Two-Site Voronoi Diagrams in Geographic Networks

Matthew T. Dickerson
Middlebury College
https://seguecommunity.middlebury.edu/sites/dickerso/

Michael T. Goodrich
University of California, Irvine
http://www.ics.uci.edu/~goodrich/

ABSTRACT

We provide an efficient algorithm for two-site Voronoi diagrams in geographic networks. A two-site Voronoi diagram labels each vertex in a geographic network with their two nearest neighbors, which is useful in many contexts.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

Keywords: geographic graphs, shortest paths, Voronoi diagrams.

1. INTRODUCTION

The Voronoi diagram is a fundamental geometric structure (e.g., see [3, 4, 15, 24]). Given a set K of n points in $\mathbf{R}^d$, called *sites*, the *Voronoi diagram* of K is defined to be the subdivision of $\mathbf{R}^d$ into cells, one for each point in K, where the Voronoi cell for point $p \in S$ is the loci of all points closer to p than to any other point in K. This simple definition, which is now more than 100 years old [13, 25], has been used in a host of different application domains, including astronomy (as a way of defining clusters), crystallography (as a way of defining growing regions from seeds), and robotics (as a way of defining safe zones for motion planning purposes).

When coupled with a data structure for performing point locations in $\mathbf{R}^d$, the Voronoi diagram provides a solution to Knuth's well-known *post office problem* [18]: given a set of n sites defining post offices, create a structure that allows us to identify, for each house, what is its nearest post office. Unfortunately, there is a major problem with this "solution."

The problem with using the Voronoi diagram, as defined above, as a solution to Knuth's post office problem is that it is inappropriately applying a geometric structure to a geographic problem. Given that the main purpose for knowing the closest post office to a given house is so that its inhabitants can find the nearest place to drive to in order to mail a package, it should be clear that post offices should not be viewed as points in $\mathbf{R}^d$. Post offices are points in a geographic network—the graph defined by the set of roads in a given geographic region—not points in $\mathbf{R}^d$. Therefore, if we desire a more authentic solution to Knuth's post office problem, we should be using a version of the Voronoi diagram that is defined for geographic graphs.

Graph-Theoretic Voronoi Diagrams. A *geographic network* is a graph $G = (V, E)$ that represents a transportation or flow network, where commodities or people are constrained to travel along the edges of that graph. Examples include road networks, flight networks, railroad networks, utility distribution grids, and sewer lines. We assume that the edges of a geographic network are assigned weights, which represent the cost, distance, or penalty of moving along that edge, or some combination of these and other factors, such as scenic or ecological value. The only requirement we make with respect to these weights is that they be non-negative and that the weight of traversing a path in G be the sum of the weights of the edges in that path. This allows us to define the distance, $d(v, w)$, between two vertices v and w in G as being the length of a shortest (i.e., minimum weight) path between v and w. (Note: we are restricting our attention to undirected geographic networks in this paper.)

Formally, we define a geographic network, $G = (V, E)$, to be a set V of *vertices*, a set E of *edges* (which are unordered pairs of distinct vertices), and a *weight function* $w : E \to \mathbf{R}^+$ mapping E to non-negative real numbers. Furthermore, we are given a subset $K \subset V$ of special vertices called *sites*. These are the "post offices" in Knuth's post office problem. Each site $v \in K$ is uniquely labeled with a natural number $n(v)$ from 0 to $|K| - 1$, so that we can refer to sites by number. The graph-theoretic *Voronoi diagram* [14, 21] of G is a labeling of each vertex w in V with the number, $n(v)$, of the vertex v in K that is closest to w. All the vertices with the same label, $n(v)$, are said to be in the *Voronoi region* for v. Intuitively, if a site v in K is considered a post office, then the Voronoi region for v consists of all the homes in v's zip code.

We use these numbers, labeling the vertices in K, to break ties in distances, which allows us to speak of unique closest sites in K for each vertex in V. That is, if we have two distinct sites $v, w \in S$ and a third vertex $x \in V$ such that $d(v, x) = d(w, x)$, then we say that x is closer to v if and only if $n(v) < n(w)$, and otherwise x is closer to y. For example, consider two distinct sites $v, w \in S$ with $n(v) = 0$ and $n(w) = 1$, and a third vertex $x \in V$, and with two edges (v, x) and (w, x) such that $d(v, x) = d(w, x)$. Then x is in the Voronoi region for site v.

