Additively Weighted Voronoi Diagrams for Optimal Sequenced Route Queries*

Mehdi Sharifzadeh and Cyrus Shahabi

Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
[sharifza, shahabi]@usc.edu

Abstract

The *Optimal Sequenced Route (OSR)* query strives to find a route of minimum length starting from a given source location and passing through a number of *typed* locations in a specific sequence imposed on the types of the locations. In this paper, we propose a pre-computation approach to OSR query in vector spaces. We exploit the geometric properties of the solution space and theoretically prove its relation to *Additively Weighted Voronoi diagrams*. Our approach recursively accesses these diagrams to incrementally build the optimal sequenced route. Our experimental results verify that our pre-computation approach outperforms the previous index-based approaches in terms of query response time.

1 Introduction

Suppose that we are planning a Saturday trip in town as following: first we intend to visit a shopping center in the afternoon to check the season's new arrivals, then we plan to dine in an Italian restaurant in early evening, and finally, we would like to watch a specific movie at late night. Naturally, we intend to drive the minimum overall distance to these destinations. In other words, we need to find the locations of the shopping center s_i, the Italian restaurant r_j, and the theater t_k that shows the specific movie, which driving toward them considering the sequence of the plan shortens our trip (in terms of distance or time). Note that in this example, a time constraint enforces the order in which these destinations are to be visited; we usually do not have dinner in the afternoon, or do not go for shopping at late night. This type of query is also essential in other application domains such as *crisis management*, *air traffic flow management*, *supply chain management*, and *video surveillance*.

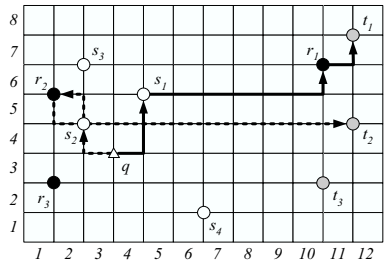


Figure 1: 3 different types of point sets

We call this type of query, where the order of the sequence in which some points must be visited is *enforced* and cannot be changed, the *Optimal Sequenced Route* or *OSR* query. Similar types of the planning queries have received recent attention by the database community [5, 9, 3, 2] to solve problems in the domains of air traffic flow and supply chain management, which shows the importance of these query types. We use the first example described above as our running example throughout the paper. Figure 1 shows three different types of *point sets* shown by white, black and gray circles, which represent shopping centers, Italian restaurants, and theaters, respectively, and a starting point q (shown by $\triangle$). The distance between two points is their Manhattan distance. Here, the route (q, s_1, r_1, t_1) (shown with solid lines in the figure) with the length of 12 units is the optimum answer to our query. One can show that OSR query may not be optimally answered by simply performing a series of independent nearest neighbor queries at different locations (e.g., 15-unit length route (q, s_2, r_2, t_2)).

We, for the first time, introduced the OSR query in [8] and proposed two online approaches for vector and metric spaces (e.g., road networks). The R-LORD algorithm of [8] subsequently performs range queries utilizing an R-tree index structure to filter out the points that cannot possibly be on the optimal route, and then builds the optimal route in reverse sequence (i.e., from

*This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0238560 (PECASE), IIS-0324955 (ITR), and unrestricted cash gifts from Google and Microsoft. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Proceedings of the third Workshop on STDBM Seoul, Korea, September 11, 2006

ending to the starting point). In this paper, we exploit the geometry of the OSR problem space to propose a novel *pre-computation* algorithm which utilizes Voronoi diagrams for processing OSR queries in vector spaces. In our running examples, we use Manhattan distance in $\mathbb{R}^2$ which is a representative of the network distance if the road network consists of a set of north-south and east-west roads [1].

1.1 Contributions

Assume that we are only interested in the optimal sequenced routes that follow a fixed sequence M. For example, assume that this fixed sequence for Figure 1 is $(white, black, grey)$. Suppose we can partition the space of points of interest (e.g., $\mathbb{R}^2$) into regions each including all the points with a common OSR. For instance, in Figure 1, there are many points similar to q for which the optimal sequenced route to a *white*, then a *black*, and finally a *grey* point is the route (s_1, r_1, t_1). Suppose that our partitioning identifies the region including these points. Furthermore, assume that the partitioning also identifies the region corresponding to each and every possible route (s_i, r_j, t_k) to a *white*, a *black*, and a *grey* point. Assume that we associate and store all regions and their corresponding OSRs. Therefore, for a given starting point q we can address the OSR query with sequence M by simply locating the *unique* region that includes q and reporting its corresponding optimal sequenced route. In Section 3, we prove that such partitioning exists as an *Additively Weighted Voronoi diagram*, a specific type of general Voronoi diagrams. We theoretically exploit interesting properties of these diagrams which makes them appropriate data structures for efficient OSR query processing.

