축에 평행한 도로들이 놓여 있을 때의 L_1 최단 경로

배 상 원[1]*, 김 재 훈[2], 좌 경 룡[1]
[1]한국과학기술원 전자전산학과 전산학전공
{swbae*, kychwa}@tclab.kaist.ac.kr
[2]부산외국어대학교 컴퓨터공학과
jhoon@pufs.ac.kr

L_1 Shortest Paths with Isothetic Roads

Sang Won Bae[1]*, Jae-Hoon Kim[2], Kyung-Yong Chwa[1]
[1]Div. of Computer Science, Dept. of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology
[2]Division of Computer Engineering, Pusan University of Foreign Studies

요 약

We present a nearly optimal ($O(\nu \min(\nu, n) n \log n)$ time and $O(n)$ space) algorithm that constructs a shortest path map with n isothetic roads of speed ν under the L_1 metric. The algorithm uses the continuous Dijkstra method and its efficiency is based on a new geometric insight; the minimum in-degree of any nearest neighbor graph for points with roads of speed ν is $\Theta(\nu \min(\nu, n))$, which is first shown in this paper. Also, this algorithm naturally extends to the multi-source case so that the Voronoi diagram for m sites can be computed in $O(\nu \min(\nu, n)(n+m)\log(n+m))$ time and $O(n+m)$ space, which is also nearly optimal.

1 Introduction

A *transportation network* models a facility for faster movement, such as a highway network and a subway or bus network, and consists of *roads* which one can move along at a certain fixed speed. Furthermore, it is assumed that one can access or leave a road through any point on the road. Thus, in this situation, finding a shortest (or quickest) path between two points is a very fascinating and important problem to study not only in theory but also in practice.

As previous results, Aichholzer, Aurenhammer, and Palop [1] presented an $O(n^2 \log n)$ time and $O(n)$ space algorithm for constructing a shortest path map (SPM) for a given source and show that the SPM has linear size, and Bae and Chwa [2] obtained the same bound independently and a different one ($O(n^2)$ time/space). Most recently, an $O(n \log^5 n \log\log n)$ time and $O(n \log^5 n)$ space algorithm was presented by Görke and Wolff [5].

Following is our main result.

Theorem 1. *An L_1 shortest path map with n isothetic roads with speed ν in the plane can be computed in $O(\nu n \min(\nu, n) \log n)$ time and in $O(n)$ space.*

This time bound follows from the fact that the minimum in-degree of any nearest neighbor graph for points with roads of speed ν is $\Theta(\nu \min(\nu, n))$, which is first shown in this paper (but the proof is omitted). Since the speed ν is regarded as a constant, the time bound can be argued to be $O(n \log n)$. The resulting SPM is a polygonal subdivision of linear size and allows us to find the length of the shortest path to a query point in $O(\log n)$ time by point location and to report a shortest path in $O(\log n + k)$ time, where k is the complexity of the reported path.

2 Preliminaries

A *transportation network* on the L_1 plane is represented as a planar straight-line graph $G(V, E)$ such that any edge $e \in E$ has its supporting *speed* $v(e) > 1$. An edge in E is often called a *road* and a vertex in V a *node*. A transportation network together with its underlying metric induces a new metric, called a *transportation distance* [2,3]. In this paper, we deal with transportation networks of n isothetic roads with $v(e) = \nu$ for every road $e \in E$.

A *needle* is a generalized Voronoi site proposed by Bae and Chwa [3] for easy explanation of the geometry induced by transportation networks. A needle p can be represented by a 4-tuple $(p_1(p), p_2(p), t_1(p), t_2(p))$ with $t_2(p) \geq t_1(p) \geq 0$, where $p_1(p), p_2(p)$ are two endpoints and $t_1(p), t_2(p)$ are additive weights of the two endpoints, respectively. For convenient references to a needle, we define some terms associated with a needle: We let $s(p)$ be the segment $\overline{p_1(p)p_2(p)}$ and $v(p)$ the *speed* of p defined as $d_1(p_2(p), p_1(p))/(t_2(p) - t_1(p))$.

The L_1 distance from any point x to a needle p is measured as $d_1(x,p) = \min_{y \in s(p)}\{d_1(x,y) + w_p(y)\}$, where $w_p(y)$ is the weight assigned to y on $s(p)$, given as $w_p(y) = t_1(p) + d_1(y, p_1(p))/v(p)$, for all $y \in s(p)$.

It has been shown that the SPM in our setting has a linear size [1] and further can be represented as a Voronoi diagram for $O(n)$ needles, which can be computed in an optimal time and space [2].

