An ϵ - Approximation Algorithm for Weighted Shortest Path Queries on Polyhedral Surfaces [*][§]

Lyudmil Aleksandrov[†], Mark Lanthier[‡], Anil Maheshwari[‡] and Jörg-Rüdiger Sack[‡]
E-mail: {lanthier,maheshwa,sack}@scs.carleton.ca

1 Introduction

Shortest path problems are among the fundamental problems studied in computational geometry and other areas such as graph algorithms, geographical information systems (GIS) and robotics. Due to their relevance in practice, 3-dimensional shortest paths problems, in particular, have received considerable attention. For surveys on shortest path problems see e.g., [13, 14].

Canny and Reif [4] showed that the problem of computing a shortest path between two points among a set of polyhedra (avoiding the interiors) is NP-Hard. The NP-hardness and the large time complexities of 3-d shortest paths algorithms even for special problem instances have motivated the search for approximate solutions to the shortest path problem. The quality of an approximate solution is assessed in comparison to the correct solution. One particular class of approximation algorithms produces ϵ- approximations of a shortest path. A path $\Pi'(s,t)$ between two points s and t is said to be an ϵ-approximation of a (true) shortest path $\Pi(s,t)$ between s and t, if $\frac{\|\Pi'(s,t)\|}{\|\Pi(s,t)\|} \leq 1+\epsilon$, for some $\epsilon > 0$.

For Euclidean shortest path problems on a single polyhedron, a significant amount of activity has recently emerged with work presented for example by Hershberger and Suri [8], Har-Peled et. al. [7], and Agarwal et. al. [1]. We refer to reader to the full version of this paper for more details on these results.

Weighted shortest paths problems are of particular interest as the Euclidean distance does not always capture the true cost of a path. In the weighted scenario each face of a region has an associated weight which represents the cost of travel through that face. The cost of travel through a face is the distance traveled multiplied by the face's weight. Papadimitriou [16] provided an ϵ-approximation algorithm for computing weighted shortest paths amidst polygonal obstacles in the plane; their algorithm runs in $O(n^4(L+\log(n/\epsilon))^2/\epsilon^2)$ time, where L represents the bit precision. Lanthier et. al. [10] presented several practical algorithms for approximating shortest paths in weighted domains. In addition to their experimental verification and time analysis, they provide theoretically derived bounds on the quality of approximation. More specifically, the cost of the approximation is no more than the shortest path cost plus an (additive) factor of WL, where L is the longest edge length, W is the largest weight among all faces. They also use spanners to get at most β times the shortest path cost plus βWL, where $\beta > 1$ is an adjustable constant. Mata and Mitchell [12] presented an algorithm that constructs a graph (pathnet) which can be searched to obtain an approximate path, but they do not state any bounds on the accuracy of the path obtained.

Very recently, Aleksandrov et al. [2] presented an ϵ- approximation algorithm for weighted shortest paths between two vertices on a polyhedral surface. Their algorithm runs in $O(mn\log mn + nm^2)$ time where $m = O(\log_\delta(L/r))$ and L is the length of the longest edge and $r = f(\epsilon)$ times the minimum distance from any vertex to the boundary of the union of its incident faces. In the unweighted case, $f(\epsilon) = \min(\frac{\epsilon}{4}, 1/6)$ and $\delta \geq 1 + \frac{\epsilon \sin\theta}{4}$, where θ is the minimum angle between any two adjacent edges of $\mathcal{P}$. In the weighted case, $f(\epsilon) = \min(\frac{\epsilon}{2+3(W/w)}, 1/6)$ and $\delta \geq 1 + \frac{\epsilon\sin\theta}{2+3(W/w)}$, where W and w are the largest and smallest face weights of $\mathcal{P}$, respectively. The work presented here is a generalization of [2], which is extended to allow arbitrary query points, (both source and destination) and is easily parallelizable. Our techniques also allow for the computation of a weighted shortest path map. This work represents the first ϵ-approximation algorithm for weighted shortest

[*]Research supported in part by ALMERCO Inc. & NSERC.
[†]Bulgarian Academy of Sciences, CICT, Acad. G. Bonchev Str. Bl. 25-A, 1113 Sofia, Bulgaria
[‡]School of Computer Science, Carleton University, Ottawa, Ontario K1S5B6, Canada
[§]Part of this research was carried out while the first author listed was visiting Carleton University.

path queries on a non-convex polyhedron.

