigma\Gamma 2(A, B)$, *then*

$$A\hat{C}B \geq A\hat{B}C + B\hat{A}C \qquad (1)$$

and conversely (1) *implies* $C \in \Sigma\Gamma 2(A, B)$.

The equality sign occurs when $C \in \Gamma 2(A, B)$.

Proof. The result of the first part is given in [50]. Let us prove the converse, i.e., if (1) holds, then we are to prove $C \in \Sigma\Gamma 2(A, B)$. Assume that O is the nearer pole of $\Gamma 2(A, B)$ (see Fig.1). If possible, let $C \notin \Sigma\Gamma 2(A, B)$. Construct great circle arc joining C to B.



FIG·1

$C \notin \Sigma\Gamma 2(A, B) \Rightarrow OC > OB = OA.$

Consider the spherical triangles OCB and OAC. From Theorem 3.1.2 we have

$$O\hat{B}C > O\hat{C}B \text{ and } O\hat{A}C > O\hat{C}A$$

Adding the above inequalities we get,

$$A\hat{B}C + B\hat{A}C > A\hat{C}B$$

which contradicts (1), completing the proof of the converse. □

Lemma 3.1.2. *For a given* $\Gamma 3(A, B, C)$ *if* $A\hat{C}B > A\hat{B}C + B\hat{A}C$ *then* $\Sigma\Gamma 2(A, B)$ *contains* C *and the SR of* $\Sigma\Gamma 2(A, B) <$ *SR of* $\Sigma\Gamma 3(A, B, C)$.

Lemma 3.1.3. *Let* O *be the nearer pole of* $\Gamma 3(A, B, C)$. *Further, if*

$$\hat{A} < \hat{B} + \hat{C},\ \hat{B} < \hat{C} + \hat{A} \text{ and } \hat{C} < \hat{A} + \hat{B}$$

then there exists no $\Sigma\Gamma 3(P, Q, R)$ *containing* A, B, C *and having SR of* $\Gamma 3(P, Q, R) <$ *SR of* $\Gamma 3(A, B, C)$.

Proofs of lemmas 3.1.2 and 3.1.3 are given in [50].

Lemma 3.1.4. *Let* L *be the mid point of* AB. *If* C *and* B *lie on the same side of the plane of the great circle* γ *through* L *and orthogonal to* AB *then* AC > BC. *Conversely,* AC > BC *implies* B *and* C *lie on the same side of the plane through* γ.

Proof. Join A and C to intersect γ at D and also join B to C, and B to D (see Fig.2). ISET property implies AD = BD.

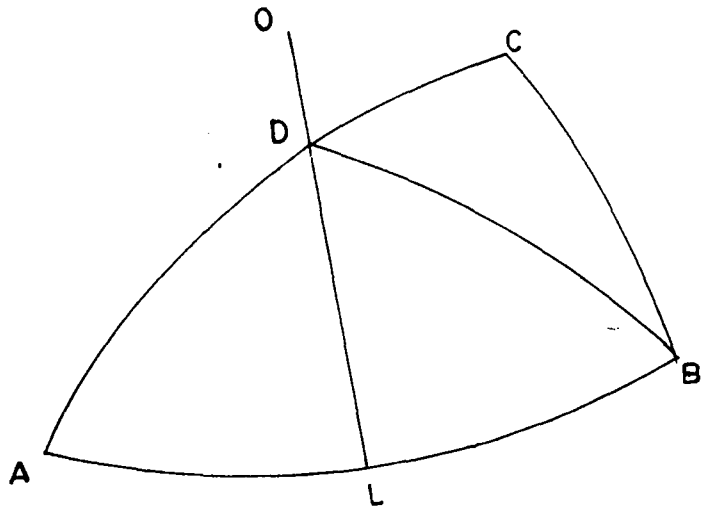


FIG. 2

It follows from Theorem 3.1.1,

$$BD + CD > BC$$

$$\Rightarrow AC > BC.$$

To prove the converse, let us assume that A and C lie on the same side of γ. It follows from the result mentioned above

$$BC > AC,$$

a contradiction. □

Lemma 3.1.5. *Let* D *be a point on* AC *produced. Assume that* AC *and* AD *are both less than* π*;* B *is a point not on the great circle* AC*, and* $B \in \Sigma\Gamma 3(A, C, D)$*. Suppose that the triplets* A, B, C *and* A, B, D *satisfy the conditions of Lemma* 3.1.3*, then*

$$SR \text{ of } \Gamma 3(A, B, C) < SR \text{ of } \Gamma 3(A, B, D).$$

Proof. Let us draw great circles γ_1 and γ_2 through the mid points of AC and AD, respectively, and perpendicular to AD. If γ_1 and γ_2 intersect at P, then $PC = PD = \pi/2$. Consequently, the nearer pole, O, of $\Gamma 3(A, B, C)$ and A lie on the same side of γ_2. It follows from Lemma 3.1.4, $OD > OC$, i.e., $D \notin \Gamma 3(A, B, C)$. Now applying Lemma 3.1.3 we get the result of Lemma 3.1.5.

□

Lemma 3.1.6. *Let* A *and* B *be two given points;* P_1 *and* P_2 *are situated on the same side of the plane of the great circle* AB*. Assume that the spherical angles of the triangles* ABP_1 *and* ABP_2 *satisfy the conditions of lemma 3.1.3. Then* $\theta_1 > \theta_2$ *implies that*

$$SR \text{ of } \Gamma 3(A, B, P_1) > SR \text{ of } \Gamma 3(A, B, P_2)$$

and P_2 *is contained in* $\Sigma\Gamma 3(A, B, P_1)$.

