Minimum-Link Paths Revisited*

Joseph S. B. Mitchell† Valentin Polishchuk‡ Mikko Sysikaski†

Abstract

A path or a polygonal domain is C-oriented if the orientations of its edges belong to a set of C given orientations; this is a generalization of the notable rectilinear case ($C = 2$). We study exact and approximation algorithms for minimum-link C-oriented paths and paths with unrestricted orientations, both in C-oriented and in general domains. We obtain the following results:

- An optimal algorithm for finding a minimum-link rectilinear path in a rectilinear domain; using only elementary data structures, our algorithm and its analysis are simpler than the earlier ones of [Sato, Sakanaka and Ohtsuki, 1987], [Das and Narasimhan, 1991].
- An algorithm to find a minimum-link C-oriented path in a C-oriented domain; our algorithm is simpler and more time-space efficient than the earlier one of [Adegeest, Overmars and Snoeyink, 1994].
- An extension of our techniques to find a C-oriented minimum-link path in a general (not necessarily C-oriented) domain.
- 3SUM-hardness of finding a minimum-link path with unrestricted orientations (even in a C-oriented domain). This answers a question from the survey of Mitchell [Goodman and O'Rourke, eds., 1997, 2004] and Problem 22 in The Open Problems Project.

We also present approximation methods for link distance, including the following:

- A more time-space efficient algorithm to find a 2-approximate C-oriented minimum-link path.
- A notion of "robust" paths. We show how minimum-link C-oriented paths approximate the robust paths with unrestricted orientations to within an additive error of 1.
- A subquadratic-time algorithm with a non-trivial approximation guarantee for the general (unrestricted-orientation) minimum-link paths in general domains.

All of our algorithms not only find minimum-link paths but also build, within the same time and space bounds, the corresponding link distance maps—exact or approximate. For instance, using our algorithms, one can construct in subquadratic time linear-size approximate (additive or multiplicative) maps for general minimum-link paths in general domains; this is in contrast with the exact link distance maps, which may have quartic complexity [Suri and O'Rourke, 1986].

1 Introduction

Minimum-link problems arise in motion planning domains were turning is expensive, in line simplification, guarding applications, VLSI, wireless communication, and other areas. An instance of the problem is specified by an n-vertex polygonal domain P with h holes, and two points $s, t \in P$; the goal is to find an s-t path with fewest edges (links). In the query version of the problem, the goal is to build a data structure (link distance map) to answer efficiently link distance queries to s.

We give algorithms for computing exact and approximate C-oriented minlink paths and show how they approximate "robust" paths with unrestricted orientations; we also give approximation algorithms for the general minlink problem.

1.1 Previous work

If P is a simple polygon, a minlink path can be found in linear time [7, 9, 21]. The general idea is to do the "staged illumination" of P [8, Sections 26.4, 27.3] that starts from illuminating the visibility polygon of s. At the beginning of any stage the boundary between the lit and the dark parts of P is a set of "windows"; each window w is a chord cutting out a dark part $P_w \subset P$. The crux of the linear-time algorithm is that P_w is "unique" to w: the part of P_w illuminated at the next stage is exactly what is seen from w (even if a point $p \in P_w$ is seen from another window $w' \neq w$, we do not care about shining light from w' into P_w). At any stage, the illumination is guided by DFS in the dual of a triangulation of P; since the dual is a tree, any triangle is lit only once (triangles intersected by windows are first lit only partially, and are fully lit at the next stage).

In polygonal domains with holes, the algorithm of Mitchell, Rote and Woeginger [16] computes a minlink

*Longer version is available online at: `http://hiit.fi/valentin.polishchuk/ml.pdf`

†Stony Brook University, `jsbm@ams.stonybrook.edu`

‡University of Helsinki, `polishch@cs.helsinki.fi`

path in $O(n^2\alpha^2(n)\log n)$ time. It was believed that a faster algorithm is possible (e.g., in [6, p. 263] the result of [16] is called "suboptimal"). Nevertheless, the only previously known lower bound, also due to [16], was $\Omega(n \log n)$; the same bounds for the *rectilinear* case are given in [6, 14]. Also, no subquadratic-time approximation algorithm is known.