Parallelization of ϵ-approximation weighted shortest paths algorithms is of particular interest as, with more computing power, better approximations are obtainable in the same time or in even less time. Parallel shortest paths problems in graphs have received some attention. As described for example in Leighton, [11], the all-pairs shortest path in a weighted graph can be computed using transitive closure. This leads however to algorithms which have a processor utilization no less than $M(n)$, where $M(n)$ is the time required to multiply to n by n matrices. For other related work see e.g., [6, 17, 9, ?]. Single-source parallel shortest path problems in graphs have e.g., been developed by Paige and Kruskal [15]. Their algorithm runs in $THETA(n \log n)$ time and uses $THETA(n)$ EREW PRAM processors. Cohen presented an NC algorithm that computes shortest paths in a digraph and makes use of a k^{μ}-separator decomposition. The algorithm computes shortest paths from s sources to all other vertices using $O(n^{3\mu} + s(n + n^{2\mu}))$ work. Reachability from s sources can be computed using $O(M(n^{\mu}) \log^2 n + s(n + n^{2\mu}))$ work, where $M(k) = o(k^{2.37})$ is the best known work bound for $k \times k$ matrix multiplication. These bounds are applicable to planar graphs when $\mu = 0.5$. The authors know of no non-trivial parallel algorithms for the weighted shortest paths problem with unknown source and destination queries other than the one presented here.

2 Our Algorithm

Our approach to solving the problem is to discretize the n-face polyhedron $\mathcal{P}$ in a natural way, by placing up to m Steiner points along each edge of $\mathcal{P}$. We construct a graph G containing Steiner points as vertices and edges as those interconnections between Steiner points that correspond to segments which lie completely in the triangular faces of $\mathcal{P}$. The geometric shortest path problem on polyhedra is thus stated as a graph problem so that the existing efficient algorithms (and their implementations) for shortest paths in graphs can be used. We preprocess the given polyhedron $\mathcal{P}$ in order to answer shortest path queries. We will discuss both the preprocessing step and how to answer queries using data structures computed in the preprocessing step.

The vertices (and Steiner points), edges and the faces of $\mathcal{P}$ define a planar graph, which we call the *polygonal graph* G, with the corresponding set of vertices, edges and faces, respectively. Frederickson [?] has shown that any planar n-vertex graph can be partitioned into $O(n/r)$ parts (regions), so that each part consists of $O(r)$ vertices and has $O(\sqrt{r})$ vertices on the boundary of the region. This construction is based on applying the separator theorem due to Lipton and Tarjan [?]. Aleksandrov and Djidjev [?] very recently provided an algorithm for computing such partition; their algorithm runs in linear time. The preprocessing stage of our shortest path algorithm does the following:

1. Using the technique of Aleksandrov et. al. [2], compute a graph G on $\mathcal{P}$ (they introduce m Steiner points on each edge of $\mathcal{P}$, where m is a function of the topology of each triangle of $\mathcal{P}$).
2. Using the algorithm of Aleksandrov and Djidjev [?], partition G into mn/r regions $R_1, R_2, ..., R_{mn/r}$, where r is an adjustable parameter. Let $G_1, G_2, ..., G_{mn/r}$ be the graphs corresponding to each of these regions, respectively. Note that their algorithm partitions a planar graph, where as G may be non-planar. We partition a subgraph of G, consisting of all vertices and only those edges of G that correspond to the region's polygonal boundary. Once the partition is computed on the subgraph, we add the rest of the edges of G to obtain the partition of G.
3. For each region $R_i, 1 \leq i \leq mn/r$, compute shortest paths among all pairs of vertices in G_i.
4. Create for each region R_i, a complete graph G'_i on its boundary vertices. The weight on an edge in G_i between two boundary vertices u and v of region R_i represents the length of the shortest path between them when the path is restricted to be in R_i. Let $G''_i = G_i \bigcup G'_i$.
5. Compute shortest paths between all pairs of boundary vertices from all regions, i.e., compute shortest paths among all pairs of vertices in $G'_1 \bigcup G'_2 \bigcup ... \bigcup G'_{mn/r}$.
6. For each region R_i, compute once again shortest paths between all pairs of vertices in G''_i.

