EPiC Series in Computing

Volume 98, 2024, Pages 56–60

Proceedings of 39th International Conference on Computers and Their Applications

Constant Storage for Computing Shortest Paths for a Polyhedron

Jindong Chen[1], Sraawya Chintala[2], and Yijie Han[3]

[1] Google Research, 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA.
jdchen@google.com
[2] University of Missouri, Kansas City, MO 64110, USA.
scx4f@umkc.edu
[3] University of Missouri, Kansas City, MO 64110, USA.
hanyij@umkc.edu

Abstract

We present a new scheme for storing shortest path information for a polyhedron. This scheme is obtained by applying the constant storage scheme of Han and Saxena [4] on the outward layout of Sharir and Schorr [8]. We achieve constant storage and $O(\log n+k)$ time for computing the shortest path from the source point to a query point on the polyhedron, where k is the number of polyhedron edges this shortest path passes through. This improves the result of Chen and Han [3] which uses $O(n \log n/d)$ storage and $O(d \log n/ \log d + k)$ time, where d is an adjustable parameter.

1 Introduction

The single source shortest path problem on the surface of a polyhedron has been studied by a number of researchers [2, 5, 6, 7, 8]. Initially Sharir and Schorr solved this problem for a convex polyhedron [8], their solution can be stored in $O(n^2)$ space such that, for a query point $p = (f, c)$, where f is the face point p lies on and c is the position of p on f, the shortest distance from the source to p can be computed in $O(\log n)$ time and the edge sequences of the shortest path can be computed in $O(\log n + k)$ time, where k is the number of edges of the polyhedron the shortest path passes through. Followed by this Mount achieved a result of $O(n \log n)$ storage and $O(\log n+k)$ time for computing the shortest path from the source point to a query point [7]. Mount showed that the problem of storing shortest path information can be treated separately from the problem of computing the shortest paths. What Mount observed is that the edges of the polyhedron, after certain processing, form a total order according to their "distance" from the source. Later Chen and Han showed that shortest path information for a polyhedron can be stored in $O(n \log n/ \log d)$ space which supports the computing of shortest path distance in $O(d \log n/ \log d)$ time and shortest paths in $O(d \log n/ \log d+k)$ time, where k is the number of polyhedron edges the shortest path passes through and $1 < d \leq n$ is an adjustable integer [3]. Sharir and Schorr [8] and Mount [6, 7] only studied algorithms for convex polyhedron. Chen and Han's algorithm [3] works for nonconvex polyhedron as well.

In this paper we show an algorithm that uses constant storage for computing the shortest path distance in $O(\log n)$ time and shortest path in $O(\log n + k)$ time. We use the constant storage algorithm of Han and Saxena [4] and apply to the outward layout of Sharir and Schorr [8].

The rest of the paper is organized as follows. In Section 2 we explain the two layouts of a convex polyhedron, one is due to Sharir and Schorr [8] which is called the outward layout [2, 3], the other is due to Chen and Han which is called the inward layout [2]. Note that Agarwal *et al.* discovered this inward layout independently [1] and called it star unfolding.

In Section 3 we present our main result, namely the scheme of storing the shortest path information in constant storage for computing the shortest path distance in $O(\log n)$ time and the shortest path in $O(\log n + k)$ time.

2 Inward and Outward Layouts

The planar layout of the surface of a convex polyhedron due to Sharir and Schorr is obtained by cutting all the ridge lines [8]. It is called the outward layout [2]. Ridge lines are the loci of ridge points while a ridge point is a point for which there are more than one shortest paths from the source point. After cutting the ridge lines the surface of the polyhedron can be laid out on a plane. This layout is a star shaped polygon with the edges consisting of ridge lines. The vertices of the layout polygon are either vertices of the polyhedron or the Voronoi vertices on the ridge lines after being transformed into the layout plane. If two ridge lines intersect each other at a vertex, they form a notch. An example of such a layout is illustrated in Figure 1.

There is another layout of the surface of a polyhedron due to Chen and Han [2]. This layout is called the inward layout [2]. This layout is obtained by cutting the shortest paths from the source point to the vertices of the polyhedron. This layout is also a star shaped polygon with edges consisting of the shortest paths from the source point to vertices of the polyhedron. The vertices of the polygon are either the vertices of the polyhedron (after being transformed into the plane) or the images of the source point (there are $O(n)$ points which are images of the source point). Note that when a vertex of the polyhedron is also a ridge point, the shortest paths to this vertex form closed curves. A notch is formed in the layout when a shortest path to a vertex is cut. An example of such a layout is shown in Figure 2.