We apply the continuous Dijkstra paradigm to obtain a shortest path tree (SPT). The framework of our algorithm is not so different from that of Mitchell [7] but most of specific details are quite distinguishable. The *continuous Dijkstra method* simulates the wavefront propagation by tracking effects of the *pseudo-wavefront* that is easy to maintain but sufficient to express the *true* wavefront.

In our case, the pseudo-wavefront is represented as a set of straight line segments, called *wavelets*. A wavelet ω is an arc of a "circle" centered at *root* $r(\omega)$.[1] Note that the root $r(\omega)$ of a wavelet ω can be a needle along a certain road. Each wavelet ω has a left and a right *track*, denoted by $\alpha(\omega)$ and $\beta(\omega)$, respectively, and expands with its endpoints sliding along its tracks. Each track is either a portion of an isothetic line or a portion of a bisecting curve between two certain roots. A bisector $B(r, r')$ between two roots r and r' is a piecewise linear and can be computed in constant time, even if the two roots are needles in general.

On the L_1 plane, the wavelets are basically inclined at angle either 45° or 135°. However, a road e inclined at angle θ makes wavelets inclined at angles $\theta \pm \tan^{-1} 1/\nu(e)$, where $\nu(e)$ is the speed of e [1]. We shall denote the set of such angles by A_e. Since we deal only with isothetic roads of speed ν, either $A_e = \{\tan^{-1} 1/\nu, -\tan^{-1} 1/\nu\}$ or $A_e = \{\pi/2 + \tan^{-1} 1/\nu, \pi/2 - \tan^{-1} 1/\nu\}$. We thus have only 6 angles for inclination of wavelets.

After running the continuous Dijkstra method, we obtain the SPT, the vertex-labeled tree rooted at the given source s. In our case, a vertex of the SPT is either a node incident to a road (in V) or the first intersection point of the first crossing road with an isothetic ray from a node, or just s. We let V' be the set of such vertices including V and s. We observe that V' has a linear number of vertices and is sufficient to detect and maintain topological and geometric changes of wavelets by the following lemma based on the lemma for primitive paths [2] and other previous results [1].

Lemma 2. *For any two points s and t, there exists an L_1 shortest path $P = (s = v_0, \cdots, v_k = t)$ such that $v_i \in V'$ for $i = 0, 1, \cdots, k-1$.*

[1]Here, a "circle" means a set of all the equidistant points from a given center, generally being a set of points or a needle.

3 The Algorithm

Our algorithm works with two phases: we compute an SPT rooted at s by applying the continuous Dijkstra method and then construct an SPM from the SPT.

During the first phase, to apply the continuous Dijkstra method, we maintain the set of active wavelets. Each wavelet is instantiated and terminated at an event with a certain value of δ. An event is associated with each wavelet and has a corresponding *event point* and *event distance*. We store the wavelets with their associated information (such as its left/right tracks $\alpha(\omega)$ and $\beta(\omega)$ including their closure point if any, its root $r(\omega)$, and its left/right neighbors $L(\omega)$ and $R(\omega)$ if any) in a priority queue, called the *event queue*, indexed by event distance. We have two kinds of events: *Closure events* occur when two tracks of a wavelet meet at a closure point p, and cause a wavelet disappearing. *Vertex events* occur when a wavelet, either its interior or its endpoint along a track, hits a vertex in V'.

Every time we instantiate or modify a wavelet, we determine the corresponding event point and event distance by taking the minimum of the following: the distance to its closure point if any, and the distance to the first vertex $v \in V'$ that is encountered as the wavelet expands. The second one can be computed via segment dragging query; this is originally considered by Chazelle [4] and its variations and applications for handling wavelets under the L_1 metric are well described in Mitchell [6]. Also note that even under non-degeneracy a vertex event and a closure event can occur at the same place and event distance. We break this type of ties by "closure-first" rule.

Each vertex $v \in V'$ has a label $\ell(v)$ and initially $\ell(v) = \infty$. After a vertex event on v at event distance δ, v has a finite label $\ell(v) = \delta < \infty$ and its predecessor $pre(v)$ in the SPT. In this way, the recent partial shortest path tree SPT(δ) can be maintained.

We also maintain the SPM(δ) *triangulation*, a decomposition of multiple copies of the plane, called the *sweep space* $S(\delta)$, as Mitchell [7] did. We call the procedure *Clip-and-Merge* whenever we detect an overlapping portion of the swept region. *Clip-and-Merge* resolves overlaps in the projection of the sweep space onto the plane, and operates in two phases; the first phase finds a "seed" point by exploring the SPM(δ) triangulation and the seed will be used in the second phase, called *Trace-Bisector*. The procedure *Trace-Bisector* traces out a merge curve between the overlapped portion of the swept region with a given seed point, and thus merges two overlapped regions into one "sheet". We detect an overlapping of the swept region when an already labeled vertex is hit again by a wavelet. We refer to Mitchell [7] about details for procedures *Clip-and-Merge* and *Trace-Bisector*.