Proof. Let L be the mid point of AB (see Fig.3), and O, O′ be the nearer poles of $\Gamma 3(A, B, P_1)$

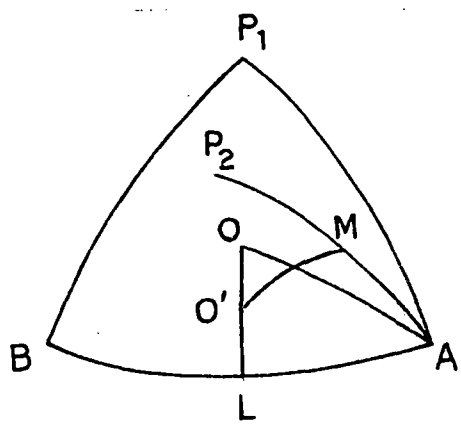


Fig. 3.

and $\Gamma 3(A, B, P_2)$ respectively. Clearly O, O′ lie on the great circle through L and orthogonal to AB. If we apply the two sides, the included angle, and another angle formula (see article 48, [54]) to the spherical triangle ALO, we have

$$\cos A L \cos L\hat{A}O = \sin A L \cot A O$$

$$\therefore \tan AO = \frac{\tan\left(\frac{AB}{2}\right)}{\cos\theta_1} \tag{2}$$

since $L\hat{A}O = \theta_1$ (see article 122, [54])

Similarly, from spherical triangle ABP_2, we get

$$\tan AO' = \frac{\tan\left(\frac{AB}{2}\right)}{\cos\theta_2} \tag{3}$$

$\theta_1 > \theta_2$, and equations (2) and (3) imply that

$$AO > AO' . \tag{4}$$

From (4) we get

$$SR \text{ of } \Gamma 3(A, B, P_1) > SR \text{ of } \Gamma 3(A, B, P_2)$$

Now $OA > O'A$, $OL < \pi/2$, and OL orthogonal to AB imply $OL > OL'$. Therefore, O and P_2 lie on the same side of MO′ where M is the mid point of AP_2. Hence from Lemma 3.1.4 one can conclude that $OP_2 < AO$. This proves the last part of the Lemma 3.1.6. □

Theorem 3.1.3. *Assume that* O *is the nearer pole of* $\Gamma 3(A, B, C)$ *and angles of the spherical triangle* ABC *satisfy the conditions of Lemma* 3.1.3. *Let* P *be a point on the sphere and* P *not in* $\Sigma\Gamma 3(A, B, C)$ *such that* P *and* C *lie on the same side of the plane of the great circle* AO *and let* AP *be greater than* BP *and* CP, *and* $AP < \pi$.

If $A\hat{B}P < A\hat{P}B + B\hat{A}P$ *then*

$C \in \Sigma\Gamma 3(A, B, P)$ *and SR of* $\Gamma 3(A, B, P) > SR$ *of* $\Gamma 3(A, B, C)$

Else

$C \in \Sigma\Gamma 2(A, P)$ *and SR of* $\Gamma 2(A, P) > SR$ *of* $\Gamma 3(A, B, C)$

Proof. Let L and M be the mid points of AB and AC respectively. We have to consider the following two cases:

Case I $A\hat{B}P < A\hat{P}B + B\hat{A}P$

Let O′ be the nearer pole of $\Gamma 3(A, B, P)$ (see Fig. 4).

From Theorem 3.1.2, $BP < AP \Rightarrow B\hat{A}P < A\hat{B}P \Rightarrow B\hat{A}P < A\hat{B}P + A\hat{P}B$

Again P is outside $\Sigma\Gamma3(A, B, C)$ implies $OB = OA < OP$. From these relations and Theorem 3.1.2 we get

$$O\hat{P}B < O\hat{B}P \text{ and } O\hat{P}A < O\hat{A}P$$

From the above inequalities we obtain

$$A\hat{P}B < O\hat{B}P + O\hat{A}P < A\hat{B}P + B\hat{A}P$$

Consequently, angles of the spherical triangle ABP satisfy the conditions of Lemma 3.1.3. From Lemma 3.1.5, we get

$$SR \text{ of } \Gamma3(A, P, B) > SR \text{ of } \Gamma3(A, Q, B) = SR \text{ of } \Gamma3(A, B, C),$$

where Q is on AP such that $OQ = OA$. Hence O' lies on the extended portion of LO. Consequently, C and O' lie on the same side of the great circle through M orthogonal to AC. Lemma 3.1.4, therefore, implies that

$$AO' > CO', \text{ i.e., } C \in \Sigma\Gamma3(A, P, B).$$

Case II $A\hat{B}P \geq A\hat{P}B + B\hat{A}P$.

It follows from Lemma 3.1.1 that $B \in \Sigma\Gamma2(A, P)$. Consequently the nearer pole O' of $\Gamma2(A, P)$ is the mid point of AP (see Fig.5). Since $O'B < O'A$, from Lemma 3.1.4 we conclude that O' and B lie on the same side of OL. Let E denote the point of intersection of AP and LO produced. Since O' and B lie on the same side of LO, $AO' > AE$. Again E and A being on opposite sides of OM, O' and C lie on the same side of OM. From Lemma 3.1.4 we have

$$CO' < AO', \text{ i.e., } C \in \Sigma\Gamma2(A, P).$$

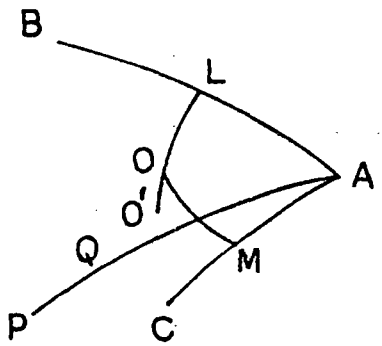


Fig. 4.

