Available online at www.sciencedirect.com
SCIENCE DIRECT®
Mathematical and Computer Modelling 39 (2004) 641–647
MATHEMATICAL AND COMPUTER MODELLING
www.elsevier.com/locate/mcm

An Algorithm for Finding the Shortest Sailing Distance from Any Maritime Navigable Point to a Designated Port

E. R. BEEKER
The MITRE Corporation
7515 Colshire Drive
McClean, VA 22102, U.S.A.
ebeeker@mitre.org

Abstract—Many simulations and deployment analyses use networks to evaluate ship paths at sea. Ships are constrained to travel along links between specified nodes. When ships may need to change their destination while at sea, the network path can deviate significantly from an actual shortest path. Additionally, when it is desired to initialize simulation data from actual ship positions, ships must move to a node to use the network. By triangulating open navigable water areas, a shortest path from all points at sea to selected ports can be calculated. A Delaunay triangulation of the open areas can be calculated in $O(n \log n)$. Using the Delaunay triangulation, the shortest path triangulation from a point (port) can be calculated in $O(nk)$ where n is the number of points specifying the sea-land border and k is the number of islands. The resulting data can provide distance from any point on the sea surface to the port in $O(1)$ time when the triangle containing the point is known, or in $O(n/2)$ if the triangle must be determined. © 2004 Elsevier Ltd. All rights reserved.

Keywords—Deployment, Strategic sealift, Delaunay triangulation on a sphere, Shortest path.

1. INTRODUCTION

For many years, the primary method for representing ship movement in computer models has been taken from the SAIL model. The SAIL model consists of a set of nodes, arcs, and lines. Ship routes are represented as arcs between nodes. The Maritime domain (oceans, seas, canals, rivers, straights) are represented as a network. In the SAIL network, there is a node every 15° of latitude and longitude. There are additional nodes to represent critical locations, such as the Straights of Gibraltar, the Panama Canal, and Horn of Africa. Although the arcs between the nodes are great circle routes, and therefore, shortest paths on the surface of the Earth, the approximations using the nodes are not. Long routes are broken into smaller segments, and while each segment is a shortest path, the overall route deviates significantly from a shortest path. Figure 1 shows a portion of the SAIL network (solid lines) between New York and Gibraltar along with the actual shortest path (dashed line). The path selected on the SAIL network would be along the bottom solid lines. When used in a simulation, the path of a ship will deviate from the shortest path and

0895-7177/04/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2004.02.004
Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

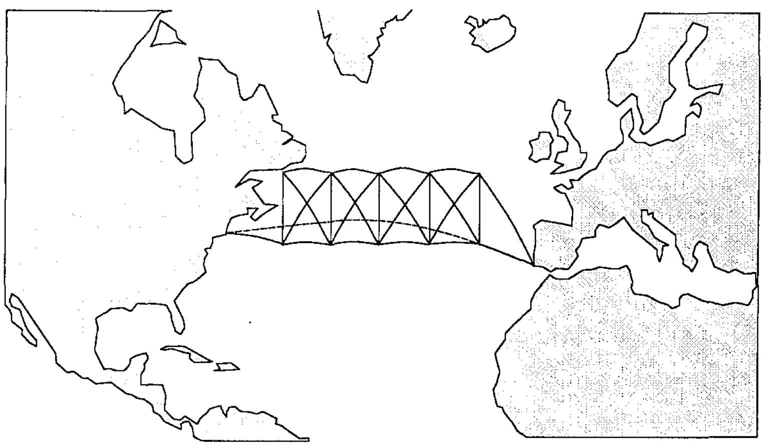

Figure 1.