3.1 Computing the SPT

First, we compute V' as noted earlier and then preprocess V' for segment dragging queries [4,6]. Then, we initialize the event queue to consist of these four wavelets at the source s. While the event queue is not empty, we extract the upcoming event from the front of the queue and process the event in a natural way according to its type. Here, we will not state details for event handling procedures due to lack of space, but we briefly summarize the procedures below:

Closure Event If the current event is a closure event, the associated wavelet has just been degenerated to a point as the two tracks meet at the point. Thus, we do remove the wavelet from the set of active wavelets and merge up its neighbor wavelets. There are two cases: If the two neighbor wavelets have doubly swept over a common portion, we call the subroutine *Trace-Bisector*. Otherwise, we update tracks of the neighbor wavelets appropriately.

Vertex Event When a wavelet ω hits a vertex $v \in V'$, a vertex event happens. If $l(v) < \infty$, another wavelet has hit v already, which means that the current pseudo-wavefront self-overlaps. Thus, we call the subroutine *Clip-and-Merge* in this case. If $l(v) = \infty$, we label v with the current event distance and set $pre(v)$ as the root $r(\omega)$ of ω. Then, we create new wavelets or modify existing ones accordingly. We have several cases and subcases here, but we will not state each of them here. Our brief strategy is as follows: Changes and updates happen only locally at v and neighboring wavelets of ω. Wavelets have inclination only of one of 6 angles as mentioned, and thus the critical task is computing proper tracks for new or existing wavelets, which can be done by searching a constant number of isothetic rays and bisecting curves between roots of neighboring wavelets.

3.2 Building the SPM from the SPT

The SPT obtained in the first phase consists of labeled vertices and directed links among the vertices along the roads. From this information, we can build a set $\mathcal{N}$ of *needles* such that the Voronoi diagram $\mathcal{V}(\mathcal{N})$ for $\mathcal{N}$ coincides with the SPM. This has been already argued in earlier results [1,3]. For needles, we refer to Bae and Chwa [3].

More precisely, we construct $\mathcal{N}$ as follows: for every directed link pq of the SPT, the needle r belongs to $\mathcal{N}$ only if $\overline{pq}$ is along a road, where $p_1(r) = p, p_2(r) = q, t_1(r) = \ell(p)$, and $t_2(r) = \ell(q)$. We also observe that $\mathcal{N}$ is pairwise non-piercing, since all wavelets come into roads through a vertex in V'. This implies that $\mathcal{V}(\mathcal{N})$ can be computed in optimal time and space [2]. Consequently, we can build the SPM from the SPT in $O(n \log n)$ time and $O(n)$ space.

4 The Complexity

Mitchell [7] observed in his results that the maximum in-degree of any nearest neighbor graph among obstacles under the Euclidean metric is 6. This observation provides the efficiency of his algorithm since 7 or more roots around one vertex always cause at least one clipping; as a result, any point in the plane can be swept over at most 6 times by the wavelets. An analogous argument can be naturally applied also to our situation.

Lemma 3. *Let Δ be the maximum in-degree of any directed nearest neighbor graph under the transportation metric induced by a transportation network and the L_1 metric. Then, any point in the plane can be swept over at most Δ times during the algorithm.*

Lemma 4. $\Delta = \Theta(\nu \min(\nu, n))$.

The above lemmas help us bound the number of events processed during the algorithm.

Lemma 5. *The algorithm processes at most $O(\nu n \min(\nu, n))$ events.*

Each event, before occurring, may involve a segment dragging query among the vertices in V', which costs $O(\log n)$ time, $O(n \log n)$ preprocessing time, and $O(n)$ space. Finally, we conclude the main theorem.

References

[1] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest paths, straight skeletons, and the city voronoi diagram. In *Proc. 18th SoCG*, pages 151–159, 2002.

[2] S. W. Bae and K.-Y. Chwa. Shortest paths and voronoi diagrams with transportation networks under general distances. In *Proc. of 16th ISAAC*, 2005. to appear.

[3] S. W. Bae and K.-Y. Chwa. Voronoi diagrams with a transportation network on the euclidean plane. Technical report, KAIST, 2005. A preliminary version appeared in proceedings of ISAAC 2004.

[4] B. Chazelle. An algorithm for segment dragging and its implementation. *Algorithmica*, 3:205–221, 1988.

[5] R. Görke and A. Wolff. Computing the city voronoi diagram faster. In *Proc. 21st EuroCG*, pages 155–158, 2005.

[6] J. S. B. Mitchell. l_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8:55–88, 1992.

[7] J. S. B. Mitchell. Shortest paths among obstacles in the plane. *IJCGA*, 6(3):309–331, 1996.