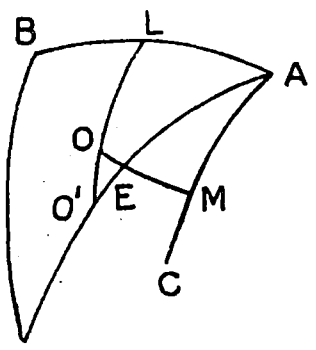


Fig. 5

Hence from Lemma 3.1.3 we obtain

SR of $\Gamma 3(A, B, C) <$ SR of $\Gamma 2(A, P)$.

3.2 Algorithm

We now proceed to describe the solution procedure of the problem. The set of demand points S is partitioned into two disjoint sets S_1 and S_2, where $S_1 = \varnothing$ and $S_2 = S$ initially. With each iteration S_1 absorbs at least one point which S_2 gives up. This continues until at last S_2 becomes void.

Algorithm Hemispherical Minimax Location Problem

Step 1. *Take any two points of S. Denote these points by* A, B.

Let $S_1 = S \cap \Sigma\Gamma 2(A, B)$ *and* $S_2 = S - S_1$ *and go to Step* 2.

Step 2. *If* $S_2 = \varnothing$, *stop; the nearer pole of* $\Gamma 2(A, B)$ *is the required facility point*

Else choose some point $C \in S_2$, *say, at a maximum distance (geodesic) from the nearer pole of* $\Gamma 2(A, B)$. *If* $AC < BC$ *then swap* A *with* B, B *with* C *and go to Step*3.

Step 3. *Let*

$\theta_k = max \{\theta_i : P_i \in S_1 - (A, B, C)$; P_i *and* C *lie on the same side of the plane of the great circle* AB$\}$

If $A\hat{P}_kB > A\hat{B}P_k + B\hat{A}P_k$ *then* $B \leftarrow P_k$, $S_1 = S \cap \Sigma\Gamma 2(A, B)$ *and go to Step* 2

Else $C \leftarrow P_k$, $S_1 = S \cap \Sigma\Gamma 3(A, B, C)$, $S_2 = S - S_1$ *and go to Step* 4.

Step 4. *If* $S_2 = \varnothing$, *stop. The nearer pole of* $\Sigma\Gamma 3(A, B, C)$ *is the required facility point.*

Otherwise choose $D \in S_2$, *and label the point among* A, B, C *that is farthest from* D *as* A. *Rename the other points* B *and* C. *Denote the nearer pole of* $\Gamma 3(A, B, C)$ *by* O.

If B *and* D *lie on the same side of* OA *then swap* B *with* C; $C \leftarrow D$ *and swap* B *with* C. *Repeat Step* 3.

Remarks

1. From Lemma 3.1.6 and Theorem 3.1.3 it follows that at each step of the algorithm, the spherical radius of the small circle which bounds the sphere containing the nearer pole, strictly increases. The algorithm needs the pole and the spherical radius of $\Gamma 2(A, B)$ or $\Gamma 3(A, B, C)$ at each iteration. Since $\Sigma\Gamma 2(A, B)$ and $\Sigma\Gamma 3(A, B, C)$ are defined uniquely, the same set of points cannot occur in any two iterations. Also the number of demand points is finite, and therefore the algorithm is finite.

2. Lemma 3.1.6 and Theorem 3.1.3 imply that if a demand point P belongs to S_1 at a particular iteration then $P \notin S_2$ in subsequent iterations. Consequently in no case the algorithm requires more than (n - 2) iterations. If there are k points to be considered in a particular iteration then the algorithm needs to compare at most k geodesic distances. Each distance can be calculated by using "three sides and an angle" formula (see article 41, [54]). Hence in the worst case we have to consider $O(n^2)$ distances to obtain the exact solution.

3. If conditions of Lemma 3 hold in a plane triangle then we have the following:

$$A < B + C \Rightarrow A < \pi/2.$$

The condition max(B + C - A) in Lemma 6 implies A being minimum. That is in step 3 we search for a minimum acute angle.

3.3 Numerical Example

Consider the following example given in Table 2, section 2.3. The reason for choosing this problem arises from the fact that all demand points are situated in the northern hemisphere. We have appended a comparative estimate of running time of algorithms [45] and [50] with that of the present one in Table 1. It would not be out of place to mention that the computer CPU time to run our program depends on the initial choice of the pair of demand points. The time .06 mentioned in Table 1 is the maximum computer CPU time corresponding to the initial choice of Athens and Stockholm.

Table 1: Comparison of execution times for different algorithms

Algorithm	Execution Time	Computer Type
Patel	Less than 5 sec.	Convex C220 main frame
Sarkar-Chaudhuri	0.16 sec.	PC AT 486 DX2 66 MHz
Present Algorithm	0.06 sec.	PC AT 486 DX2 66 MHz

For making a comparative estimate of the present algorithm and that of Sarkar-Chaudhuri [50] we have developed the PASCAL codes of both with Turbo Pascal (Borland) compiler 6.0.

