Computing Transportation Voronoi Diagrams in Optimal Time

Yaron Ostrovsky-Berman *

Abstract

We present the first time-optimal algorithm for computing the Voronoi Diagram under the metric induced by a transportation network with discrete entry and exit points. For input with n sites, k stations, and e transportation lines, the algorithm computes the Voronoi Diagram in $O\big((n+k)\log(n+k)+e\big)$ time.

1 Introduction and related work

Shortest Path Maps (SPM) and Voronoi Diagrams are well known geometric tools for answering distance related queries. The SPM is a subdivision of space, which allows finding the shortest path from a source point to a query point. The Voronoi Diagram is a subdivision of space which allows finding the site closest to a query point (multiple sources). These tools are useful in Geographic Information Systems, where the realism of the results depends on the underlying terrain model.

There have been many attempts to model the complexity of real world terrain with a simple mathematical model. The most general of these is the weighted region model [6] in which the plane is divided into regions with weights corresponding to the difficulty of crossing the terrain. There is no efficient algorithm for finding shortest paths in this general model without approximating the solution. One important special case is shortest paths amidst polygonal obstacles in the plane, in which the obstacles have infinite weight and the free space has unit weight. Mitchell [5] gave the first sub-quadratic solution, which runs in $O(n)$ space and $O(n^{3/2+\epsilon})$ time. Hershberger and Suri [4] presented the first optimal time algorithm, with $O(n\log n)$ space and time complexity. Both employ the continuous Dijkstra paradigm. Abellanas et al. [1] survey recent results obtained for models of urban environments.

In this paper, we model public transportation networks which provide time saving routes with discrete entry and exit points. We describe the network by an undirected graph with positive edge weights proportional to travel time, and assume the entire plane is accessible by foot. This type of transportation network cannot be modelled with weighted regions because it allows crossing the transportation line at no time cost, but restricts entrance to and exit from the network to the fixed stations. The proposed model was introduced by Münch in his PhD thesis [7], which studies a network of airlifts over the Euclidean plane. The algorithm for computing the Voronoi Diagram in the induced airlift metric was briefly discussed in Aicholzer et al. [2] and later expanded by Palop in her PhD thesis [9]. The algorithm completes the network graph by assigning Euclidean weights to disconnected stations, and applies the discrete Dijkstra algorithm to compute the station weights. The station weights are used as input to the Additively Weighted Voronoi Diagram (AWVD) algorithm, from which the desired subdivision is obtained. The complexity of the algorithm is $O(k^2+n)$ space and $O\big(k^2+(n+k)\log(n+k)\big)$ time, where n is the number of sites and k is the number of stations. The authors also raised the question of whether there exists a matching lower bound for the time complexity.

Our previous work [8] improves upon this result by computing the station weights with an input sensitive algorithm having the same worst case time complexity and $O(n+k\log k+e)$ space complexity, where e is the number of transportation lines. The algorithm has provably better time bounds under realistic regularity assumptions on the input.

In this paper, we describe the properties of the metric induced by the transportation network, and settle the question of the time complexity by combining the continuous Dijkstra method with the reduction to AWVD to obtain optimal $O(k\log k+e)$ and $O((n+k)\log(n+k)+e)$ time algorithms for the SPM and the Voronoi Diagram, respectively.

2 Definitions and properties

Let $T\subset\mathbb{R}^2$ denote a set of station positions in the plane, and let E denote the connection relation between the stations, such that $(t_1,t_2)\in E$ if and only if $t_1,t_2\in T$ and the stations are connected by a line. Denote the positive weight of a transportation line $(t_1,t_2)\in E$ by $w(t_1,t_2)$ and define it to be infinity when the points are not connected stations. The graph $\langle T,E,w\rangle$ describes the transportation network. For the Voronoi diagram problem, the set $S\subset\mathbb{R}^2$ denotes the sites. Let $d(p,q)$ denote the Euclidean distance between two points in the plane. The transportation distance between two points is defined re-

*School of Engineering and Computer Science, The Hebrew University of Jerusalem, yaronber@cs.huji.ac.il

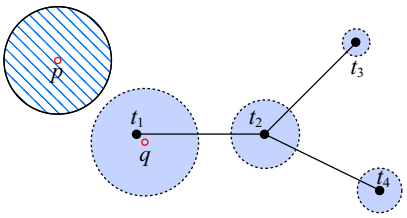

Figure 1: The unit disc. Stations t_1, t_2, t_3, t_4 are connected by solid transportation lines. The hatched disc is the unit disc of p, and the shaded discs comprise the unit disc of q.