the distance calculation will be in error. To overcome this, the SAIL network adds long routes across the oceans, for example from the Panama Canal to the Straights of Gibraltar. These added arcs bypass the oceanic nodes to yield a true shortest path. The SAIL set of lines is used to represent land. The model is initialized by connecting ports to the network and placing ships on the network. Ports are connected by finding the nearest node to the port's position that is not separated by a line and adding a link from the port to the node. Ships are placed by finding the closest node to each ship position that is not separated by a line and having the ship sail to the node. Because ships initially travel to the closest node, they may not be traveling towards their next port. After a beginning period, ships are confined to the network and so no more anomalies enter the ship simulation. However, the restriction of ships to the network limits the flexibility of the simulation. If a simulation wanted to change the destination of a ship at sea, perhaps because a canal was closed, the ship would have to continue to the next node before changing course. If the ship were on one of the long arcs, this could take it thousands of miles out of its way.

The area based ship model solves these problems by dividing the ocean into areas such that the next point on the shortest path to a port is the same for all the points in the area. This is illustrated in Figure 2, showing the Mediterranean Sea. A ship anywhere in the vertically shaded area can travel directly to Naples. Figure 3 shows two areas for Gibraltar. Any ship in the vertically shaded area can travel directly to Gibraltar. A ship anywhere in the horizontally shaded area must first travel to Point A and then to Gibraltar. For every possible destination, the area-based ship model divides the navigable waterways into a set of regions. Each region is a regular polygon and it is represented as a set of triangles for computational purposes. All of

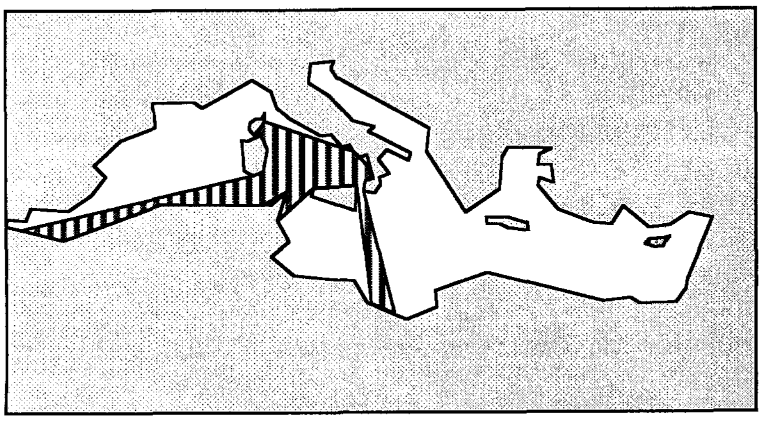

Figure 2.

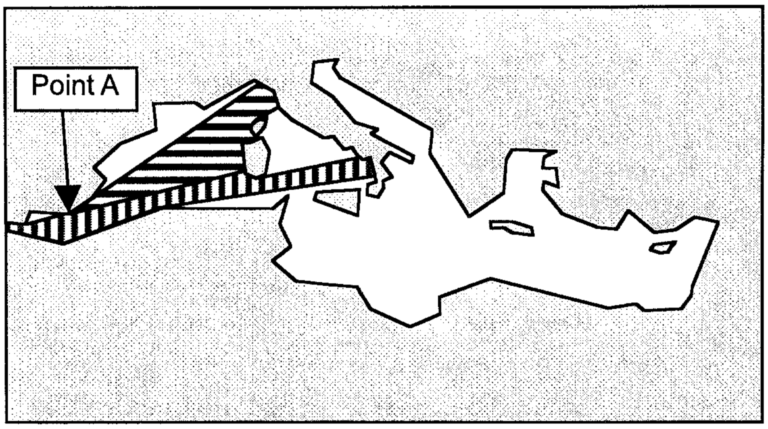

Figure 3.

the triangles that form a region have the same next point on the shortest path to a destination. By determining the triangle containing a ship's position, the shortest path to a destination is determined. No matter where the ship is, it can find the shortest path from its current position to a destination. The ship never backtracks or continues to a node that is not on the great circle shortest path to its destination. The triangulation is combined with a network algorithm for line-oriented routes, such as the Saint Lawrence Seaway, and the Panama and Suez Canals to provide complete coverage of sea routes. The following sections describe the procedures used to build the triangulation.