Robustness of paths was previously addressed in terms of *distance* tolerance by employing high-clearance [18, 24, 25] or thick-paths [2, 5, 13, 17] models. Similar, distance-based robustness was considered in the context of *curvature-constrained* paths in [3, 12, 23].

Minimum-link C-oriented paths in C-oriented domains were studied by Adegeest, Overmars and Snoyeink [1]. Two algorithms are presented in [1]: one running in $O(C^2 n \log n)$ time and space, the other—in $O(C^2 n \log^2 n)$ time and $O(C^2 n)$ space. Both algorithms build C trapezoidations of the domain and label the trapezoids with link distance from s; this way an $O(Cn)$-space structure is created that answers link distance queries in $O(C \log n)$ time. The labeling of the trapezoids proceeds in n *steps*, with label-k trapezoids receiving their label at step k. By induction on k, any such trapezoid must be intersected by a label-$(k-1)$ trapezoid of a different orientation; hence, step k boils down to detecting all unlabeled trapezoids intersected by label-$(k-1)$ trapezoids. In terms of the *intersection graph* of trapezoids (which has nodes corresponding to trapezoids and edges corresponding to intersecting pairs of trapezoids), the labeling is the BFS starting from the C-oriented segments through s.

The idea of performing an efficient BFS in the intersection graph without building the (potentially quadratic-size) graph explicitly, dates back to the work on minimum-link *rectilinear* paths [6, 10, 11, 14, 15, 19, 20, 26, 27]. Imai and Asano's [10, 11] data structure allows one to do the BFS—and hence find a minlink rectilinear path—in $O(n \log n)$ time and space. Still, it was not until the work of Das and Narasimhan [6] (and the lesser known work of Sato, Sakanaka and Ohtsuki [20]) that an optimal, $O(n \log n)$-time $O(n)$-space BFS implementation was developed. In the implementation, one step of the BFS is reduced to a pair of sweeps—the UpSweep and the DownSweep.

2 Overview of the results

Rectilinear paths in rectilinear domains The data structures employed in [6, 20] are simpler than the (much more general-purpose) structure of Imai–Asano [10, 11], but they are still more complicated than one would hope for the basic problem of finding rectilinear paths amidst rectilinear obstacles. We present a simplified implementation of the BFS step in the intersection graph; our modification does not affect the asymptotic time and space optimality of the algorithm. The crux of the simplification is the use of a *single* tree for storing the intersection of the sweepline with the trapezoids that were lit at the previous step.

Another, minor modification present in our algorithm comes from performing only one, upward sweep, at any step of the BFS. The sweep starts from what we call the "pot" trapezoids—those into which the different-orientation trapezoids lit at the previous step are "planted". The planting is nothing but a means to initialize the sweep (without the planting, it is not clear how to discover efficiently even a single edge in the intersection graph). While our modification saves only a factor of 2 in running time, it serves as the basis of our improvements for the general case of C-oriented paths with $C > 2$.

C-oriented paths in C-oriented domains As noted in [1], the efficient methods developed to perform BFS in the trapezoids intersection graph for the rectilinear version do not extend to C-oriented paths when $C > 2$. We look closely at why this is the case. One reason is that for $C > 2$ some trapezoids may get labeled only partially during a BFS step; this complicates the BFS because the intersection graph changes from step to step, and, in the final link distance map, trapezoids may get split into subtrapezoids. The partial labeling and splitting are due to the possibility that two different-orientation trapezoids do not "straddle" each other; instead they both may be "flush" with an obstacle edge whose orientation is different from the orientations of both trapezoids (this was not the case in the rectilinear version since there were only 2 orientations). However, such flushness can be read off easily from lists of incident trapezoids stored with every edge of P; thus, discovering partially labeled trapezoids becomes the easy part of the algorithm.

After the partially labeled trapezoids are processed, we are left with discovering unlabeled trapezoids "fully straddled" by trapezoids labeled at the previous step. As with the rectilinear case, it is the straddling that leads to a superlinear-size intersection graph and makes a subquadratic algorithm less trivial. It is tempting to reuse here the sweeping techniques developed for the rectilinear version; the stumbling block, though, is choosing the direction of the sweep. Indeed, no matter in which direction the sweep proceeds, the intersection of the sweepline with a trapezoid does not change only at discrete "events": while the sweepline intersects a non-parallel side of the trapezoid, the intersection changes continuously. The good news is that this is the *only* reason for a continuous change of the intersection. This prompts to get rid of the non-parallel sides of the trapezoids by clipping them into parallelograms, with the new sides parallel to the

sweepline; after the clipping (and planting the parallelograms appropriately) is done, we are able to reuse the rectilinear-case machinery and finish the BFS step with $C(C-1)$ sweeps—one per pair of orientations. Overall we obtain an $O(C^2 n \log n)$-time $O(Cn)$-space algorithm.

