Minimum Link Path Finding Methods in 2D Polygons

Will McAveney

CS 633 Fall 2007

December 8, 2007

Abstract

The author surveys methods for finding a path between two arbitrary points within a 2D polygon under the link-distance metric. Early algorithms limited to simple polygons are explored first, and categorized into two different approaches. These methods are then evaluated for expansion into non-simple polygons (i.e. spaces with "Obstacles" within the polygon boundary). Then, the problem of finding a minimum link path with obstacles is examined. Visibility based approaches are discussed, and similarities and optimizations made to algorithms for simple polygons are discussed.

A critique of these methods based on bit-precision is noted, as well as related problems such as rectilinear variants, combined metrics (link and Euclidean distance), and pre-processing algorithms which charge a large amount of work to building an initial data structure, which then can perform arbitrary link distance queries quickly.

1. Introduction

The minimum link path problem can be stated formally as follows: For two points s and t within a polygon P, find a straight-line path contained entirely within P that contains the minimum number of line segments. Points s and t can be vertices, non-vertex points on the polygon boundary, or arbitrary points within the polygon depending upon the problem variant. P can be a simply polygon, or a complex polygon contained holes/obstacles.

The problem is partially motivated by a simple problem in robot motion planning. Suppose you have a robot capable of only two operations, forward motion and in-place rotation. For this simplified model, assume forward motion is infinitely "cheap" compared with rotation, and that the robot is point-sized. In such a problem, the minimum distance problem should use a link-distance metric rather than a Euclidean distance one. Link distance refers both to the number of "bends" or "turns" in the path, or the number of individual line segments contained in the path (they are logically equivalent).

Real-world situations that might have something resembling this property include situations where forward motion is free (provided by gravity, etc.), or where vulnerability is a concern. In-place rotation in a military situation makes the automaton easy to attack or destroy. Aside from this, link-distance is generally considered to be a relevant distance metric in robotics, owing to robot wear-and-tear, the fact that acceleration uses fuel, braking wastes momentum, and that the rotation is more time consuming than moving forward toward the goal.

Outside of robot motion planning, the minimum link path problem has a direct application in the field of communications. There are types of communication (most notably certain types of wireless communication) that require line-of-sight in order to communicate. "Turns" in these communication networks can only be accomplished by the installation of expensive towers or communication centers to act as repeaters and change the direction of the signal.

This problem is closely related to the minimum Euclidean distance problem. Techniques for solving link-distance queries often involve heavy use of the Euclidean shortest path (see Section 2) or use visibility graph techniques similar to solutions for the Euclidean distance problem (see Section 3). In the literature, a reader can often find publications by the same authors studying the two problems around the same time under the same problem constraints [1] [2]. In addition, the two distance metrics are often complementary, so there are techniques for combining the two metrics into a weighted average; algorithmic solutions have been developed for this combined metric problem (see Section 5). The problem also has some important variants, such as rectilinear link distance, which have other applications (see Section 5). In addition, there are some pre-processing methods that build data structures in relatively high asymptotic runtime in order to support minimum link queries in more efficient runtimes. These can be useful in applications that must calculate many paths for a given polygon (see Section 6).

This problem differs from the Euclidean distance problem, however, in that the minimum link path is not confined to the vertices of the polygon. In the Euclidean distance problem, the path "hugs" the polygon and/or obstacle boundaries. This tendency is due to the metric itself; the shortest path always takes the "tightest" path that it can around a turn. Link distance problems do not exhibit this property, and can include arbitrary points in the polygon. In addition, the Euclidean shortest path is generally unique (though it is possible to engineer a problem where two or more shortest paths exist). The minimum link path is normally not unique; in fact, in a typical problem, there are an infinite number of minimum link paths. This additional complexity makes the solutions to the problem subject to robustness constraints, which will be discussed in Section 7.

As we will see, the minimum link path is well studied in simple polygons and linear time algorithms exist which solve the problem. The linear time algorithms fall into two categories of approach, one of which explicitly uses the Euclidean shortest path as a basis for developing the minimum link path, and the other uses visibility regions to compute a solution. Once we move to non-simple polygons, we see only one of the two approaches used. Minimum link paths for non-simple polygons are an open problem in computational geometry, and the best-known solutions are quadratic [3] [4].