We enhance the above query processing approach by exploiting an interesting property of OSRs proved in Lemma 1 (see Section 3). The lemma states that if (s_1, r_1, t_1) is q's optimal route to a *white*, a *black*, and a *grey* point, then (r_1, t_1) is s_1's optimal route to a *black* and then a *grey* point. *Recursively*, (t_1) is r_1's optimal route to a *grey* point. Lemma 1 enables us to avoid storing the complete OSR corresponding to each region of the diagram for a given sequence M; we store only the first point of the OSR for each region of the Voronoi diagram (e.g., s_1 for the region corresponding to (s_1, r_1, t_1)). Instead, we require to compute and store the Voronoi diagrams corresponding to all suffixes of M.

For instance, to answer queries with sequence $(white, black, grey)$ in Figure 1, our approach requires the diagrams corresponding to three sequences $(white, black, grey)$, $(black, grey)$, and $(grey)$. We refer to these diagrams as the *OSR-Voronoi family* of sequence $(white, black, grey)$. Now, we iteratively process a given OSR query. For the starting point q, we first find s_1, the white point associated with the region including q in the diagram of $(white, black, grey)$. Then, we find r_1, the black point stored with the region including s_1 in the diagram of $(black, grey)$. Finally, we seek for t_1, the grey point corresponding to the region including r_1 in the diagram of $(grey)$. Now, the final OSR of q is the route (s_1, r_1, t_1), which includes s_1, r_1, and t_1 in the order of their retrieval.

Our OSR query processing using Voronoi diagrams best suits the applications where the sequences of user's interest are known in advance. This allows pre-computation of the corresponding diagrams. Notice that using the OSR-Voronoi family of a sequence M, we can address OSR queries of *all suffix sequences* of M. Through extensive experiments with a real-world dataset, we show that our approach significantly outperforms the R-LORD approach proposed in [8] in terms of response time. We also show that the off-line process used to build the OSR-Voronoi family of a single sequence is reasonably fast. With a given sequence M, this pre-computation phase takes $O(\sum_{i=1}^{|M|} |U_{M_i}| \log |U_{M_i}|)$ computation steps.

2 Preliminaries

2.1 Formal Problem Definition

In this section, we describe the terms and notations used in [8] to formally define the OSR query. We use the same notations throughout the paper .

Let $U_1, \ldots, U_n$ be n sets, each containing points in a d-dimensional space $\mathbb{R}^d$, and $D(.,.)$ be a distance metric defined in $\mathbb{R}^d$ where $D(.,.)$ obeys the triangular inequality. To illustrate, in the example of Figure 1, U_1, U_2, and U_3 are the sets of black, white, and gray points, representing restaurants, shopping centers and theaters, respectively. We first define the following four terms.

Definition 1: Given n, the number of point sets U_i, we say $M = (M_1, M_2, \ldots, M_m)$ is a *sequence* if and only if $1 \leq M_i \leq n$ for $1 \leq i \leq m$. That is, given the point sets U_i, a user's OSR query is valid only if she asks for existing location types. For the example of Figure 1 where $n = 3$, $(2, 1, 2)$ is a sequence (specifying a shopping center, a restaurant, and a shopping center), while $(3, 4, 1)$ is not a sequence because 4 is not an existing point set. We use $sfx(M, i) = (M_{i+1}, \ldots, M_m)$ to refer to the suffix of M with size $m - i$.

Definition 2: We say $R = (p_1, p_2, \ldots, p_r)$ is a *route* if and only if $p_i \in \mathbb{R}^d$ for each $1 \leq i \leq r$. We use $p \oplus R = (p, p_1, \ldots, p_r)$ to denote a new route that starts from starting point p and goes sequentially through p_1 to p_r. The route $p \oplus R$ is the result of adding p to the head of route R. We use $sfx(R, i) = (p_{i+1}, \ldots, p_r)$ to refer to the suffix of R with size $r - i$.