3.4 Pascal Code of the Hemispherical Minimax Problem

In this section we are going to develop the Pascal code of the Algorithm given in section 3.2.

```
program dualfeasible(input,output,infile);
```

```
{This program uses the concept of dual feasibility to determine optimum solution of the
minimax location problem when the demand points are situated on a hemisphere}
uses crt,dos;
type position=array[1..1000] of real;
var
 infile:text;
 x,y:position;
{these two vectors contain the latitudes and longitudes of the demand points on a unit sphere}
 i,j,k,n,i1,i2,i3,i4:integer;
 u,v,w,z,u1,u2,u3,v1,v2,v3,w1,z1,s,s1,s2,s3,r:real;
 flag:boolean;
 hh,mm,ss,hs:word;
procedure cosine(var a:real;i,j:integer);
{this procedure obtains the cosine of a side of a spherical triangle when other two sides and
the opposite angle are known}
 begin
   a:=sin(y[i])*sin(y[j])+cos(y[i])*cos(y[j])*cos(x[i]-x[j])
 end; {end of the procedure cosine}
procedure distance(var uu:real;ii,jj:integer);  {this procedure finds cot(side/2)}
 begin
    cosine(uu,ii,jj);
    uu:=sqrt((1+uu)/(1-uu));
 end;{end of the procedure distance}
procedure distance1(var uu:real;ii,jj:integer);   {this procedure determines tan(side/2)}
  begin
     cosine(uu,ii,jj);
     uu:=sqrt((1-uu)/(1+uu));
  end;{end of the procedure distance1}
procedure spangle(var v11,v22,v33:real;u11,u22,u33:real);    {when three sides of a spherical
triangle are known this procedure determines the spherical angles}
  begin
     s1:=sin(u22+u33-u11);
```

```
    s2:=sin(u33+u11-u22);
    s3:=sin(u11+u22-u33);
    s:=sin(u11+u22+u33);
    v11:=2*arctan(sqrt(s2*s3/(s*s1)));
    v22:=2*arctan(sqrt(s3*s1/(s*s2)));
    v33:=2*arctan(sqrt(s1*s2/(s*s3)));
  end;{end of the procedure spangle}
procedure midpoint;
{this procedure finds latitude and longitude of the mid point of a side of a spherical triangle}
  begin
    distance(u,i1,i2);
    v:=cos(y[i1])*sin(y[i2])/cos(y[i2])-sin(y[i1])*cos(x[i1]-x[i2]);
    v:=v/abs(sin(x[i1]-x[i2]));
    w:=cos(y[i1])*sqrt(1+v*v)*u-sin(y[i1])*v;
    z:=(v+sin(y[i1])*w)/(sqrt(1+w*w)*cos(y[i1]));
    if w>0 then w:=arctan(1/w)
    else w:=4*arctan(1)-arctan(-1/w);
    if x[i2]>x[i1] then w:=x[i1]+w
    else w:=x[i1]-w;
    z:=arctan(z);
  end;{end of the procedure mid point}
procedure angle(var v1,v2,v3:real;ii,jj,kk:integer);
 {this procedure uses latitude and longitude to obtain sides of the spherical triangle then with
the help of the procedure spangle it finds spherical angles}
  begin
    distance1(u1,jj,kk);u1:=arctan(u1);
    distance1(u2,kk,ii);u2:=arctan(u2);
    distance1(u3,ii,jj);u3:=arctan(u3);
    spangle(v1,v2,v3,u1,u2,u3)
  end;{end of procedure angle}
procedure interchange(var ii,jj:integer);
  begin
```

```
    j:=ii;
    ii:=jj;
    jj:=j
  end;{end of the procedure interchange}
 procedure centre;
 {this procedure determines the latitude and longitude of the nearer pole of the small circle}
 var
  v11,v22,v33:real;
  begin
    midpoint;
    angle(v11,v22,v33,i1,i2,i3);
    u1:=(sin(y[i1])*sqrt(1+u*u)-u*sin(z))/cos(z);
    v11:=(v11+v22-v33)/2;
    v1:=sqrt(1+u*u)*cos(v11)/sin(v11);
    v2:=(cos(z)*v1-sin(z)*sqrt(1-u1*u1))/abs(u1);
    z1:=(sin(z)*v2+sqrt(1-u1*u1)/abs(u1))/(sqrt(1+v2*v2)*cos(z));
    z1:=arctan(z1);{z1 denotes the latitude of the nearer pole}
    if v2<0 then
      v2:=4*arctan(1)-arctan(-1/v2)
    else
      v2:=arctan(1/v2);
    if u1<0 then w1:=w-v2
    else w1:=w+v2;    {w1 represents longitude of the nearer pole}
    u:=u*cos(v11);
    u:=u/sqrt(1+u*u);
  end;{end of procedure centre}
 procedure step1;
 { this procedure finds if there is any optimum solution of the problem given by two points}
  begin
    midpoint; {gives w, z,  i.e., longitude and latitude}
    cosine(u,i1,i2);
    u:=sqrt((1+u)/2);
```

```
 i3:=0;
 for i:=1 to n do
  begin
   if (i<>i1) and (i<>i2) then
    begin
      u1:=sin(z)*sin(y[i])+cos(z)*cos(y[i])*cos(w-x[i]);
      if u1<u then      {u1<u implies point i is outside}
       begin
         u:=u1;
         i3:=i
       end
    end;
  end {end of for}  {aflage=true represents optimal solution}
end;{end of the procedure step1}
procedure step2; {this procedure examines whether the solution of the problem is obtained by
two or three points}
 begin
  cosine(u,i1,i3);
  cosine(v,i2,i3);
   if u>v then interchange(i1,i2);
   angle(v1,v2,v3,i1,i2,i3);
   if  v2>v1+v3 then   {v2>v1+v3 represents two point problem}
    begin
     i2:=i3;i3:=0
    end
 end;{end of the procedure step2}
procedure step3;{this procedure finds whether there is a point outside the spherical disc}
 begin
 i4:=0;
 if y[i1]>y[i3] then interchange(i1,i3);
 if y[i2]>y[i3] then interchange(i2,i3);
 if x[i1]>x[i2] then interchange(i1,i2);
```

```
centre;
for i:=1 to n  do
 begin
  if (i<>i1) and (i<>i2) and (i<>i3) then
   begin
    u3:=sin(z1)*sin(y[i])+cos(z1)*cos(y[i])*cos(w1-x[i]);
    if u3<u then {u3<u implies that the point is outside the spherical disc}
      begin
        u:=u3;
        i4:=i
      end
   end;
 end; {end of for}
if i4<>0 then    {i4=0 implies optimal solution}
 begin
   cosine(u1,i1,i4);
   cosine(u2,i2,i4);
   cosine(u3,i3,i4);
   if u1>u2 then interchange(i1,i2);
   if u1>u3 then interchange(i1,i3);
   if u2>u3 then interchange(i2,i3);
   angle(u1,u2,u3,i1,i2,i4);
   angle(v1,v2,v3,i1,i3,i4);
   s1:=u3+u1-u2;
   s2:=v3+v1-v2;
   if (s1<0) and (s2<0) then
{from this condition we get optimum solution of the problem by two demand points}
     begin
       i2:=i4;i3:=0
     end
   else
```

```
{if this condition is true then we get the optimum solution of the problem from three demand
points}
      begin
       if s1<0 then i2:=i4
       else
         if s2<0 then i3:=i4
         else
           begin
             if s1>s2 then i2:=i4
             else i3:=i4
           end
      end
   end
 end;{end of step 3}
begin{main action block}
 clrscr;
 assign(infile,'file.pkc'); {this file contains latitude and longitude of the demand points}
 reset(infile);
 writeln('supply the value of n, i.e., no. points');
 readln(n);
{from the infile the vectors x[i], y[i] read the latitudes and longitudes of the demand points
expressed in degrees}
 for i:=1 to n do
    readln(infile,x[i],y[i]); {from the following transformation law one gets the latitudes and
longitudes in terms of radians}
 r:=arctan(1)/45;
 for i:= 1 to n do
  begin
    x[i]:=x[i]*r; y[i]:=y[i]*r
  end;
 gettime(hh,mm,ss,hs);
 writeln(hh,':',mm,':',ss,'.',hs);
```

```
flag:=true;i1:=7;i2:=8;k:=0;i3:=0;r:=1/r;
while flag do
 begin
  if i3=0 then
   begin
    step1;
    if i3=0 then
     begin
       flag:=false;
       writeln('the lat. and long. are ',z*r,' and ',w*r)
     end
   end;
 if i3<>0 then
  begin
   step2;
   if i3<>0 then
    begin
      step3;
      if i4=0 then
      begin
        flag:=false;
        writeln('the lat. and long. are ',z1*r,' and ',w1*r)
      end
    end
  end;
 k:=k+1;
end;{end of while}
writeln('no. of iteration= ',k);
gettime(hh,mm,ss,hs);
writeln(hh,':',mm,':',ss,'.',hs);
close(infile)
end.
```