2. Greedy Path Algorithms for Simple Polygons

Minimum-link paths for simple polygons can be computed with an explicit reference to the Euclidean shortest path. The basic idea with this class of algorithm is to find interesting edges of the shortest path that can be lengthened and then strung together to compute the minimum link path.

The concept of an "eave" [5] or "inflection edge" [6] is central to this means of computing the minimum link path. An inflection edge is a link on the Euclidean shortest path whose predecessor and successor edges lie on opposite sides of the line containing the inflection edge (or, if you prefer, the extension of that edge to the polygon boundary). An interesting property of these edges is that any minimum link path (in a simple polygon) must pass through the infection edge and its extensions. Therefore, whatever section of the minimum link path that passes through the edge may, in fact, be replaced by that edge. By computing the minimum link path between these inflection edges on the shortest path, the minimum link path can be computed [5]. This can be computed in O(log n log log n) with an algorithm using a parallel model of computation with O(n) processors.

It seems from the lack of attention in the literature that this type of computation is effectively limited to simple polygons. Intuitively, it seems that, once obstacles are introduced, that the minimum link path may not have any edges in common with the Euclidean shortest path. It seems that these techniques rely on the property within simple polygons that the shortest path and minimum link path travel in the same direction and can contain edges in common.

This does not appear to be the case with the next type of algorithm, which deals explicitly with the visibility characteristics of the polygon, and seems to generalize to any 2 dimensional polygon.

3. Visibility-based Approaches in Simple Polygons

Visibility-based solutions to the minimum link path problem rely on the property that a visibility region for a given point (or set of points) contains every point reachable on a single line segment path from that point or set of points. Because of this, the minimum link path can be viewed instead as a minimum set of visibility regions. From a given point, you can compute a visibility region V(0). Then, from that visibility region V(0), you can compute the set of points visible from V(0), which is then V(1). Once you compute a visibility region that contains your destination, you can develop a path running through the successive visibility regions that is a minimum-link path from the start to the destination point.

However, visibility region computation is an expensive process, and a naive algorithm can have O(n^3) time [7]. However, O(n) algorithms for computing visibility polygons within a simple polygon. For example, Suri [8] develops a minimum link path algorithm in pre-triangulated simple polygon relying on prior algorithms for computing the shortest path tree of a triangulated polygon and the linear time weak visibility polygon computation of Guibas, et al. [9].

The notion of weak visibility can be stated as follows: the region V(e) visible from line segment e in a simple polygon can be computed by computing the shortest path trees for the endpoints of e to all other points on the polygon. This can be done in linear time. Once these are constructed, draw sets of tangent lines across the regions where the shortest path tree is inward convex. These tangent lines exit the inward convex region of the shortest path tree and delineate what is visible in the next “section” of the polygon.

Suri refers to these lines as “windows” into the next visibility region of the polygon. You can then compute the weak visibility region of each of the windows in order to compute the next visibility region in the polygon.

Once you have iteratively computed all the visibility regions in the polygon, a path can be computed using the series of windows which is the minimum link path for the polygon. When Suri developed this algorithm, the best known algorithm for polygon triangulation was O(n log log n), while his minimum link path algorithm was O(n) once the polygon was triangulated. The running time of the algorithm was dominated by the triangulation, so it was originally an O(n log log n) algorithm. However, subsequent work succeeded in triangulating simple polygons in linear time [10], so Suri’s algorithm now can be implemented completely in linear time.

The practical benefit of this type of algorithm is that, unlike the techniques in the previous section, the basic approach is extensible to polygons containing “holes” or “obstacles”. However, the presence of obstacles greatly increases the complexity of the visibility region computation, so some additional techniques are needed to accomplish this task in complex polygons without incurring an unacceptable runtime. The next section describes one such set of techniques.