C-oriented paths in general domains We generalize our techniques to compute the C-oriented link distance map also in domains with *unrestricted* orientations of edges. We observe that the algorithm for C-oriented path in a C-oriented domain uses C-orientedness of the domain only to bound the complexity of the final map (without the C-orientedness, the complexity may blow up). However, the blowup happens only "deep inside" of certain trapezoids. Thus, we declare the deep parts of such trapezoids as a separate cell. The path to a query point q inside the cell consists of 2 parts: path from s to enter the trapezoid, and a "zigzag" of extreme orientations to q. The number of links in the first part is given by the usual map, the number of links in the second part can be determined in constant time (assuming constant-time floor function) because of the regular pattern of the path—it bounces off the sides of the trapezoid until reaching the query point.

Unrestricted paths Our focus on C-oriented path is partially justified by observing that the general minlink path problem is 3SUM-hard—even in C-oriented domains.

2.1 Approximations

2-approximate C-oriented paths The C-oriented link distance may be approximated by requiring that every second link of the path is horizontal. To find a minlink C-oriented path with this requirement one can do the BFS in the trapezoids intersection graph with one modification: instead of checking for intersection between trapezoids for all pairs of orientations, only check for intersections between the horizontal trapezoids and the other ones. Such a modification decreases the running time of our algorithm to $O(Cn \log n)$, but the space remains $O(Cn)$ because the C trapezoidations of P are still constructed.

To reduce the space to $O(n)$ we only do the horizontal trapezoidation, and go back to the rectilinear-case ideas of Das and Narasimhan [6] who label the horizontal trapezoids *without* using the vertical ones (in our case, we label the horizontal trapezoids without using *any* other ones). In particular, without the other trapezoidations we cannot use our planting (there is simply nothing to plant!) to initialize the sweep; thus we do both the UpSweep and the DownSweep à la [6] starting from trapezoids labeled on the previous step.

Approximating robust paths with C-oriented paths We define *robust* paths as those whose edges may be "wiggled" without intersecting the obstacles. We prove that C-oriented paths can approximate a robust minlink path to within a one-sided additive error of 1: we show how to build a data structure to answer efficiently the approximate robust link distance queries; to output the approximating C-oriented path itself takes additional time proportional to the link distance. We emphasize that we compute the data structure for approximate link distance queries in *subquadratic time*, which compares favorably with the $\Theta(n^4)$ worst-case complexity of the *exact* map [22].

$O(\sqrt{h})$-approximation for unrestricted paths We show that in $O(n\sqrt{h} + n \log n + h^{3/2} \log h)$ time one can find an $O(\sqrt{h})$-approximate minlink path with arbitrary orientations; this is the first subquadratic-time approximation algorithm with a non-trivial approximation guarantee for the general case. In fact, our algorithm builds a linear-size $O(\sqrt{h})$-approximate link distance map (which again compares well to the $\Theta(n^4)$-size exact map).

The two ingredients of our approximation are low-stabbing-number bridges and staged illumination (window partition) in simple polygons. We bridge the holes to the outer boundary of P, thereby obtaining a simple polygon. We do not make the bridges fully opaque; rather, they are "semi-transparent": we triangulate the simple polygon and do the staged illumination from s, but as we go, each time a bridge is illuminated on one of its sides, at the next step we consider it to be illuminated also on its other side, and continue the staged illumination. In comparison to opt, we are delayed by 1 link every time an edge of opt crosses a bridge; i.e., on every edge of opt we have as many additional vertices as there are bridges that the edge crosses. Using a low-stabbing-number tree for the bridging [4], we ensure that each edge of opt crosses $O(\sqrt{h})$ bridges, and thus we obtain an $O(\sqrt{h})$ multiplicative approximation.