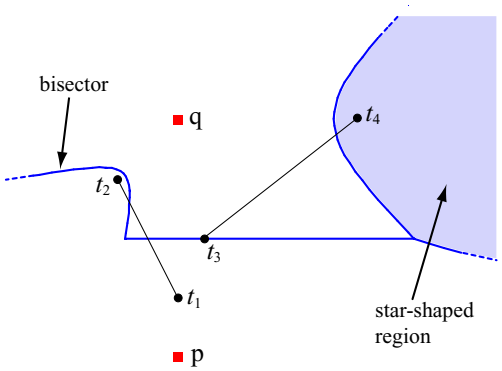

Figure 2: The bisector of two points. Solid circles are stations, the thin solid lines connecting them are transportation lines. The weight of the transportation lines is tenth of the Euclidean distance between the stations. The thick curve is the bisector of p and q. The shaded region is part of the bisector because $d_T(p, t_4) = d_T(q, t_4)$.

cursively as follows:

$$d_T(p,q) = \min\{d(p,q),\ w(p,q),\ \min_{t\in T}\{d_T(p,t) + d_T(t,q)\}\}$$

The unit disc of a point p in the metric induced by the transportation distance, $B_T(p) = \{x | d_T(p,x) \leq 1\}$, depends on the proximity of p to the network (Figure 1). This leads to the following properties:

1. The Voronoi cell of a site can have several connected components.

2. The transportation distance bisector of two points p and q, $b_T(p,q) = \{x | d_T(p,x) = d_T(q,x)\}$, consists of line segments (points whose shortest path to p and q is a straight line), hyperbolic arcs (points whose shortest path to p or q uses the transportation network), and star shaped regions bounded by line and hyperbolic segments (points whose shortest paths to p and q begins with a shared network station). Figure 2 illustrates these cases.

As shown in [8, 9], the shortest path map is constructed by computing the transportation distance from the source to all the stations, then using this distance as a negative weight in the AWVD of the source and the stations. In the AWVD, the distance between a point p and a site s with weight w_s is $d_{AWVD}(p,s) = d(p,s) - w_s$. Fortune [3] showed how to construct the AWVD in $O(n \log n)$ time. The transportation Voronoi diagram is constructed similarly to the SPM, only now the weights are assigned according to the *closest* site in transportation distance.

3 The continuous Dijkstra method

We now show how to modify the continuous Dijkstra method [4, 5] to handle the transportation distance metric instead of the geodesic distance induced by polygonal obstacles. For simplicity of presentation, we address the single source problem only. The extension to multiple sources is straightforward.

The continuous Dijkstra method simulates the propagation of a wavefront from the source point. At simulation time δ, the wavefront is the locus of points with transportation distance δ from the source. The wavefront is comprised of wavelets, which are points having the same predecessor in the shortest path to the source. In its purest form, the simulation advances between discrete events of the following type: 1. A wavelet collides with a station; 2. A wavelet collides with another wavelet; 3. A wavelet is engulfed by neighboring wavelets and disappears. In practice, identifying these events as they occur is difficult, and the method of [5] does not attempt to do so. Instead it ensures that when the events are discovered, there is only local work to do to fix the discrepancy. The method of [4] uses a clever subdivision of the plane that guides the wavefront propagation, and computes "approximate wavefronts" at the edges of this subdivision, which are used in the final stage to construct the shortest path map.