2. DELAUNAY TRIANGULATION

The shortest path algorithm depends upon a Delaunay triangulation. A set of points can be triangulated in a number of different ways. A Delaunay triangulation has several features that make it desirable. Delaunay triangles connect nearest neighbors. Also, a Delaunay triangulation makes the minimum angle of a triangle, as large as possible. In a sense, the triangles are as fat as possible so that the ratio of length of sides to area is as small as possible. A Delaunay triangulation can be considered a normalized triangulation for a set of points and is unique with an exception to be described.

The test for a Delauney triangulation uses the circumscribing circle defined by the three vertices of a triangle. In a Delaunay triangulation, no other points lie within the circle. Four points can be triangulated in two ways. Figure 4 shows four points in a non-Delaunay triangulation, since Point D lies within the circle defined by Points ABC. The triangulation can be turned into a Delaunay triangulation by "flipping" the shared side (AC) to the other option (BD). Figure 5 shows the same points in a valid Delaunay triangulation. If four points form a square, then two triangulations exist using either of the diagonals of the square, as shown in Figure 6. With this exception, a Delaunay triangulation is unique for a set of points. Some authors call a set of points that contain squares, degenerate, and will move one of the points to eliminate the condition. In this application the diagonal can be chosen arbitrarily.

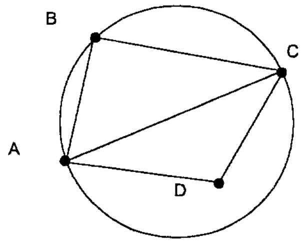

Figure 4.

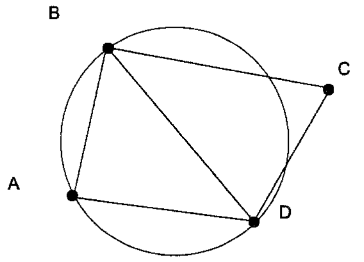

Figure 5.

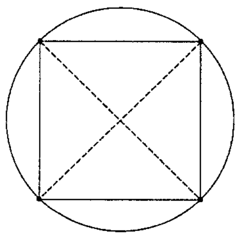

Figure 6.

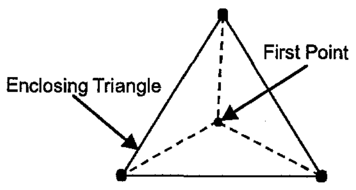

Figure 7.