Definition 3: We define the *length* of a route $R = (p_1, p_2, \ldots, p_r)$ as

$$L(R) = \sum_{i=1}^{r-1} D(p_i, p_{i+1}) \quad (1)$$

Note that $L(R) = 0$ for $r = 1$. For example, the length of the route (s_2, r_2, s_3) in Figure 1 is 4 units where D is the L_1 norm (i.e., Manhattan distance).

Definition 4: Let $M = (M_1, M_2, \ldots, M_m)$ be a sequence. We refer to the route $R = (p_1, p_2, \ldots, p_m)$ as a

sequenced route that follows sequence M if and only if $p_i \in U_{M_i}$ where $1 \leq i \leq m$. In Figure 1, (s_2, r_2, s_3) is a sequenced route that follows $(2, 1, 2)$ which means that the route passes only through a white, then a black and finally a white point.

Now, we formally define the OSR query.

Definition 5: For a given *starting* point q in $\mathbb{R}^d$ and the sequence $M = (M_1, M_2, \ldots, M_m)$, the **Optimal Sequenced Route** (OSR) Query, $Q(q, M)$, is defined as finding a sequenced route $R = (p_1, \ldots, p_m)$ that follows M where the value of the following function L is minimum over all the sequenced routes that follow M:

$$L(q, R) = D(q, p_1) + L(R) \quad (2)$$

Note that $L(q, R)$ is in fact the length of route $R_q = q \oplus R$. Throughout the paper, we use $Q(q, M) = (p_1, p_2, \ldots, p_m)$ to denote the answer to the OSR query Q. Without loss of generality, we assume that this optimal route is unique for given q and M. For the example in Section 1 where $(U_1, U_2, U_3) = (black, white, gray)$, $M = (2, 1, 3)$, and $D(.,.)$ is the shortest path distance, the answer to the OSR query is $Q(q, M) = (s_1, r_1, t_1)$.

One special variation of OSR is when the user asks for an optimal sequenced route that ends in a *given* destination q'. A special case of this query is where she intends to return to her starting location q. This variation of OSR can simply be addressed by defining a new point set $U_{n+1} = \{q'\}$ and a new sequence $M' = (M_1, \ldots, M_m, n+1)$. Consequently, $Q(q, M')$ will be the optimal route following M which ends in q'.

Table 1 summarizes the notations we use throughout the paper.

2.2 Voronoi Diagrams

The general Voronoi diagram of a given set S including points in $\mathbb{R}^d$ partitions the space into regions each including all points with a common closest point in the given set according to a distance metric $\mathbf{d}(.,.)$. The region corresponding to the point p contains all the points $q \in \mathbb{R}^d$ for which we have

$$\forall p' \in S, p' \neq p \Rightarrow \mathbf{d}(q, p) \leq \mathbf{d}(q, p') \quad (3)$$

The equality holds for the points on the borders of p's and p''s regions. Incorporating arbitrary distance metrics $\mathbf{d}(.,.)$ in the above equation results in different variations of Voronoi diagrams. Figure 2 shows the *standard* Voronoi diagram of nine points in $\mathbb{R}^2$ where the distance metric is Euclidean. We refer to the region $V(p)$ containing the point p as its Voronoi *cell*. We also refer to point p as the *generator* of Voronoi cell $V(p)$.

Now assume that S is a set of points $p \in \mathbb{R}^d$ each assigned a weight $w(p) \in \mathbb{R}$. We define the distance metric $\mathbf{d}(x, p)$ as $D(x, p) + w(p)$ where $D(.,.)$ denotes a distance metric. Without loss of generality, we assume that $D(.,.)$ is Euclidean distance. The Additively Weighted (AW) Voronoi diagram of the points in S with respect to distance function $D(.,.)$ is defined using Equation 3. Here, the cell corresponding to point p

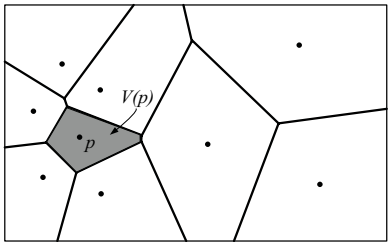


Figure 2: The standard Voronoi diagram of 9 points, the point p and its Voronoi cell $V(p)$