Mehlhorn [21] shows that the graph-theoretic Voronoi diagram of a graph G, having n vertices and m edges, can be constructed in $O(n \log n + m)$ time, and a similar algorithm is given by Erwig [14]. At a high level, these algorithms basically are performing n simultaneous runs of Dijkstra's single-source shortest-path algorithm [12] (see also [10, 16]). In this paper, however, we are not

ACM GIS '08, November 5-7, 2008. Irvine, CA, USA

interested in these types of *single-site* Voronoi diagrams.

Two-Site Distance Functions. In a number of applications, we are interested in labeling the vertices of a geographic network, G, with more information than just their single nearest neighbor from the set of sites, K. We may instead wish to label each vertex v in G with the names of the two closest sites in K. For example, the sites in K may be fire stations, and we may wish to know the two closest fire stations for each house in our network, just in case there is a double-alarm fire at that location, and we need to call two fire stations to put it out. See Figure 1.

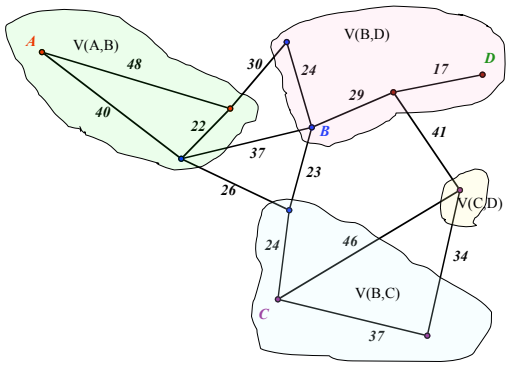

Figure 1: An example graph-theoretic Two-Site Voronoi diagram, under the distance metric where we minimize the Sum to two of the distinguished sites.

Related Prior Work. Unlike prior work on graph-theoretic Voronoi diagrams, there is a abundance of prior work for geometric Voronoi diagrams. It is beyond the scope of this paper to review all this work and its applications, however. Thus, rather that attempting to review all of this work, we refer the interested reader to any of the excellent survey articles on the subject (e.g., see [3, 4, 15, 24]) and we focus here on previous work on multi-site geometric Voronoi diagrams and on graph-theoretic Voronoi diagrams.

Lee [20] studies k-nearest neighbor Voronoi diagrams in the plane, which are also known as "order-C Voronoi diagrams." These structures define each region, for a site p, to be labeled with the C nearest sites to p. These structures can be constructed for a set of n points in the plane in $O(n^2 + C(n-C)\log^2 n)$ time [9]. Due to their computational complexity, however, order-C Voronoi diagrams have not been accepted as practical solutions to C-nearest neighbor queries. In ACM GIS 2007, Patroumpas *et al.* [22] study methods for performing C-nearest neighbor queries using an approximate order-C network Voronoi diagram of points in the plane, which has better performance than its exact counterpart.

2-site distance functions and their corresponding Voronoi diagrams were introduced by Barequet, Dickerson, and Drysdale [7] (see also [8] for a visualization of this structure). A two-site distance function is measured from a point to a pair of points. In Euclidean space, it is a function

$$D_f : \mathbf{R}^2 \times (\mathbf{R}^2 \times \mathbf{R}^2) \to \mathbf{R}$$

mapping a point p and a pair of points (v, w) to a non-negative real number. In a graph, it is a function mapping a point p on the graph (either a vertex or a point on an edge) to a pair of vertices (v, w)–usually sites. Two-site distance functions D_f are symmetric on the pair of points (v, w), though not necessarily on p, thus $D_f(p,(v,w)) = D_f(p,(w,v))$. (For some two site distant functions, $D_f(p,(v,w)) = D_f(v,(p,w))$ but this is incidental.) The Sum combination function, D_S, results in the same Voronoi diagram as the 2-nearest neighbor (order-2) Voronoi diagram, but the authors considered a number of other combination rules as well.

As we mentioned above, single-site graph-theoretic Voronoi diagrams were considered by Mehlhorn [21], who presented an algorithm running in $O(n \log n + m)$ time, and a similar algorithm was given by Erwig [14]. More recently, Aichholzer *et al.* [2] study a hybrid scheme that combines geometric distance with a rectilinear transportation network (like a subway), and Abellanas *et al.* [1] study a similar approach where the subway/highway is modeled as a straight line. Likewise, Bae and Chwa [5, 6] study hybrid schemes where distance is defined by a graph embedded in the plane and distance is defined by edge lengths.