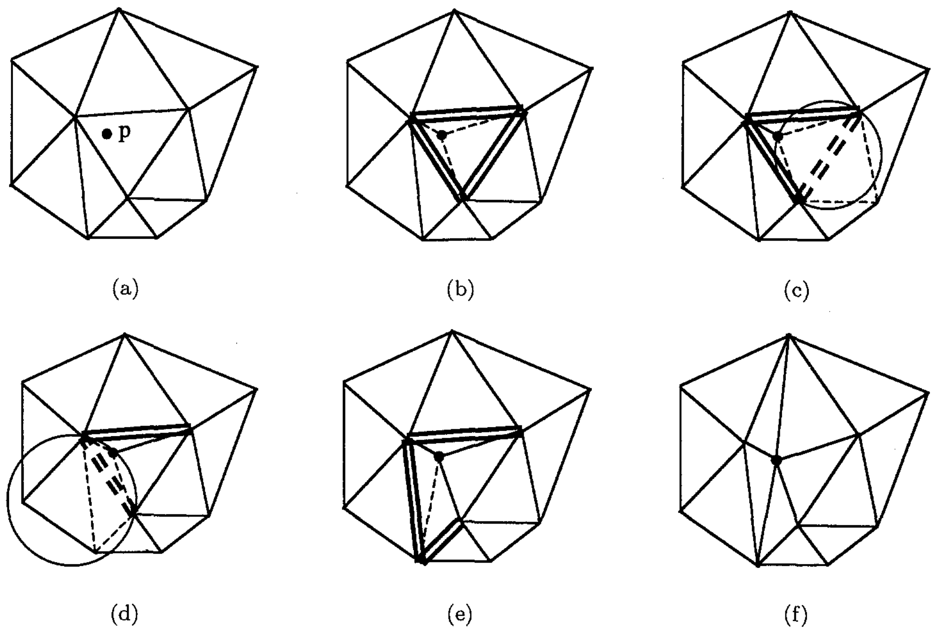

Figure 8.

This application uses an insertion algorithm to form the Delaunay triangulation, adapted from Shewchuck's triangle program [1]. In the plane, first an enclosing triangle is created that is guaranteed to contain all of the points in the input set. Then, each point is added, one at a time. Figure 7 shows the enclosing triangle and the first point. When a point is added, it is attached to the vertices of the enclosing triangle. Then each of the newly formed triangles is tested for the Delaunay condition. The triangulation is adjusted to maintain the Delaunay condition. After all the points are added, the enclosing triangle can be deleted, leaving the input point set in a valid Delaunay triangulation.

The process of adding a point and adjusting the triangulation is shown in Figure 8. In Figure 8a, the existing triangulation meets the Delaunay condition when Point P is added. Point P is connected to the vertices of the triangle that contains P as shown in Figure 8b. The three new triangles whose bases are the edges of the enclosing triangle and have the Point P as apex must be tested for the Delaunay condition. The bases, shown with a double line in Figure 8b, are added to a list of sides to be checked. Figure 8c shows the test of the first side, which is successful. The second test, shown in Figure 8d is unsuccessful. Therefore, the base of the test triangle is an incorrect edge. It is flipped by replacing it with the diagonal that crosses it as shown in Figure 8e. When it is flipped, the attached two sides of the triangles just formed are added to the list for testing. The algorithm continues testing and flipping diagonals until all of the marked edges are tested with no more changes. In practice, the swaps die out quickly. Figure 8f shows the final triangulation after adding Point P.

3. PARTITIONING THE POLYGON

The Delaunay triangulation is used for traversing the nodes of the polygon. The following description is adapted from Storer [2]. Figure 9 shows a polygon without obstacles. The al-

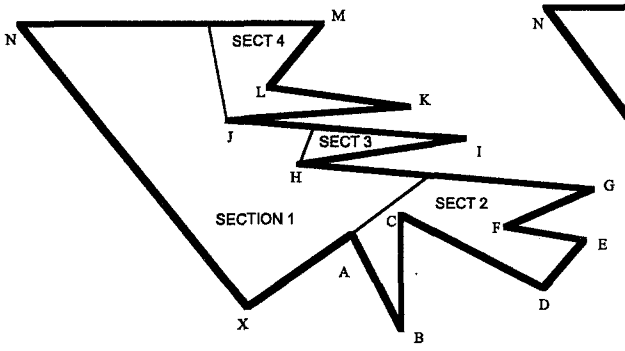

Figure 9.

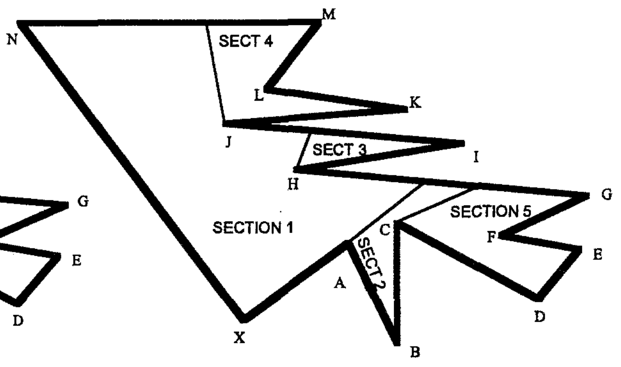

Figure 10.