In the geodesic problem, an event of type 1 is a collision of the wavelet with an obstacle vertex. This vertex becomes the generator of a new wavelet. In the transportation metric, the station encountered is not a new generator, but its neighbors in the transportation network possibly become generators in the future (after a time equal to the connection time, which is the edge weight). We call the station that schedules new generators a *transporter*. For each station t we store the time $g(t)$ in which t becomes a generator (initialized to infinity), and the set $p(t)$ of predecessors of t in the shortest path to the source s (the cardinality of $p(t)$ is greater than one when the path is not unique). Suppose a wavelet hits station t at time δ. The simulation algorithm performs the update operations required by an obstacle vertex collision event, but instead of creating a new generator at t, it first checks if $\delta \leq g(t)$, and if so proceeds as described

Procedure propagate-transporter(t, δ)
1. set $g(t) = \delta$
2. **for-each** r with $(t, r) \in E$:
2a **if** $\delta + w(t, r) < g(r)$ **then**
- set $g(r) = \delta + w(t, r)$ and set $p(r) = t$
- insert new-generator-event(r) into event queue at time $g(r)$
2b **else-if** $\delta + w(t, r) = g(r)$ **then**
- set $p(r) = p(r) \cup t$
end for-each

Table 1: Wavefront propagation through the network.

in Table 1, possibly scheduling new generators at the neighboring stations.

When the simulation time reaches a new-generator-event(r), a new wavelet with generator r is created (since the algorithm does not clear future events for the same r in step 2a, the simulation simply continues if r was already processed). The new wavelet automatically triggers an event of type 1 for a collision with r. Events of type 2 and 3 are handled as in the geodesic problem.

Observation 1 *The wavefront propagation algorithms of [4, 5] both detect collisions with an obstacle vertex t and set $g(t)$ correctly before the vertex is used as a generator, and this property holds when network stations are treated as obstacle vertices.*

We now prove that this observation holds after the modification made in propagate-transporter.

Lemma 1 *For every station t in the transportation graph, the propagation algorithm sets $g(t) = d_T(s, t)$ before t is used as a generator or a transporter.*

Proof. By induction on the number of links in the shortest transportation distance path from s to t. If there is one link, then the shortest path is Euclidean, and according to Observation 1 a wavelet leaving s will encounter t and set $g(t)$ correctly. If the path is longer, then by induction, the predecessor p of t has $g(p) = d_T(s, p)$ set before it is used as a generator or a transporter. If $(p, t) \notin E$ then part of the shortest path uses the transportation network, and p is the terminating station. Thus p was scheduled to be a generator by its predecessor in the path (step 2 of propagate-transporter). Since $d_T(s, p) < d_T(s, t)$, a wavelet is generated by p which, according to Observation 1, will collide with t and set $g(t)$ correctly. If $(p, t) \in E$ then p is a transporter and step 2 of propagate-transporter correctly sets $g(t) = g(p) + w(p, t) = d_T(s, t)$. □

Lemma 1 proves the correctness of the modification to the wavefront simulation. The rest of the algorithm is unchanged, thus when it terminates, the values of

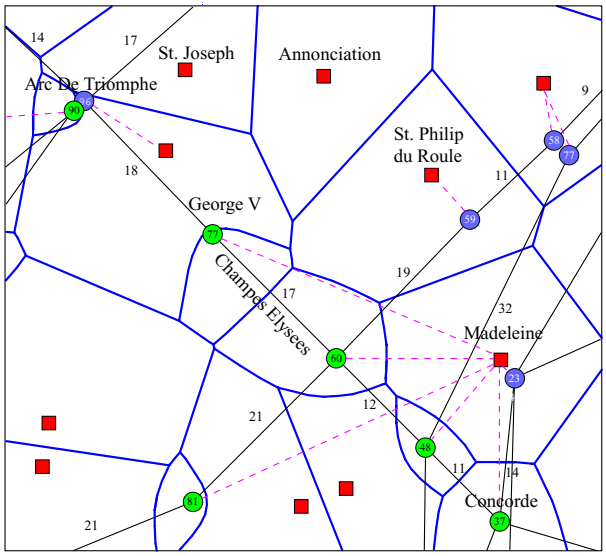

Figure 3: The transportation Voronoi diagram of the Paris Metro and churches. Solid discs are Metro stations, the number inside is the value of $d_T(s, t)$ for the closest church. Solid thin lines are the Metro lines, and numbers near their middle denote their weights. Solid squares are churches. A dashed line from a church to a station means that the church is closest to the station. Solid curves are the boundaries of the AWVD cells of the stations and the churches.