Polynomial Time Algorithm for a Hemispherical Minimax Location Problem

4.1 Background

In this section we are going to develop an efficient algorithm to solve the problem (2), for the hemispherical minimax location problem, mentioned in section 1.1. We assume that all demand points lie on a hemisphere. We first mention the strategy of the Sarkar-Chaudhuri Algorithm [22]. Let I = {1, 2,...,n}. Then Sarkar-Chaudhuri algorithm may be described as follows:

Initial Step. Choose any point P on the surface of the hemisphere which contains all the demand points and let A_k, $k \in I$, be the farthest demand point from P. Denote this farthest point by A. Let $I \leftarrow I - \{k\}$. Connect A with P by a great circle arc. Let P_1 be a point on the arc PA such that

$$\text{arc } P_1A = \text{arc } P_1A_i ,$$

where $i \in I$ and arc PP_1 is minimal. Let the demand point satisfying the above be denoted by B and $I \leftarrow I - \{i\}$. Go to Step 1.

Step 1. If all the demand points lie on $\Sigma\Gamma 2(A, B)$ then stop; P*, the nearer pole of $\Gamma 2(A, B)$ is the required facility point. Else $P \leftarrow P_1$, and go to Step 2.

Step 2. Join P and the middle point, D, of AB by the arc of a great circle. Find a point P_1 on arc PD such that arc P_1A = arc P_1A_k, $k \in I$ and arc P_1P is minimum. Denote this point by C.

If $\hat{A} < \hat{B} + \hat{C}$, $\hat{B} < \hat{C} + \hat{A}$ and $\hat{C} < \hat{A} + \hat{B}$

then stop; $P^* = P_1$ is the required facility point.

Else go to Step 3.

Step 3. If $A_k\hat{A}_iA_j > A_i\hat{A}_jA_k + A_j\hat{A}_kA_i$ where $A_i, A_j, A_k \in \{A, B, C\}$ and i, j, k are all different then drop A_i. Denote the point A_j and A_k by A and B respectively. Let $P \leftarrow P_1$, $I \leftarrow I - \{i\}$ and repeat Step 1.

The optimality criteria of the hemispherical minimax location problem have been given in Steps 1 and 2. Let us see the geometrical significance of these two.

Assume that P* is the midpoint of the arc AB and if all demand points lie on $\Sigma\Gamma 2(A, B)$ then Step 1 states that P* is the required facility point. For optimality, arc AP* = arc BP*. The result follows from Lemma 4.1.1.

Lemma 4.1.1. *Let* $P \in \Sigma\Gamma 2(A, B)$. *Then either arc* AP *or arc* BP *is greater than or equal to arc* AP*.

Proof. Join AP, BP, and AB by great circle arcs (see Figure 1). From Proposition 1.2.2, we have from the spherical triangle ABP

$$\text{arc AP} + \text{arc BP} \geq \text{arc AB} = 2 \text{ arc P*A}$$

The above inequality implies either arc AP or arc BP is greater than or equal to arc AP*. □

Step 2 states that if all demand points lie on $\Sigma\Gamma 3(A, B, C)$ then P*, the nearer pole of $\Gamma 3(A, B, C)$, is the required facility point provided

$$\hat{A} < \hat{B} + \hat{C}, \hat{B} < \hat{C} + \hat{A} \text{ and } \hat{C} < \hat{A} + \hat{B}$$

The following lemma explains the significance of the stopping criteria given in Step 2 of the Sarkar-Chaudhuri algorithm [22].