4. Minimum Link Paths in Polygons with Holes

The problem that motivated this project was open question #22 in the Open Problems project in computational geometry, as maintained by Demaine, O’Rourke, and Mitchell [4]: Can a minimum link path in a polygon with obstacles be found in sub-quadratic time? A naive extension of the visibility method described in Section 3 can result in an exponential number of paths. A well-known adaptation of Suri’s basic approach was made by Mitchell, et. al [2], and runs in essentially quadratic time.

The method described in Section 3 does not immediately generalize to polygons with holes. However, certain simple optimizations may be made to that basic approach in order to eliminate unnecessary branches and simplify the calculation of successive visibility regions given the possibility of multiple outgoing paths from the current visibility region. Lacking these optimizations, an straightforward extension of Suri’s approach could have a runtime approaching O(n^4). Mitchell’s algorithm avoids computing all the visibility regions explicitly, and avoids much unnecessary calculation.

The first optimization made rests on the observation that the partitioning of a convex polygon results in a decomposition of the polygon into cells. Two sections of the polygon may be completely cut off from each other by a visibility region calculated as part of this algorithm. It is trivial to observe that only one of these two cells can possibly contain the destination point. Effectively, any cell not containing the destination point is ignored by this algorithm (in implementation, it is actually added to the illuminated region, but it is irrelevant to the calculation of the minimum link path).

Second, Michell et al. prove a lemma which helps to simplify the illuminated region at V(k), namely that any point visible from V(k) is also visible from the relative convex hull of V(k). That is, the illuminated region at V(k) can be extended to cover the convex hull relative to the polygonal obstacles. This augmented visibility polygon is then used in the next iteration of visibility region calculation. The use of a relative convex hull simplifies the calculation by allowing treatment of the obstacles touched by the convex hull as polygon boundaries. This reduces the number of weak visibility regions we need to extend into the next cell. In addition, making the relative convex hull part of the visibility polygon may divide the target cell into two or more cells. With the previous optimization, any of those cells which gets cut off from the destination can be added to the visibility polygon, further simplifying the target cell.

As with Suri's algorithm, we iteratively calculate visibility regions until we capture the destination point. It is straightforward to construct the minimum link path from the set of visibility polygons calculated by storing backward pointers to previous visibility regions.

The runtime of this algorithm once all these optimizations is made is $O(E(n)a(n)\log^2 n)$. E(n) is the size of the visibility graph, which makes this algorithm output sensitive. a(n) is a slow growing inverse of Ackermann's function, and can be treated essentially as a constant.

In sum, and taking into account the output sensitivity of the technique, this is essentially a quadratic algorithm. The paper also establishes an n log n lower bound for a weaker version of the problem (that of calculating whether the minimum link path between s and t consists of at most L links), so there is implied some suspicion that the minimum link path in a complex polygon might be calculable more efficiently than quadratic time. However, this author could find nothing in the literature to suggest this has been done, and recent (2006) literature surveys like that contained in [11] do not indicate the problem has been solved in sub-quadratic time.

This is not to say that the subject area has not been explored since the 1990s. On the contrary, there are many related problems (or sub-problems) of the general minimum link path problem in 2D that have been examined, and sometimes optimally solved, in the past two decades. A brief sampling of these problems follows in the next section.

5. Related Problems

A major variation of this problem is the rectilinear minimum link path problem. In a rectilinear problem, all paths must be axis-aligned; that is, they can only move in one dimension at a time. An example might be a set of city streets aligned in a grid (although this also adds a resolution constraint to the problem; this is usually referred to as a "Manhattan Distance" problem or similar).

An important real world application for rectilinear paths occurs in Very Large Scale Integration (VSLI) circuit design. In such a problem, putting "bends" in the circuit paths is a relatively expensive operation due to high fabrication costs. Circuit paths designed with the minimum possible number of links reduce the costs. Fortunately, it is actually easier to compute rectilinear minimum link paths (even in the presence of obstacles) than it is to solve the general minimum link distance problem. O(n log n) solutions exist to the rectilinear problem with obstacles, as compared with the quadratic method available for the general link-distance metric ([11] contains a survey of these techniques).