The layout due to Sharir and Schorr is called the outward layout [2] (for shortest paths emanate from the center, *i.e.,* the source point, outwards toward the destinations) and the layout due to Chen and Han is called the inward layout [2] (where a shortest path from the source to a destination going inward into the interior of the layout polygon).

In a layout the vertices of the polyhedron except the images of the source point (which can be a vertex of the polyhedron as well) can be arranged in a circular order. In the outward layout this circular order has been shown by Sharir and Schorr [8] and Mount [6]. The same circular order used in the outward layout can be used in the inward layout to order the vertices of the polyhedron. This is because if we cut along the shortest paths to the vertices of the layout polygon in the outward layout (all these shortest paths are contained within the layout polygon) and then coalesce along the ridge lines of the layout polygon we obtain the inward layout. Observe that if we assume that the circular order in the outward layout is counter-clockwise then the circular order in the inward layout is clockwise.

Suppose we connect the adjacent vertices in the circular order by a straight line segment in the layouts (in case of a nonconvex polyhedron, they could be hyperbolas). These line segments form a closed curve. This closed curve decomposes each layout polygon into two parts, one contains the source (we shall call it the arctic) while the other does not contain the source

(we shall call it the antarctic). We shall call this closed curve the equator of the polyhedron with respect to the source point. Note that the equator is determined by the polyhedron and the source point. This equator can be used to devise a scheme for storing the shortest path information for a polyhedron as shown by Chen and Han [3].

The words arctic, antarctic and equator were used in [2, 3]. Chen and Han recalled that the use of words arctic and equator was not started from their papers [2, 3]. At the time they wrote papers [2, 3] they got the words arctic and equator from other researchers' paper(s) for describing the outward layout and then used these words in their papers [2, 3]. Because this information was neglected in [2, 3] and thus we note it here. We have tried to identify the paper(s) that introduced words arctic and equator but have not solved this issue yet before the deadline for us to uploading the final version of this paper to CATA'2024. We will solve this issue before we submit this paper for journal publication.

Chen and Han are basically sure that the word antarctic was added by them for describing the inward layout [2, 3] in addition to the words arctic and equator used before their papers for describing the outward layout.

The words outward layout and inward layout are from Chen and Han and were first used in [2, 3].

The vertices of the outward layout polygon are vertices of the polyhedron and the vertices of the Voronoi diagram. There are a total of $O(n)$ vertices. We connect each of these vertices with the source by the shortest paths to the source. We thus obtain $O(n)$ ordered "triangles" by the circular order as shown in Figure 3.

3 Storing Shortest Path Information in Constant Storage for Computing Shortest Paths

We now make use of the outward layout by Sharir and Schor as in Figure 1 to present the scheme for reducing storage usage for shortest path information. Han and Saxena [4] showed that n real numbers can be stored in 5 real numbers, and each original real number can be fetched in $O(\log n)$ time. Once we obtained the outward layout[8] with triangulation. We initially store the angles made by these $O(n)$ triangle edges emanating from the source point in order. They would take $O(n)$ storage and we pack them into 5 real numbers as in [4]. These angles would be stored in the form of a tree compacted into 5 real numbers. When we compute the shortest distance for a query point p, we find the triangle of the subdivision p lies in by using binary search of the angle of the ray from the source point to p against the angles of the edges of triangles emanating from the source point. Thus, this takes $O(\log n)$ time. Because the angles of the edges of the triangles are sorted and arranged at the leaves of a binary tree and then packed into 5 real numbers [4] and thus we can decompose the 5 real numbers at the root of the binary tree and downwards along a tree path from the root to a leaf and thus find the triangle T the query point p locates in $O(\log n)$ time.

The edge segments of the polyhedron intersects T was sorted by the distance of the end point of these edge segments on one edge of the T that emanates from the source point and then arranged at the leaves of a binary tree E, and then packed into 5 real numbers [4]. Then these information of triangles are packed into 5 real numbers as shown in the previous paragraph.

We first use $O(\log n)$ time to identify triangle T where the query point p locates. We then find the value of k which is the number edge segments of the polyhedron the shortest path from the source to the query point passes through. This can be done using binary search on E and it takes $O(\log n)$ time.

We only need to decompose out the first k edge segments that the shortest path from the source point to the query point passes through from E. These k edge segments are at the left-most of leaves of E. We first identify the lowest common ancestor A of these k edge segments in E. If the number of leaves of the subtree rooted at A is L then $k \geq L/2$. Thus we decompose all the leaves of the subtree rooted at A and this should take $O(\log n + k)$ time. Thereafter we find the intersections of the shortest path from the source point to the query point with these k edges in $O(k)$ time.