Lemma 4.1.2. *In a spherical triangle* ABC

$$\hat{A} < \hat{B} + \hat{C},\ \hat{B} < \hat{C} + \hat{A} \text{ and } \hat{C} < \hat{A} + \hat{B}$$

imply Δ ABC *is an acute triangle.*

Proof. Let P be the nearer pole of $\Gamma 3(A, B, C)$ and let O be the centre of $\Gamma 3(A, B, C)$. Assume that the great circle arcs AP, BP and CP intersect $\Gamma 3$ (A, B, C) at A_1, B_1 and C_1 respectively. It follows from Lemma 3 and Corollary 2 (see, [22]) that B and C lie on opposite sides of AA_1, C and A lie on opposite sides of BB_1, and A and B lie on opposite sides of CC_1 (see Figure 2). Therefore, ΔABC is an acute triangle. □

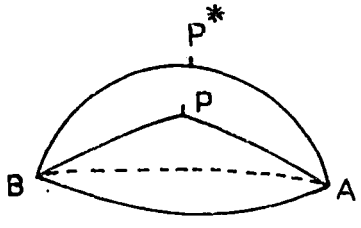


Figure-1

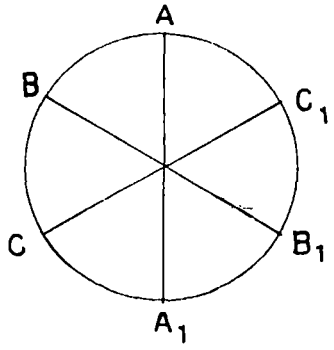


Figure - 2

It is clear form Lemmas 4.1.1 and 4.1.2 that the optimal solution of the hemispherical minimax location problem is the nearer pole P* of $\Gamma 2(A, B)$ [$\Gamma 3(A, B, C)$] provided $\Sigma\Gamma 2(A, B)$ [$\Sigma\Gamma 3(A, B, C)$] contains all demand points. Thus a hemispherical minimax problem reduces to finding a small circle of a maximum radius on the surface of the sphere which contains either two demand points at the ends of a diameter or three demand points forming an acute triangle such that all demand points lie on one side of the plane of the small circle and the centre of the sphere on the other side.

Our algorithm is based on the following lemma:

Lemma 4.1.3. *Let* A *and* B *be two points on the surface of the sphere* Σ *such that* A *and* B *are not the ends of a diameter of the sphere. Let* $Q \in \Sigma$ *such that* $Q \notin \Sigma\Gamma 2(A, B)$. *Then* $<AQB$ *is an acute angle.*

Proof. Construct the sphere S(A, B) with $\Gamma 2(A, B)$ as a great circle. Clearly all points of $\Sigma\Gamma 2(A, B) - \Gamma 2(A, B)$ lie within S(A, B) and all points of $\Sigma - \Sigma\Gamma 2(A, B)$ lie outside S(A, B). Now Q being a point outside S(A, B) it is obvious that $<AQB$ is less than a right angle. □

Corollary 4.1.1 *Let* A, B, *and* C *be three points on the surface of the sphere* Σ *such that* ΔABC *is an acute triangle. Further assume that* $Q \in \Sigma$ *and* $Q \notin \Sigma\Gamma 3(A, B, C)$ *where* $\Sigma\Gamma 3(A,B,C)$ *is a small circle. Then* $OQ > OA = OB$, *where* O *is the centre of* $\Gamma 3(A, B, C)$.

Proof. The result of the corollary follows immediately from the proof of the Lemma 4.1.3. □

Using Lemma 4.1.3 and the corollary 4.1.1 we have the following algorithm for the hemispherical minimax location problem given in section 4.2.

4.2 Algorithm Hemispherical Minimax Location

Let the set S, of demand points, lie on the surface of the sphere Σ. We assume that these demand points lie on a hemisphere. Then our algorithm may be described as follows:

Initial Step *Take any two demand points, say* A *and* B, *and go to Step* 1.

Step 1 *If all demand points lie on* $\Sigma\Gamma 2(A, B)$, *stop. The nearer pole,* P*, *of* $\Gamma 2(A, B)$ *is the required facility point.*

Else chose a demand point, C *say, not in* $\Sigma\Gamma 2(A, B)$ *such that* $<ACB$ *is a minimum. Go to Step* 2.

Step 2 *If all demand points lie on* $\Sigma\Gamma 3(A, B, C)$ *and* ΔABC *is an acute triangle then stop. The nearer pole* P* *of* $\Gamma 3(A, B, C)$ *is the required facility point.*

Else go to Step 3.

Step 3 *If* ΔABC *is an obtuse triangle then drop the point with the obtuse angle, rename the remaining points* A *and* B, *and go to Step* 1

Else find a demand point, D *say, in* $\Sigma - \Sigma\Gamma 3(A, B, C)$ *such that the distance of* D *from the centre of* $\Gamma 3(A, B, C)$ *is maximum and go to Step* 4.

Step 4 *Find the maximum distance of* D *from* A, B *and* C. *Denote the point having maximum distance from* D *by* A *and rename the other two points as* B *and* C. *Find minimum* {<ABD, <ACD}.

If this minimum is greater than $\frac{\pi}{2}$ *then* B ← D *and go to Step* 1.

Else drop the point with the maximum angle. Denote the other point by B. *Let* C ← D *and go to Step* 2.

Remarks

1. The optimality conditions, of the present algorithm, are given in Steps 1 and 2. To obtain optimality conditions we may proceed as follows: Let

$f(x, y, z) = lx + my + nz - p = 0$, where

$l^2 + m^2 + n^2 = 1$ and $p > 0$,

be either the plane $\Sigma\Gamma 2(A, B)$ or the plane $\Sigma\Gamma 3(A, B, C)$. If for all demand points (x, y, z), $f(x, y, z) \geq 0$ then the coordinates of the facility point P* are (l, m, n).

2. The point D(x, y, z) mentioned in Step3 of the algorithm has the following characteristic properties;

(i) $f(x, y, z) < 0$

(ii) f(x, y, z) is a minimum.