The literature on this subject also contains much discussion and analysis of "combined" metrics for robot path planning. In real world applications, the reasons for preferring a link-distance metric are not absolute, and the advantages of using a minimum link path are irrelevant if, for example, a minimum link path has a Euclidean distance so far in excess of the shortest geodesic path that the shortest path is preferred for the terrain. A rectilinear version of this problem can be found in [12].

Finally, one would be remiss in not mentioning that the shortest (geodesic) path in a plane with obstacles was eventually solved in O(n log n) time [13]. Considering the similarities for calculating paths based upon the geodesic and link-distance metrics, this adds to the suspicion that a more optimal algorithm must exist for the link-distance problem.

6. Pre-processing Methods

A number of additional approaches have been proposed, which offer an inefficient (quadratic or higher) pre-processing step that enables efficient link distance queries. These techniques can be useful in applications where the polygon environment will be queries for multiple paths between a number of different points within the polygon. The central insight to these problems is that many of the pieces of information needed to construct a minimum link path are common to many paths, and can be extracted at the beginning into efficient data structures to facilitate calculation of the path. These do not improve the efficiency of the basic problem, but rather charge back inefficiency to a pre-processing step that can be computed at a convenient, non-time sensitive moment in a program.

An example can be found in [14]. The authors pre-process a simple polygon in O(n^3) time, and are then able to process link distance queries in O(log n) time, a large improvement over the O(n) algorithms known for link distance within a simple polygon. The data structure built in cubic time contains the visibility graph for the polygon, a set of "atomic segments", which are portions of the polygon boundary which are divided from one another by the extensions of the visibility graph, and the various window partitions for the vertices of the polygon. The data structures are optimized to locate so-called "pinned" edges and inflection edges in the shortest paths between arbitrary points in the visibility graph of P. The pinned edges are rotationally constrained due to having polygon vertices tangent on opposite sides of the edge.

From this data structure, the query algorithm can determine in O(log n) time the shortest geodesic path between two arbitrary points in the polygon. If the path contains an inflection edge, the portion of the minimum link path at this edge can be determined in a manner similar to Section 2. If there is no inflection edge, there is an O(log n) procedure using the window partitions and atomic edges to calculate the minimum link path.

An example in the rectilinear world can be found in [15]. de Berg first applies a standard polygon cutting procedure to rectilinear worlds and proves that a cutting can be done in a rectilinear manner and (within provable constraints) evenly divide the weight of the polygon. Using the cutting procedure, a data structure can be developed in O(n log n) time that facilitates a divide and conquer approach to the minimum link problem. Once the data structure is created, minimum link path queries can be processed on O(log n) time. Again, the rectilinear variant of the problem restricts the problem space fairly significantly, allowing for a relatively efficient pre-processing solution. For VSLI, it seems that the pre-processing method would be particularly useful, as you may have to compute many different circuit paths in a given environment.

7. Robustness Constraints

In the mid to late 1990s, many theoretical techniques in computational geometry were scrutinized on the basis of robustness. Many algorithms assumed a Euclidean space where exact coordinate computation was possible. As these algorithms were intended for implementation in digital processors, this assumption would not hold in real-world computation. There are precision issues associated with the representation of floating point numbers in a finite bit space. In many cases, this limitation can significantly change the worst-case runtime or even the correctness of a geometric algorithm. Minimum link path-finding methods are no exception.

For example, in [16] Kahan and Snoeyink argue that merely storing a minimum link path can require a super-quadratic number of bits in the worst case. Most previous work, including most of what is covered in this paper, assumes a “real RAM” model of computation where exact precision is possible. This simplifying assumption trivially does not hold. Alone, this is not a very interesting observation, but Kahan and Snoeyink actually quantify both the worst-case bit complexity of minimum link techniques, and analyze its effect on the correctness of the algorithm (i.e. how many links imprecision adds in the worst case).