gorithm begins from a source Point X and partitions the polygon into the section that can be directly reached from the source point, and other sections that must go through an intermediate point. The other sections will be partitioned recursively until there are no more sections to be partitioned. The basic algorithm begins by examining the first point counter-clockwise along the boundary of the polygon (Point A). As the angle at A is more than 180°, the X-A edge is extended until it strikes a boundary edge, dividing the polygon into two sections. The right section (Section 2) will be saved for partitioning from Point A. Partitioning of the left section continues until the polygon has been partitioned into four pieces. All of Section 1 has a direct path to X. Any point in Section 2 must go through Point A to reach Point X. Any point in Section 3 must go through Point H and any point in Section 4 must go through Point J. After partitioning from Point X, Section 2 can be further partitioned using Point A as the origin. This creates Section 5, as shown in Figure 10. The shortest path from Section 5 must go through Point C then Point A.

A similar approach can be taken for obstacles, with the proviso that an area behind an obstacle may be reached more than one way, and the first way may not be the shortest. If not, then the triangulation must be changed. If a triangle behind an obstacle can be reached from two directions, then it should be split. The boundary generally is a series of quadratic curves, as seen in Figure 11. In the ship sailing model, the quadratic curves are replaced by straight edges, as the error is negligible for this use. Another alternative would be to keep the next point for both triangles in each triangle. Then a simple test could determine whether it would be shorter to go to one or the other next point.

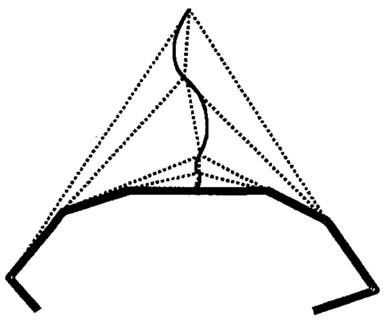

Figure 11.

4. PRIMITIVES

There are two basic primitives used to create the Delaunay triangulation: the test for counter-clockwise and the test for in-circle. In a plane, the test for counterclockwise is straight-forward. Given two vectors <u>**AB**</u> and <u>**AC**</u>, the cross product of the two vectors is positive when <u>**AC**</u> is counterclockwise to <u>**AB**</u> and negative if <u>**AC**</u> is clockwise to <u>**AB**</u>. If <u>**AB**</u> and <u>**AC**</u> are collinear,

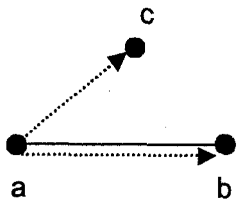

Figure 12.

$$\begin{vmatrix} b_x - a_x & b_y - a_y \\ c_x - a_x & c_y - a_y \end{vmatrix}$$

Figure 13.

then the cross product is zero. The cross product can be evaluated by finding the determinant of the matrix in Figure 13.

In three-space, the vector product of three vectors gives the volume of a parallelepiped and the sign shows their orientation as seen from the vector origination point as shown in Figure 14. Using Earth centric coordinates, the vector product can be calculated by the determinant as shown in Figure 15.

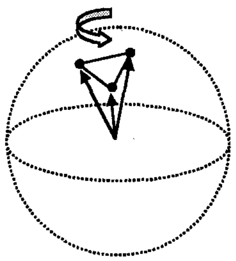

Figure 14.

$$\begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}$$

Figure 15.

The in-circle test in a plane is illustrated in Figure 16. It can be performed by taking the determinant of the matrix in Figure 17. Again, a positive determinant means the point under test is inside the circle, a negative determinant means the point is outside of the circle, and zero indicates the point is on the circle.

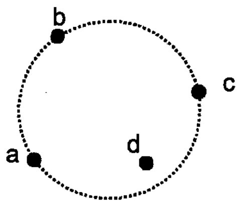

Figure 16.

$$\begin{vmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{vmatrix}$$

Figure 17.