$g(t)$ equal $d_T(s, t)$ for all $t \in T$. As noted above, the AWVD of the source and the stations, with weights equal to the negative value of the transportation distance, gives the shortest path map of s. The predecessor information $p(t)$ is used backwards to determine the tree of shortest paths from s.

Figure 3 shows the transportation Voronoi diagram of part of the Paris Metro and nearby churches. The station weights were computed with our implementation of the reduction to AWVD in [8]. The AWVD was computed with the CGAL Apollonius graph class. Note that the Madeleine church controls more area because of its proximity to a Metro station. Its Voronoi cell has two connected components and consists of the the church's cell plus the cells of five nearby stations.

4 Complexity analysis

We now analyze the complexity of the algorithm, starting with the modification described in Section 3. In what follows n, k, and e stand for the number of sites, stations and transportation lines, respectively. We assume the event queue is implemented with a Fibonacci heap, which supports insertions and minimum extractions in amortized $O(1)$ and $O(\log n)$ time, respectively. Assuming the wavefronts are efficiently pruned (as is the case in [5, 4]), there are $O(1)$ events of type 1 per station. The modification

to the event handler performs $O(1)$ comparison operations and $O(1)$ insertions to the queue per neighbor of t in the graph. Because each edge is visited $O(1)$ times, the total added complexity of the modification is $O(e)$.

The complexity of the entire algorithm depends on the continuous Dijkstra method used. The algorithm of Mitchell [5] results in optimal $O(n+k+e)$ space complexity and $O\big((n+k)^{3/2+\epsilon}+e\big)$ time complexity, while the algorithm of Hershberger and Suri [4] results in $O\big((n+k)\log(n+k)+e\big)$ space and time complexity, which is time-optimal.

From a practical standpoint, we note that the algorithms presented in [4, 5] are complicated and have not been implemented. Our previous algorithm [8] ignores events of types 2 and 3 altogether, and instead prunes the propagation of wavelets by bounding the radius of their effect on the transportation distance. The bound equals the difference between the maximal and minimal transportation distance from a station to a site, and thus decreases monotonically as the algorithm advances, lowering the cost of each iteration. On classes of transportation networks such as clusters or uniform distribution of stations and sites, the complexity of this method is provably lower than the quadratic worst case, and experiments show that this is true of most networks, including random networks and sites.

5 Conclusion

We have presented the first time-optimal algorithm for computing the transportation Voronoi Diagram. The existence of an optimal space and time algorithm is an open problem related to geodesic distance Voronoi Diagram.

To make the transportation network more realistic, the model can be augmented by access, connection, and waiting times of the stations and lines. See [8] for details. Furthermore, the combination of the algorithm proposed here with the original algorithm for the geodesic problem can be used for solving shortest path problems in the presence of polygonal obstacles as well as a transportation network, enriching the urban environment model.

References

[1] M. Abellanas, F. Hurtado, and B. Palop. Transportation networks and Voronoi Diagrams. In *International Symposium on Voronoi Diagrams in Science and Engineering*, Tokyo, 2004.

[2] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest paths, straight skeletons, and the city Voronoi Diagram. In *Proc. of the 18th annual symposium on Computational Geometry*, pages 151–159, 2002.

[3] S. Fortune. A sweepline algorithm for Voronoi Diagrams. *Algorithmica*, 2(2):153–174, 1987.

[4] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.

[5] J. S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry and Applications*, pages 309–332, 1996.

[6] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, (38):18–73, 1991.

[7] O. Münch. *Das Voronoi-Diagramm in der Airline-Metrik: Untersuchung der strukturellen Eigenschaften und Veranschaulichung durch ein Java-Programm.* PhD thesis, FernUniversität Hagen, Fachbereich Informatik, 1998.

[8] Y. Ostrovsky-Berman. Transportation Voronoi Diagrams. Technical Report 2003-2, Leibniz Center for Research in Computer Sciences, 2003. URL: www.cs.huji.ac.il/~yaronber/

[9] B. Palop. *Algorithmic problems on proximity and location under metric constraints.* PhD thesis, U. Politécnica de Madrid, 2003.