First, they present a simple example that they use to develop a worst-case bit complexity for minimum link paths. Imagine a square spiral, exactly one unit wide, whose sides lengthen by one unit at every turn of the spiral. Further, imagine this spiral extends to infinity, and construct the minimum link path (really, the “minimum link path” is the boundary for points in the spiral reachable from the origin in one link, two links, and so on). An illustration is found in figure 1 of the paper. The intersection points of the path with the borders of the spiral fall “off the grid”, i.e. not on the integer boundaries that all the points on the spiral lie on. Further, as the minimum link path extends to infinity, the intersection points asymptotically approach, but never reach, the integer grid boundaries. It is easy to see that representing the minimum link path requires a greater amount of precision as the spiral extends out. Kahan and Snoeyink are able to make an induction proof that show the bit complexity of the minimum link path to be $\Theta(n^2 \log n)$, while the bit complexity of the coordinates of the spiral is $\Theta(n \log n)$.

In addition, Kahan and Snoeyink analyze the effects of imprecision on minimum link paths. First, they find that using only so-called first-derived points, you can approximate the minimum link path with a worst case upper bound of two times the number of links in the "true" theoretical minimum link path. A first derived point is a point derived from the intersection of two lines drawn through input vertices. However, they admit to some difficulty developing an example where the worst case of twice the number of links would occur. Instead, they present an example where a 33% increase in the link-distance occurs using first derived points only.

Finally, they develop a solution where both the input and output points are restricted to an N by N grid (i.e. an arbitrary, fixed, evenly spaced precision). Using a method which approximates a line segment using a series of grid-aligned rays, they show that the approximate path can be computed to use only $\Theta(\log N)$ additional links.

Robustness issues limit the effectiveness of some of the more mathematically based computation techniques for minimum link paths. In more recent work on visibility and minimum link paths (see [6]) you tend to see precision constraints built directly into the algorithm parametrically. The robustness critique seems to be considered as a central issue of interest to the theoretical computer scientist, rather than an implementation issue left to production programmers.

8. Findings

The problem of finding a minimum link path within a 2d polygon seems to be extremely well-studied for simple polygons. Minimum link paths can be calculated very efficiently in simple polygons with algorithms available for 15-20 years. However, interesting work continues on this and related problems such as pre-processing polygons for link distance queries, calculating the link center of a simple polygon, or simple 2d visibility.

The problem that prompted this project was finding a minimum link path in a polygon or plane in the presence of obstacles. This is something of a open question. The best known algorithms, like the one contained in [2], are quadratic time algorithms that beg improvement. Further, work on related problems such as rectilinear link distance, Euclidean shortest path, pre-processing methods, link center, visibility and convex hulls seems to suggest that a sub-quadratic solution to the link distance problem might yet be found. However, the basic visibility techniques are now 20 years old, and it may be that such an algorithm is either not possible, or requires an entirely different approach to the problem than anything tried thus far.

Finally, the theoretical work in Euclidean space is subject to some important limitations based on the limited precision of efficient calculation on digital computers. The minimum link path derives additional information from input coordinates, unlike related problems like Euclidean shortest distance, and may require more precision than a given hardware platform can provide. Techniques to limit precision incur a penalty of either runtime complexity, or “incorrectness”, i.e. paths that are only approximately minimal.

9. Conclusion and Future Work

This paper has made a basic survey of work on minimum link path-finding methods in 2 dimensional polygons. We have examined linear time methods available for simple polygons, quadratic techniques for complex polygons, and related problems such as rectilinear link distance. The relative complexities of these problems have been compared, and some real world applications discussed.

We have also looked at pre-processing methods that incur a high initial cost to thoroughly examine an input polygon for elements useful for minimum link path calculation. For many applications, where a large number of paths must be queried from a static environment, these pre-processing methods may be the best available techniques.

Finally, a well-known critique of the robustness of minimum link path-finding algorithms based on a “real RAM” model of computation has been discussed. The potential for super-quadratic bit complexity requires trade offs among run-time efficiency, correctness, and storage. The nature of the minimum link problem can require a recursive series of derived points, i.e. “first-derived” points derived from input vertices, points derived from “first-derived” points, and so on.