Suppose we wish to know a shortest path between two vertices u and v in G. There are two possibilities: (i) u and v are in the same region, say R_i, or (ii) u and v are in different regions, say R_i and R_j. In the first case, we know the shortest path between u and v, by knowing the shortest path in G''_i. In the second case, we know the shortest path from u (or v) to all the boundary vertices of R_i (resp. R_j). Also we know the shortest paths between every pair of boundary vertices of R_i and R_j. Since there are $O(\sqrt{r})$ boundary vertices per region, we can compute the shortest path between u and v by investigating $O(\sqrt{r} \times \sqrt{r})$ pairs. If the query pair on $\mathcal{P}$ happens to be the points corresponding to vertices of G, then we report the path as above. For arbitrary query points s and t on $\mathcal{P}$, we perform the following steps:

1. Locate the regions R_i and R_j containing s and t, respectively, where R_i may be same as R_j.
2. Locate the faces $f_i \in R_i$ and $f_j \in R_j$ containing s and t, respectively.

3. Add (temporary) edges to G''_i (G''_j) from s (t) to all vertices representing Steiner points of f_i (f_j).

4. Compute a shortest path from s (t) to all vertices on the boundary of R_i (R_j). Investigate all shortest paths between pairs of boundary vertices of R_i and R_j and report a shortest path between s and t.

2.1 Running Time and Accuracy Analysis

In Step 1 of the algorithm, we compute a graph G by discretizing the polyhedron $\mathcal{P}$ which has at most $O(nm)$ vertices and at most $O(nm^2)$ edges. This can be done in $O(nm^2)$ time. In Step 2, the algorithm of Aleksandrov et. al. [2] requires time linear in the number of vertices of G, hence $O(nm)$ time is required. Step 3 calls for the computation of the shortest path between all pairs of r vertices in each graph G_i. We can use any all-pairs-shortest-path algorithm which will cost $O(r^2 \log r + r^2 m)$ time per region and hence $O(nmr \log r + nm^2 r)$ overall time for all regions. Using these results, Step 4 can be completed easily within $O(\sqrt{r}\sqrt{r}) = O(r)$ time. Now using these newly created graphs, Step 5 requires an all-pairs-shortest-path computation on a graph with $\frac{mn\sqrt{r}}{r} = \frac{mn}{\sqrt{r}}$ vertices and $O(mn)$ edges, which can be done in $O(\frac{(mn)^2}{r} \log \frac{mn}{\sqrt{r}} + \frac{(mn)^2}{\sqrt{r}})$ time. Lastly, Step 6 requires the same amount of time as Step 3. As a result, the overall preprocessing phase requires $O(nmr \log r + nm^2 r + \frac{(mn)^2}{r} \log \frac{mn}{\sqrt{r}} + \frac{(mn)^2}{\sqrt{r}})$ time.

Now we can analyze the accuracy of the path obtained. From the results of Aleksandrov et. al. [2], given that s and t are vertices of $\mathcal{P}$, a Euclidean shortest path approximation $\pi'(s,t)$ can be computed such that $|\pi'(s,t)| \leq (1+4\epsilon)|\pi(s,t)|$. For the weighted scenario an approximation $\Pi'(s,t)$ is computed such that $\|\Pi'(s,t)\| \leq (1 + 2\epsilon + 3\epsilon\frac{W}{w})\|\Pi(s,t)\|$, where W and w are the largest and smallest weights of any face in $\mathcal{P}$. However, the preprocessing phase requires computation of shortest paths between Steiner points. Using a similar proof strategy as of [2], we show that a path between Steiner points s and t can be found such that $|\pi'(s,t)| \leq (1+7\epsilon)|\pi(s,t)|$ and $\|\Pi'(s,t)\| \leq (1+2\epsilon+6\epsilon\frac{W}{w})\|\Pi(s,t)\|$. However, this requires that s and t be a certain minimal distance apart. More precisely, let f_i and f_j be the two faces sharing the edge on which s lies. Let τ_s be the set of faces incident to a vertex of f_i or f_j. In order for the above bound to hold, t must lie outside τ_s. If t lies within τ_s then we can apply any existing shortest path algorithm between s and t that remains within τ_s, which contains a constant number of faces in practice, as well as recall that the underlying graph is planar. Our proof strategy is to show that there exists a path in G, that is an ϵ-approximation of a shortest path. It is very likely that Dijkstra's algorithm may find an alternate and even a better cost path in G. Assume now that for queries, s and t are arbitrary points on $\mathcal{P}$ such that $t \bigcap \tau_s = \emptyset$. We show (again in [2]) that there exist paths such that $|\pi'(s,t)| \leq (1+12\epsilon)|\pi(s,t)|$ and $\|\Pi'(s,t)\| \leq (1 + 2\epsilon(1+6\frac{W}{w}) + 6\epsilon^2\frac{W}{w}(1+3\frac{W}{w}))\|\Pi(s,t)\|$. The result is summarized in the following;