The spherical in-circle test reuses the spherical counterclockwise test. As shown in Figure 18, the three vertices of the triangle establish a plane in three-dimensional space. Because the point under test must be on the surface of the sphere, it is sufficient to determine if the point under test is on the same side of the plane established by the three vertices as the center of the Earth. If Point D in Figure 18 is on the same side of the plane as the Earth center, then it is outside of the circle established by the vertices. If the point under test is on the other side of the plane from the Earth's center, then the point is in the circle. If the point under test is on the plane, then it is on the circle established by the vertices. The test is the same as the counterclockwise test, except using the point under test as the origin of the vectors. The test consists of calculating the determinant shown in Figure 19.

Each of the primitives can be calculated by taking a determinant. However, the exact value of the determinant is not required, only the sign. In general, the sign of the determinant can be calculated more quickly than the value [3].

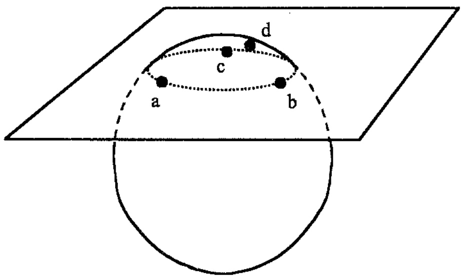

Figure 18.

$$\begin{vmatrix} a_x - d_x & a_y - d_y & a_z - d_z \\ b_x - d_x & b_y - d_y & b_z - d_z \\ c_x - d_x & c_y - d_y & c_z - d_z \end{vmatrix}$$

Figure 19.

5. LOCATING THE ENCLOSING TRIANGLE

Locating the triangle containing an arbitrary point takes $O(n/2)$ time. On a sphere, select an initial triangle with vertices less than 180° from the point. Select one of the edges of the triangle and test whether the point is potentially in the triangle using the counterclockwise test. If this test shows the point is outside of the triangle, then move to the adjacent triangle and continue testing. When a successful test is performed, check the next edge of the triangle. If the point is outside of it, then move to the adjacent triangle and continue testing. If it is inside the this edge, then we have determined the point is between two of the sides. Eventually, the point must be between two edges. Test the point against the third side. If it is inside, then we have located the triangle and we are finished. If not, move to the triangle opposite the last tested edge and continue testing.

6. CONCLUSION

The area-based sea routing has been implemented in the strategic deployment portion of the Joint Warfare System, a simulation used to provide force structure analysis and course of action analysis for the Office of the Secretary of Defense and the major combatant commands. For this use, the triangulation is calculated the first time a port is used as a destination and then saved for later use. The algorithm is being evaluated for use in the naval engagement simulation by triangulating smaller areas such as seas and gulfs.

After partitioning the maritime areas, each triangle and each point has an exit point that is the next point on the shortest path to the desired port. After arriving at the first point, from then on travel is along edges of triangles. Because the triangles can be precalculated, the next point is always a single look-up. During execution, the only calculation is to find the triangle that contains the ship. Once the ship is placed, the entire route can be found by tracing next points. As the total distance is known at each point, it is not necessary to have the whole route to calculate sailing time, only the distance to the next point.

Because the route segments (triangle edges) are great circle arcs, the routes are actual shortest paths, unlike the SAIL network. Additionally, there are no nodes between turn points, so the number of nodes on a route is less with the network-based system. Finally, it is simple to determine a shortest path to any port for a ship at sea, making it easy to change destinations.

REFERENCES

1. J. Shewchuck, Triangle: A two-dimensional quality mesh generator and delaunay triangulator, `http://www.cs.cmu.edu/~quake/triangle.html`.
2. J. Storer and J. Reif, Shortest paths in the plane with polygonal obstacles, *Journal ACM* **41** (5), 982–1012, (1994).
3. J. Shewchuck, Adaptive precision floating point arithmetic and fast robust geometric predicates. Technical Report CMU-CS-96-140, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. (1996).