As far as multi-site queries are concerned, Safar [23] studies k-nearest neighbor searching in road networks, but he does so using the first-order Voronoi diagram, rather than considering a multi-site Voronoi diagram for geographic networks. Likewise, Kolahdouzan and Shahabi [19] also take the approach of constructing a first-order Voronoi diagram and searching it to perform C-nearest neighbor queries. Instead, de Almeida and Güting [11] compute C-nearest neighbors on the fly using Dijkstra's algorithm. Alternatively, Hurtado *et al.* [17] study Voronoi diagrams for finding the nearest neighbor of a farthest color in a graph. None of these methods actually construct a two-site graph-theoretic Voronoi diagram, however.

Our Results. In this paper, we study two-site distance Voronoi diagrams on graphs. We address both the sum function D_S and the perimeter combination function D_P for defining these concepts. In particular, for a vertex p or a point p on an edge e and a pair of sites v, w, our two-site distance functions are defined as follows:

$$D_S(p,(v,w)) = d(p,v) + d(p,w)$$
$$D_P(p,(v,w)) = d(p,v) + d(p,w) + d(v,w)$$

Note that with two-site distance functions, we also have a similar rule for breaking ties in distances. In the case that $D(p,(v,w)) = D(p,(v',w'))$ for vertex p and sites v, w, v', w' and a two-site distance function D, we consider p closer to whichever of (v, w) and (v', w') has a smaller lexicographical pair of indices.

We show that two-site Voronoi diagrams in geographic networks can be constructed in $O(n \log n + m)$ time, under the Sum combination rule, for a graph with n vertices and m edges, independent of the number, k, of sites. Under the perimeter combination function, we show that such a diagram can be constructed in $O(km + kn(k + \log n))$ in the worst case.

2. CONSTRUCTING GRAPH-THEORETIC VORONOI DIAGRAMS

In this section, we review the approach of Mehlhorn [21] and Erwig [14] for constructing a (single-site) graph-theoretic Voronoi diagram of a graph G, having n vertices and m edges, which runs in $O(n \log n + m)$ time.

So suppose we are given a geographic network, $G = (V, E)$, together with a set of sites, $K \subseteq V$, and a non-negative weight function on the edges in E that define our notion of distance. The main idea for constructing a graph-theoretic Voronoi diagram for G is to imagine that we create a new vertex, a, called the *apex*,

which was originally not in V, and connect a to every site in K by a zero-weight edge. We then perform a single-source, shortest-path (SSSP) algorithm from a to every vertex in G, using an efficient implementation of Dijkstra's algorithm (e.g., see [10]). Intuitively, this algorithm grows the Voronoi region for each site out from its center, with the growth for all the sites occurring in parallel. Moreover, since all the Voronoi regions grow simultaneously and each region is contiguous and connected by a subgraph of the shortest-path tree from a, we can label vertices with the name of their Voronoi region as we go.

In more detail, the algorithm begins by labeling each vertex v in K with correct distance $D[v] = 0$ and every other vertex v in V with tentative distance $D[v] = +\infty$, and we add all these vertices to a priority queue, Q, using their D labels as their keys. In addition, for each vertex v in K, we label v with the name of its Voronoi region, $R(v)$, which in each case is clearly $R(v) = n(v)$. In each iteration, the algorithm removes a vertex v from Q with minimum D value, confirming its D label and R label as being correct. It then performs a *relaxation* for each edge (v, u), incident to v, by testing if $D[v] + w(v, u) < D[u]$. If this condition is true, then we set $D[u] = D[v] + w(v, u)$, updating this key for u in Q, and we set $R(u) = R(v)$, to indicate (tentatively) that, based on what we know so far, u and v should belong to the same Voronoi region. When the algorithm completes, each vertex will have its Voronoi region name confirmed, as well as the distance to the site for this region. Since each vertex is removed exactly once from Q and each key is decreased at most $O(m)$ times, the running time of this algorithm is $O(n \log n+m)$ if Q is implemented as a Fibonacci heap (e.g., see [10]). In addition, note that this algorithm "grows" out the Voronoi regions in increasing order by distance from the apex, a, and it automatically stops the growing of each Voronoi region as soon as it touches another region, since the vertices in an already completed region are (by induction) closer to the apex than the region we are growing.

3. TWO-SITE VORONOI DIAGRAMS AND THE SUM FUNCTION

As mentioned above, the two-site sum function Voronoi diagram is equivalent to the second order two-nearest neighbor Voronoi diagram, which identifies for each vertex v in our graph, G, the two nearest sites to v. We state and prove the equivalence of these two types of Voronoi diagrams in the following simple lemma, the proof of which holds for both Voronoi diagrams in the plane and on weighted undirected graphs.