4.3 Numerical Example

In this section we are going to compare the efficiency of the algorithm, which has been presented in section 4.2, with the existing algorithms. We now consider the following problem given in Table 2, section2.4, and develop the Turbo PASCAL (Borland) code of the algorithm. We have used a PC AT 486, DX2 66 MHz to compare the CPU time of the present method and the existing algorithms. The results are given in Table 1. The Cartesian coordinates of the facility point are (0.1191, 0.6435, 0.7561) and the optimum latitude and longitude of the facility point are 49.12° N and 79.51^0E respectively. The nearer pole of $\Gamma 2(A, B)$ is the required facility point

where A and B denote the demand points Paris and Manila respectively.

Table 1: CPU time for different algorithms

Algorithm	CPU Time	Computer
Patel	Less than 5 secs.	Convex C220 mainframe
Sarkar-Chaudhuri	0.16 sec.	PC AT DX2 66 Mhz
Method of section 2.2	0.06 sec.	PC AT DX2 66 Mhz
Present Algorithm	0.00 sec.	PC AT DX2 66 MHz

We have considered 5 sets containing 10, 20, 50, 80, and 100 data points distributed at random over a unit sphere. Each of the above set was randomly generated twenty five times. Table 2 shows the computation time (in seconds) of the present algorithm and that of the algorithm given in section 2.2. We have used Turbo PASCAL (Borland) Version 6 to run both the programs in a PC AT 486, DX2 66 MHz computer.

Table 2:

No. of Points	CPU time of the present algorithm	CPU time of the Algorithm given in Section 2.2
10	0.00	0.00
20	0.00	0.11
50	0.00	2.00
80	0.01	7.24
100	0.03	15.05

Table 3:

No. of points	CPU time of the present algorithm
500	0.11
1000	0.23
1500	0.36
2000	0.48
2500	0.61
3000	0.73

We next consider 500 to 3000 data points at an interval of 500. Table 3 above shows the corresponding CPU time (in seconds) to run the program in the same computer and using the same source code.

Conclusion.

The algorithm presented here is significantly faster than all the existing algorithms. From the computational point of view the Sarkar-Chaudhuri Algorithm [22] takes much more time than the present algorithm, despite the former being $O(n^2)$. This is so because the Sarker-Chauduri Algorithm involves inter alia computation of trigonometric functions whereas the present method uses only Euclidean distances.

In each step, the algorithm requires the equation of a plane through three demand points, forming an acute triangle, or through two demand points, which are the ends of the diameter of a small circle. Then we have to verify whether all other demand points lie on the same side of this plane. If this is true then we obtain the optimum solution of the problem which is the nearer pole of the small circle determined by the section of the sphere by this plane. Otherwise we have to determine a new set of points and the plane through it and we have to continue this process until we obtain the optimum solution. It is to be noted that the maximum number of arithmetic operations necessary to get the exact optimum solution is $O(n^2)$.