Theorem 2.1 *An ϵ-approximate path between two arbitray query points s and t on $\mathcal{P}$ can be computed, provided $t \bigcap \tau_s = \emptyset$.*

The preprocessing step of the above algorithm can be parallelized by the known techniques, since the geometric shortest path is transformed to an equivalent graph problem.

References

[1] P.K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan, "Approximating Shortest Paths on a Convex Polytope in Three Dimensions", submitted to *J. ACM*, 1996.

[2] L. Aleksandrov, M. Lanthier, A. Maheshwari and J.-R. Sack, "An ϵ-Approximation Algorithm for Weighted Shortest Paths on Polyhedral Surfaces", *Technical Report*, Carleton University, December 1997.

[3] A. Baltsan and M. Sharir, "On the Shortest Paths Between Two Convex Polyhedra", *J. ACM*, **35**, 1988, pp. 267-287.

[4] J. Canny and J. H. Reif, "New Lower Bound Techniques for Robot Motion Planning Problems",*28th IEEE Symp. on Foundations of Computer Science*, 1987, pp. 49-60.

[5] E. Cohen, "Efficient Parallel Shortest-Paths in Digraphs with a Separator Decomposition", SPAA'93, 1993, pp. 57-67.

[6] E. Dekel, D. Nassimi, and S. Sahni, "Parallel matrix and graph algorithms", *SIAM Journal of Computing*, Vol. 10, Mo. 4, pp. 657-675, 1981.

[7] S. Har-Peled, M. Sharir, and K.R. Varadarajan, "Approximating Shortest Paths on a Convex Polytope in Three Dimensions", *Proc. 12th Annual Symp. on Computational Geometry*, Philadelphia, PA, 1996, pp. 329-338.

[8] J. Hershberger and S. Suri, "Practical Methods for Approximating Shortest Paths on a Convex Polytope in $\Re^3$", *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995, pp. 447-456.

[9] J. Jenq and S. Sahni, "All pairs shortest path in sparse networks", *Journal of the ACM*, Vol. 24, No. 1, pp. 1-13, 1977.

[10] M. Lanthier, A. Maheshwari and J.-R. Sack, "Approximating Weighted Shortest Paths on Polyhedral Surfaces", *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, 1997, pp. 274-283.

[11] F.T. Leighton, "Introduction to Parallel Algorithms and Architectures", Morgan Kaufmann Publishers, San Mateo, USA, 1992.

[12] C. Mata and J. Mitchell, "A New Algorithm for Computing Shortest Paths in Weighted Planar Subdivisions", *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, 1997, pp. 264-273.

[13] J.S.B. Mitchell, "Shortest Paths and Networks", *Handbook of Discrete and Computational Geometry*, J. Goodman and J. O'Rourke Eds., CRC Press LLC, Chapter 24, 1997, pp. 445-466.

[14] J.S.B. Mitchell, "Geometric Shortest Paths and Network Optimization", *Handbook on Computational Geometry*, J.-R. Sack and J. Urrutia Eds., Elsevier Science B.V., 1998.

[15] R.C. Paige and C.P. Kruskal, "Parallel algorithms for shortest path problems", in *Proceedings of 1989 International Conference on Parallel Processing*, pp. 14-19, 1989.

[16] C.H. Papadimitriou, "An Algorithm for Shortest Path Motion in Three Dimensions", *IPL*, **20**, 1985, pp. 259-263.

[17] C. Savage, "Parallel Algorithms for Graph Theoretic Problems", Ph.D. thesis Mathematics Department, University of Illinois, Urbana, IL, 1977.

[18] M. Sharir, “On Shortest Paths Amidst Convex Polyhedra”, *SIAM Journal of Computing*, **16**, 1987, pp. 561-572.