LEMMA 1. *If v and w are the two closest sites to a vertex p in G, then the pair (v, w) minimizes $D_S(p, (v, w))$.*

PROOF. Suppose that v and w are the two closest sites to a vertex p in G, but that the pair (v, w) did not minimize $D_S(p, (v, w))$. Without loss of generality, let $d(p, v) \leq d(p, w)$. By our assumption, there exists a vertex x such that $d(p, x) < d(p, w)$. It follows immediately that $D_S(p, (v, x)) < D_S(p, (v, w))$ which is a contradiction. □

It follows that the two-site Voronoi diagram is equivalent to the two-nearest neighbor Voronoi diagram for a set of points in the plane or a graph. So we are ready to formally define our construction problem.

PROBLEM 1. *Given a graph $G = (V, K, E)$ of n vertices V, m edges E, and a subset $K \subset V$ of s special vertices called "sites", compute the 2-site Sum function Voronoi diagram of G; that is, label each vertex $v \in V$ with the closest pair of sites in K according to the 2-site Sum distance function.*

The Algorithm. Intuitively, our algorithm for constructing a two-site Voronoi diagram under the Sum combination rule is to perform a Dijkstra single-source shortest-path (SSSP) algorithm from each site, in parallel, but visit each vertex twice—once for each of the two closest sites to that vertex.

More specifically, we begin by labeling each vertex v in K with correct first-neighbor distance $D_1[v] = 0$ and every other vertex v in V with tentative first-neighbor distance $D_1[v] = +\infty$, and we add all these vertices to a priority queue, Q, using their D_1 labels as their keys. We also assign each vertex $v \in V$ (including each site in K) its tentative second-neighbor distance, $D_2[v] = +\infty$, but we don't yet use these values as keys for vertices in Q. In addition, for each vertex v in K, we label v with the name of its first-order Voronoi region, $R_1(v)$, which in each case is clearly $R_1(v) = n(v)$. In each iteration, the algorithm removes a vertex v from Q with minimum key. How we then do relaxations depends on whether this key is a D_1 or D_2 value.

- Case 1: The key for v is a D_1 value. In this case we confirm the D_1 and R_1 values for v, and we add v back into Q, but this time we use $D_2[v]$ as v's key. We then perform a *relaxation* for each edge (v, u), incident to v, according to the following test:

  Relaxation(v, u):
  if u has had its R_2 label confirmed **then**
  Return (for we are done with u).
  else if u has had its R_1 label confirmed **then**
  if $R_1(v) \neq R_1(u)$ and $D_1[v] + w(v, u) < D_2[u]$ **then**
  Set $D_2[u] = D_1[v] + w(v, u)$
  Set $R_2(u) = R_1(v)$
  if $D_2[v] + w(v, u) < D_2[u]$ **then**
  Set $D_2[u] = D_2[v] + w(v, u)$
  Set $R_2(u) = R_2(v)$.
  else
  if $R_1(v) \neq R_1(u)$ and $D_1[v] + w(v, u) < D_1[u]$ **then**
  Set $D_1[u] = D_1[v] + w(v, u)$
  Set $R_1(u) = R_1(v)$.

  In addition, if the D_1 or D_2 label for u changes, then we update this key for u in Q. Moreover, since we are confirming the D_1 and R_1 labels for v, in this case, we also do a reverse relaxation for each edge incident to v by calling **Relaxation**(u, v) on each one.

- Case 2: The key for v is a D_2 value. In this case we confirm the D_2 and R_2 values for v, and we do a relaxation for each edge (v, u), incident to v, as above (but with no reverse relaxations).

When the algorithm completes, each vertex will have its two-site Voronoi region names confirmed, as well as the distance to each of its two-nearest sites for this region.

The correctness of this algorithm follows from the correctness of the SSSP algorithm and from Lemma 1. The SSSP algorithm guarantees that vertices will be visited in increasing order of distance from the origin(s) of the search. Lemma 1 states that the closest two sites to v are also the closest pair according to the sum two-site distance function.

For the analysis of this algorithm, first note that no vertex will be visited more than twice, since each vertex is added to the queue, Q, twice—once for its first-order nearest neighbor and once for its second-order nearest neighbor. Moreover, once a vertex is added

to Q, its key value is only decreased until it is removed from Q. Thus, this algorithm requires $O(n \log n + m)$ time in the worst cast when Q is implemented using a Fibonacci heap, where n is the number of vertices in G and m is the number of edges. By the same reasoning, the priority queue Q won't grow larger than $O(n)$ during the algorithm and only a constant amount of information is stored at each vertex/edge, so the space required is $O(n)$.