As described above, the illumination takes $O(nh)$ time, since a triangle can be discovered by light emanating from h bridges. To address this inefficiency, we declare a triangle opaque as soon as it is lit $m \leq h$ times. We prove that with the right choice of m this does not delay the illumination by too much, while decreasing the runtime of the illumination to $O(nm)$.

References

[1] J. Adegeest, M. H. Overmars, and J. Snoeyink. Minimum-link c-oriented paths: Single-source queries. *Int. J. Comput. Geometry Appl.*, 4(1):39–51, 1994.

[2] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk.

Maximum thick paths in static and dynamic environments. *Computational Geometry Theory and Applications*, 43(3):279–294, 2010.

[3] J. Barraquand and J.-C. Latombe. Nonholonomic multi-body mobile robots: controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[4] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

[5] L. P. Chew. Planning the shortest path for a disc in $O(n^2 \log n)$ time. In *Proceedings of the 1st Annual Symposium on Computational Geometry*, pages 214–220, 1985.

[6] G. Das and G. Narasimhan. Geometric searching and link distance. *Algorithms and Data Structures*, pages 261–272, 1991.

[7] S. K. Ghosh. Computing the visibility polygon from a convex set and related problems. *J. Algorithms*, 12(1):75–95, 1991.

[8] J. E. Goodman and J. O'Rourke. *Handbook of discrete and computational geometry*. CRC Press series on discrete mathematics and its applications. Chapman & Hall/CRC, 2004.

[9] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom.*, 4:63–97, 1994.

[10] H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM J. Comput.*, 15(2):478–494, 1986.

[11] H. Imai and T. Asano. Dynamic orthogonal segment intersection search. *J. Algorithms*, 8(1):1–18, 1987.

[12] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In Z. Li and J. F. Canny, editors, *Nonholonomic Motion Planning*, pages 271–342. Kluwer Academic Pubishers, Norwell, MA, 1992.

[13] I. Kostitsyna and V. Polishchuk. Simple wriggling is hard unless you are a fat hippo. In *Fifth International Conference on Fun with Algorithms (FUN)*, 2010.

[14] D. T. Lee, C. D. Yang, and C. K. Wong. Rectilinear paths among rectilinear obstacles. *Discrete Appl. Math.*, 70:185–215, 1996.

[15] A. Maheshwari, J.-R. Sack, and D. Djidjev. Link distance problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry, Chapter 12*, pages 519–558. Elsevier Science, Amsterdam, 2000.

[16] J. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 8(1):431–459, 1992.

[17] J. S. B. Mitchell and V. Polishchuk. Thick noncrossing paths and minimum-cost flows in polygonal domains. In J. Erickson, editor, *Symposium on Computational Geometry*, pages 56–65. ACM, 2007.

[18] C. Ó'Dúnlaing and C.-K. Yap. A "retraction" method for planning the motion of a disc. *J. Algorithms*, 6(1):104–111, 1985.

[19] T. Ohtsuki. Gridless routers - new wire routing algorithm based on computational geometry. In *Internat. Conf. on Circuits and Systems, China*, 1985.

[20] M. Sato, J. Sakanaka, and T. Ohtsuki. A fast line-search method based on a tile plane. In *Proc. IEE ISCAS*, pages 588–591, 1987.

[21] S. Suri. A linear time algorithm with minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35(1):99–110, 1986.

[22] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the second annual symposium on Computational geometry*, SCG '86, pages 14–23, New York, NY, USA, 1986. ACM.

[23] H. Wang and P. K. Agarwal. Approximation algorithms for curvature constrained shortest paths. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 409–418, 1996.

[24] R. Wein, J. P. van den Berg, and D. Halperin. The visibility-voronoi complex and its applications. *Comput. Geom.*, 36(1):66–87, 2007.

[25] R. Wein, J. P. van den Berg, and D. Halperin. Planning high-quality paths and corridors amidst obstacles. *I. J. Robotic Res.*, 27(11-12):1213–1231, 2008.

[26] C. D. Yang, D. T. Lee, and C. K. Wong. On bends and lengths of rectilinear paths: a graph theoretic approach. *Internat. J. Comput. Geom. Appl.*, 2(1):61–74, 1992.

[27] C. D. Yang, D. T. Lee, and C. K. Wong. Rectilinear paths problems among rectilinear obstacles revisited. *SIAM J. Comput.*, 24:457–472, 1995.