Future work on this problem will involve a deeper look into related problems such as link-center, combined metrics, and visibility with precision constraints. In addition, this project was limited to “exact” (or, given the robustness constraints, “close to exact”) methods of determining a minimum link path. Randomized methods for path-finding under the link distance metric might be an area for additional study. While higher dimension motion planning, like that for high-degree-of-freedom robots, necessarily places finding a path (any path) above any sort of distance metric optimization, there are likely random or semi-random methods for 2 dimensional motion planning that were not covered in my research.

Bibliography

[1] Mitchell, J. S. 1993. Shortest paths among obstacles in the plane. In Proceedings of the Ninth Annual Symposium on Computational Geometry (San Diego, California, United States, May 18 - 21, 1993). SCG '93. ACM, New York, NY, 308-317.

[2] Mitchell, J. S.; Rote, G.; Woeginger, G. 1990. Minimum-link paths among obstacles in the plane. In *Proceedings of the Sixth Annual Symposium on Computational Geometry* (Berkley, California, United States, June 07 - 09, 1990). SCG '90. ACM, New York, NY, 63-72.

[3] Mitchell, J.S.; O'Rourke, J. 2001. Computational geometry column 42. *Internat. J. Comput. Geom. Appl.*, 11(5):573-582.

[4] Demaine, E.; Mitchell, J.S.; O'Rourke, J.. *The Open Problems Project (Problem 22)*. http://maven.smith.edu/~orourke/TOPP/.

[5] Chandru, V.; Ghosh, S.K.; Maheshwari, A; Rajan, V.; Saluja, S. NC 1995 Algorithms for minimum link path and related problems. *Journal of Algorithms* (19) 1995.

[6] Ben-Moshe, B.; Hall-Holt, O.; Katz, M.; Mitchell, J.S. 2004 Computing the Visibility Graph of Points Within a Polygon. *SCG'04*, June 8–11, 2004, Brooklyn, New York, USA.

[7] de Berg, M.; van Kreveld, M.; Overmars, M.; Schwarzkopf, O. 2000 Computational Geometry: Algorithms and Applications. Second Edition. Springer, New York, 2000.

[8] Suri, S. 1987. Minimum Link Paths in Polygons and Related Problems. Ph.D. Thesis. Johns Hopkins University, Baltimore, MD 1987.

[9] Guibas, L.; Hershberger, J.; Leven, D.; Sharir, M.; Tarjan, R. 1986. Linear time algorithms for visibility and shortest path problems inside simple polygons. In *Proceedings of the Second Annual Symposium on Computational Geometry* (Yorktown Heights, New York, United States, June 02 - 04, 1986). SCG '86. ACM, New York, NY, 1-13.

[10] Chazelle, B. 1991. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.* 6, 5 (Aug. 1991), 485-524.

[11] Wanger, D. 2006. Path Planning Algorithms under the Link Distance Metric. Ph.D. Thesis. Dartmouth College, Hanover, NH. February 2006.

[12] de Berg, M; van Kreveld, M; Nilsson, B. J. 1991. Shortest path queries in rectilinear worlds of higher dimension (extended abstract). In *Proceedings of the Seventh Annual*

Symposium on Computational Geometry (North Conway, New Hampshire, United States, June 10 - 12, 1991). SCG '91. ACM, New York, NY, 51-60.

[13] Hershberger, J.; Suri, S. 1999. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM J. Comput.* 28, 6 (Aug. 1999), 2215-2256.

[14] Arkin, E. M.; Mitchell, J. S.; Suri, S. 1992. Optimal link path queries in a simple polygon. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms* (Orlando, Florida, United States). Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, 269-279.

[15] de Berg, M. 1991. On rectilinear link distance. *Comput. Geom. Theory Appl.* 1, 1 (Jul. 1991), 13-34.

[16] Kahan, S.; Snoeyink, J. 1996. On the bit complexity of minimum link paths: superquadratic algorithms for problems solvable in linear time. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry* (Philadelphia, Pennsylvania, United States, May 24 - 26, 1996). SCG '96. ACM, New York, NY, 151-158.