4. TWO-SITE VORONOI DIAGRAMS WITH THE PERIMETER DISTANCE FUNCTION

Using the Sum function as the combination rule for a two-site graph-theoretic Voronoi diagram allowed us to label each vertex in G with its two nearest neighbors. Such a labeling is appropriate, for example, for fire stations or police stations, where we might want agents from both locations to travel to our home. If instead we want to leave our home, travel to two nearby sites, such as two grocery stores (e.g., because each has a limit on coffee purchases), and return home, then we will need a different combination rule than the Sum rule—we will need to use the Perimeter function. We can solve the two-site Voronoi diagram function by first computing the shortest paths from each site to every other vertex, which, with additional processing, also gives the distance from each vertex to each other site. Taking minimums from these distances, we can compute all distances between vertices and sites and between sites. Finally, we can compute for each vertex v the lengths of all the triangles formed by v and two sites, and determine the smallest such triangle. This algorithm runs in $O(km + kn(k + \log n))$ time for k sites in an n vertex graph with m edges.

5. REFERENCES

[1] M. Abellanas, F. Hurtado, V. Sacristán, C. Icking, L. Ma, R. Klein, E. Langetepe, and B. Palop. Voronoi diagram for services neighboring a highway. *Inf. Process. Lett.*, 86(5):283–288, 2003.

[2] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest paths, straight skeletons, and the city Voronoi diagram. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 151–159, New York, NY, USA, 2002. ACM.

[3] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, Sept. 1991.

[4] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[5] S. W. Bae and K.-Y. Chwa. Voronoi diagrams with a transportation network on the euclidean plane. In *Proc. Int. Symp. on Algorithms and Computation (ISAAC)*, volume 3341 of *LNCS*, pages 101–112. Springer, 2004.

[6] S. W. Bae and K.-Y. Chwa. Shortest paths and Voronoi diagrams with transportation networks under general distances. In *Proc. Int. Symp. on Algorithms and Computation (ISAAC)*, volume 3827 of *LNCS*, pages 1007–1018. Springer, 2005.

[7] G. Barequet, M. T. Dickerson, and R. L. S. Drysdale. 2-point site Voronoi diagrams. *Discrete Appl. Math.*, 122(1-3):37–54, 2002.

[8] G. Barequet, R. L. Scot, M. T. Dickerson, and D. S. Guertin. 2-point site Voronoi diagrams. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, pages 323–324, New York, NY, USA, 2001. ACM.

[9] B. Chazelle and H. Edelsbrunner. An improved algorithm for constructing kth-order Voronoi diagrams. *IEEE Trans. Comput.*, C-36:1349–1354, 1987.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.

[11] V. T. de Almeida and R. H. Güting. Using Dijkstra's algorithm to incrementally find the k-nearest neighbors in spatial network databases. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 58–62, New York, NY, USA, 2006. ACM.

[12] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[13] G. L. Dirichlet. Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *J. Reine Angew. Math.*, 40:209–227, 1850.

[14] M. Erwig. The graph Voronoi diagram with applications. *Networks*, 36(3):156–163, 2000.

[15] S. Fortune. Voronoi diagrams and Delaunay triangulations. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 193–233. World Scientific, Singapore, 1st edition, 1992.

[16] M. T. Goodrich and R. Tamassia. *Algorithm Design: Foundations, Analysis, and Internet Examples*. John Wiley & Sons, New York, NY, 2002.

[17] F. Hurtado, R. Klein, E. Langetepe, and V. Sacristán. The weighted farthest color Voronoi diagram on trees and graphs. *Comput. Geom. Theory Appl.*, 27(1):13–26, 2004.

[18] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 1973.

[19] M. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 840–851. VLDB Endowment, 2004.

[20] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, C-31:478–487, 1982.

[21] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, 27:125–128, 1988.

[22] K. Patroumpas, T. Minogiannis, and T. Sellis. Approximate order-k Voronoi cells over positional streams. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[23] M. Safar. K nearest neighbor search in navigation systems. *Mob. Inf. Syst.*, 1(3):207–224, 2005.

[24] K. Sugihara. Algorithms for computing Voronoi diagrams. In A. Okabe, B. Boots, and K. Sugihara, editors, *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

[25] G. M. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. premier Mémoire: Sur quelques propriétés des formes quadratiques positives parfaites. *J. Reine Angew. Math.*, 133:97–178, 1907.