Main Theorem: A polyhedron with n points can be stored in constant storage to support $O(\log n)$ time for computing the shortest distance from the source to a query point and $O(\log n + k)$ time for computing the shortest path from the source to this query point.

4 Conclusions

We have presented a new scheme for storing shortest paths information for a polyhedron in constant storage to support $O(\log n)$ time for computing the shortest distance from the source to a query point p and $O(\log n + k)$ time for computing the shortest path from the source to p, where k is the number of edges of the polyhedron the shortest path passes through. We are studying whether the $\log n$ factor in the time can be reduced further but it looks like it is not trivial.

References

[1] P. Agarwal, B. Aronov, J. O'Rourke, and C. Schevon. Star unfolding of a polytope with applications. SIAM Journal on Computing, Vol. 26, No. 6, pp. 1689 - 1713, 1997.

[2] J. Chen and Y. Han. Shortest paths on a polyhedron. Proc. of 1990 ACM Symposium on Computational Geometry, 360-369. This paper was later published on International Journal of Computational Geometry and Applications as:
J. Chen and Y. Han. Shortest paths on a polyhedron, Part I: Computing shortest paths. International Journal of Computational Geometry and Applications, Vol. 6, No. 2, pp. 127-144, June 1996.

[3] J. Chen and Y. Han. Storing shortest paths for a polyhedron. Proceedings 1991 International Conference on Computing and Information, May 1991). Lecture Notes in Computer Science 497, 169-180.

[4] Y. Han and S. Saxena. Storage in computational geometry. arXiv:2302.11821, 2023.

[5] D. M. Mount J. S. B. Mitchell and C. H. Papadimitriou. The discrete geodesic problem. SIAM J. Comput., Vol. 16, No. 4,647-668, Aug. 1987.

[6] D. M. Mount. On finding shortest paths on convex polyhedra. Tech. Report 1496, Dept. of Computer Sci.,Univ. of Maryland, Baltimore, MD, 1985.

[7] D. M. Mount. Storing the subdivision of a polyhedral surface. Proc. 2rd ACM Symposium on Computational Geometry, Yorktown Heights, NY, June 2-4, 1986.

[8] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. SIAM J. Comput., 15(1986), pp. 193-215., 1986.

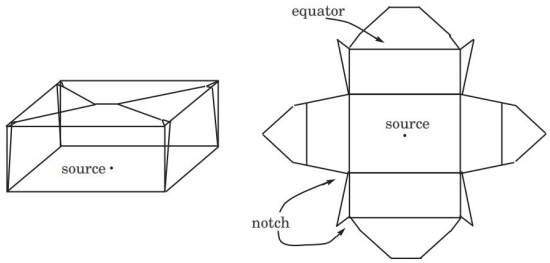

Figure 1: A polygon and its outward layout

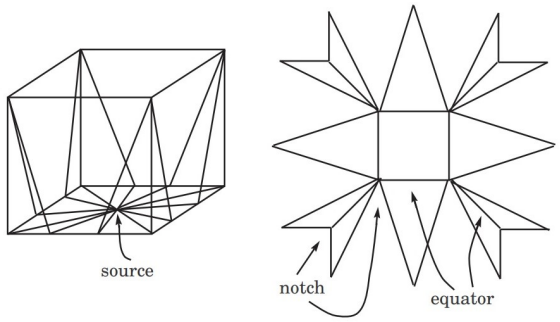

Figure 2: A polygon and its inward layout

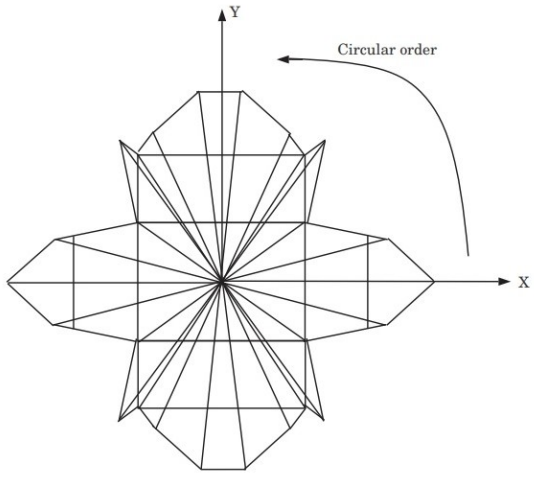

Figure 3: Triangulation and the circular order