4.4 Pascal Code of the algorithm

In this section we implement the Pascal code of the algorithm developed in section 2.2.

```
program hemisphere(input,output,infile);
{ This program finds the optimum solution of a hemispherical minimax location problem }
 uses crt,dos;
 type
   list=array[1..3500] of real;
 var
   infile:text;
   x,y,z:list;  {these three vectors contain the Cartesian coordinates of the demand points
    situated on a unit sphere}
   i,j,i1,i2,i3,i4,n:integer;
   a,b,c,u,v,d:real;
```

```
flag:boolean;
    hh,mm,ss,hs:word;
 procedure distance(var d1:real;j1,j2:integer);
{This procedure obtains square of the distance between two demand points}
  begin
   d1:=sqr(x[j1]-x[j2])+sqr(y[j1]-y[j2])+sqr(z[j1]-z[j2])
  end; {end of distance}
 procedure swap(var d1,d2:real;var j1,j2:integer);
  {This procedure interchanges the memory location of two quantities and the corresponding
indices}
 var
    u1:real;
    k:integer;
  begin
     u1:=d1;d1:=d2;d2:=u1;
     k:=j1;j1:=j2;j2:=k
  end;{end of swap}
 procedure radius(var r:real;d1,d2,d3:real);
{This procedure finds the square of the diameter of the circle containing three demand points}
   var
    u1:real;
   begin
    u1:=4*d2*d3;
    r:=(d2+d3-d1);
    r:=u1*d1/(u1-r*r);
   end;{end of radius}
 procedure update2;
{This procedure determines which two demand points to be considered for the next iteration }
  var
    u1,u2,u3:real;
   begin
    distance(u1,i2,i3);
```

```
    distance(u2,i3,i1);
    distance(u3,i1,i2);
    if (u1<u2) then swap(u1,u2,i1,i2);{interchange the points i1,i2}
    u1:=u1-u2-u3;
    if u1>=0 then {drop the point i1}
      begin
       i1:=i2;i2:=i3;i3:=0
      end
  end;{end of update2}
procedure update3;
{This procedure obtains three demand points to be considered for the next iteration}
  var
    d1,d2,u1,u2,u3,v2,v3:real;
  begin
    distance(u1,i1,i4);
    distance(u2,i2,i4);
    distance(u3,i3,i4);
    if (u1<u2) then swap(u1,u2,i1,i2);
    if (u1<u3) then swap(u1,u3,i1,i3);
    distance(v2,i1,i2);
    distance(v3,i1,i3);
    d1:=u1-u2-v2;
    d2:=u1-u3-v3;
    if (d1>=0) and (d2>=0) then {drop points i2,i3}
      begin
        i2:=i4;i3:=0
      end
    else if (d1<0) and (d2>=0) then  {drop the point i3}
      begin
        i3:=i2;i2:=i4
      end
    else if (d1>=0) and (d2<0) then i2:=i4 {drop the point i2}
```

```
    else
     begin
      radius(d1,u1,u2,v2);
      radius(d2,u1,u3,v3);
      if (d1>d2) then i3:=i4 {drop the point i3}
      else i2:=i4    {drop the point i2}
     end;
    i4:=0
   end;{end of update 3}
  procedure plane;
{ This procedure obtains the direction ratios of the  normal to the plane containing three demand points}
   begin
    a:=(y[i2]-y[i1])*(z[i3]-z[i1])-(y[i3]-y[i1])*(z[i2]-z[i1]);
    b:=(z[i2]-z[i1])*(x[i3]-x[i1])-(z[i3]-z[i1])*(x[i2]-x[i1]);
    c:=(x[i2]-x[i1])*(y[i3]-y[i1])-(x[i3]-x[i1])*(y[i2]-y[i1]);
    d:=a*x[i1]+b*y[i1]+c*z[i1]
   end;{end of plane}
  procedure plane1;
{ This procedure obtains the direction ratios of the normal to the plane containing two demand points}
   begin
    a:=x[i1]+x[i2];b:=y[i1]+y[i2];c:=z[i1]+z[i2];
    d:=a*x[i1]+b*y[i1]+c*z[i1]
   end; {end of procedure plane1}
  procedure optimum;
{ This procedure determines Cartesian coordinates of  the required facility point}
   begin
    if i3=0 then
     begin
      a:=x[i1]+x[i2];b:=y[i1]+y[i2];c:=z[i1]+z[i2];
      u:=sqrt(a*a+b*b+c*c);
```

```
      a:=a/u;b:=b/u;c:=c/u;
      writeln('Solution is obtained by the demand points',i1,', ',i2)
     end
    else
     begin
      plane;
      u:=sqrt(a*a+b*b+c*c);
      if d>0 then
       begin
        a:=a/u;b:=b/u;c:=c/u
       end
      else
       begin
        a:=-a/u;b:=-b/u;c:=-c/u
       end;
      writeln('Solution is obtained by the demand points ',i1,' ',i2,' ',i3);
     end
  end;{end of optimum}
procedure latitude; {This procedure finds the latitude of the facility point}
  begin
   i:=1;
   if c<0 then i:=-1;
   u:=d*arctan(abs(c)/sqrt(a*a+b*b));
  end;{end of latitude}
procedure longitude; {This procedure determines the longitude of the facility point}
  begin
   j:=1;
   if (a<>0) then v:=d*arctan(abs(b/a));
   if (a>0) and (b<0) then j:=-1;
   if (a<0) and (b<0) then
    begin
     j:=-1;v:=90+v
```

```
    end;
   if (a<0) and (b>0) then v:=90+v
  end;{end of longitude}
procedure twopoint;
  var
   u1,u2:real;
  begin
   v:=0;u:=0;
   plane1;
   u2:=sqrt(a*a+b*b+c*c);
   u2:=1/u2;
   for i:=1 to n do
    if ((i<>i1) and (i<>i2)) then
     begin
      u1:=d-a*x[i]-b*y[i]-c*z[i];
      if (d*u1>0) then
       begin
        v:=abs(u1*u2);
        if v>u then
         begin
          u:=v;i3:=i
         end
       end;
     end; {end of loop}
   if i3=0 then flag:=false
   else  update2;
  end;{end of twopoint}
procedure threepoint;
 var
   u1,u2:real;
  begin
    v:=0;u:=0;
```

```
    plane;
    u2:=sqrt(a*a+b*b+c*c);
    for i:=1 to n do
     if ((i<>i1) and (i<>i2) and (i<>i3)) then
      begin
       u1:=d-a*x[i]-b*y[i]-c*z[i];
       if (d*u1>0) then
        begin
         v:=abs(u1/u2);
         if v>u then
          begin
           u:=v;i4:=i
          end
        end;
      end; {end of loop}
   if i4=0 then flag:=false
   else  update3;
  end;{end of threepoint}
begin {main action block}
  clrscr;
  assign(infile,'file1.hem');{file1.hem contains coordinates of the data points}
  reset(infile);
  writeln('supply the number of demand points');
  readln(n);
  for i:=1 to n do
    readln(infile,x[i],y[i],z[i]);
  gettime(hh,mm,ss,hs);
  writeln(hh,':',mm,':',ss,':',hs);
  i3:=0;i4:=0;i1:=1;i2:=2;
  flag:=true;
  while flag do
    begin
```

```
      if (i3=0) then twopoint;
      if ((flag=true) and (i3<>0)) then threepoint;
    end;
  optimum;
  writeln('The Cartesian coordinates of the optimum point are');
  writeln(' (',a:2:4,', ',b:2:4,', ',c:2:4,')');
  d:=45/arctan(1);
  latitude;
  longitude;
  write('The latitude and longitude of the facility point are');
  if i=-1 then write('(',u:4:2,' S, ')
  else write('(',u:4:2,' N, ');
  if j=-1 then writeln(v:4:2,' W)')
  else writeln(v:4:2,' E)');
  gettime(hh,mm,ss,hs);
  writeln(hh,':',mm,':',ss,'.',hs);
  v:=sqr(x[i1]-a)+sqr(y[i1]-b)+sqr(z[i1]-c);
  u:=sqr(x[i2]-a)+sqr(y[i2]-b)+sqr(z[i2]-c);
  if u>v then v:=u;
  u:=sqr(x[i3]-a)+sqr(y[i3]-b)+sqr(z[i3]-c);
  if u>v then v:=u;
  writeln(v);
  for i:= 1 to n do
   begin
     if ((i<>i1) and (i<>i2) and (i<>i3)) then
       begin
          u:=sqr(x[i]-a)+sqr(y[i]-b)+sqr(z[i]-c);
          if u>v then writeln('u, i= ',u,'  ',i);
       end
    end;
  close(infile)
end. {end of action block}
```