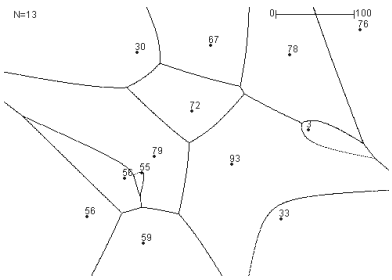


Figure 3: The AW-Voronoi diagram of 13 points each labeled with its positive weight

contains all the points $q \in \mathbb{R}^d$ for which we have

$$\forall p' \in S, p' \neq p \Rightarrow D(q, p) + w(p) \leq D(q, p') + w(p') \quad (4)$$

Figure 3 illustrates the AW-Voronoi diagram of a set of 13 points labeled with their positive weights.

The Computational Geometry literature generally uses *negative* weights in Equation 4 to define the AW-Voronoi diagrams. When negative weights are assigned to points p, their AW-Voronoi diagram can be seen as the standard Voronoi diagram of a set of d-dimensional balls each centered at a point p with a radius equal to the absolute value of $w(p)$. Here, the set S includes balls such as b centered at a point p. The distance $\mathbf{d}(q, b)$ between a ball b and the point q outside b is the smallest Euclidean distance between q and any point inside b. The literature uses this view to define the AW-Voronoi diagram of the centers of a set of balls (e.g., disks in $\mathbb{R}^2$) [6]. Each region of the space is assigned to a unique ball which is the common closest ball of all points inside that region. Figure 4 illustrates defining the AW-Voronoi diagram of five points with negative weights $w(p) < 0$ using five corresponding disks. Here, $\mathbf{d}(q, c)$, the distance of a point q to a disk c centered at p with radius $r = -w(p)$ is defined as $D(q, p) - r$. As the figure shows, point q which is inside the cell of disk c (corresponding to point p) is closer to disk c than to disk c'. It also shows that some points such as p'' have empty cells. That is, no point in the space is closer to the disk c'' than to any other disk.

The AW-Voronoi diagram demonstrates the following properties:

Property AWV-1 With Euclidean distance (i.e. L_2 norm) as metric $D(.,.)$ in Equation 4, the cell edges are either straight lines or hyperbolic curves [6].

Symbol	*Meaning*	*Symbol*	*Meaning*
U_i	a *point set* in $\mathbb{R}^d$	$\|U_i\|$	cardinality of the set U_i
n	number of point sets U_i	$D(.,.)$	distance function in $\mathbb{R}^d$
M	a sequence, $= (M_1, \ldots, M_m)$	$\|M\|$	m, size of sequence M = number of items in M
M_i	i-th item of M	R	route $(p_1, p_2, \ldots, p_r)$, where p_i is a point
$\|R\|$	r, number of points in R	p_i	i-th point in R
$L(R)$	length of R	$q \oplus R$	route $R_p = (q, p_1, \ldots, p_r)$ where $R = (p_1, \ldots, p_r)$
$sfx(R,i)$	route $(p_{i+1}, \ldots, p_r)$, a suffix of R	$sfx(M,i)$	sequence $(M_{i+1}, \ldots, M_m)$, a suffix of M
$L(q,R)$	$L(q \oplus R)$, length of the route $q \oplus R$		

Table 1: Summary of notations

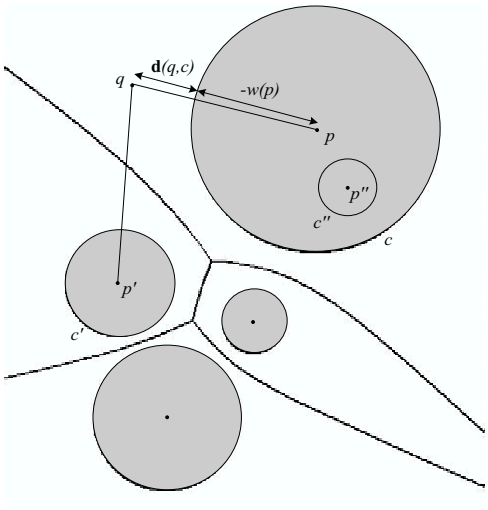


Figure 4: The AW-Voronoi diagram of five disks (i.e., five points with negative weights). Here, the distance function $D(.,.)$ is Euclidean.

Hence, these curved cells are not convex polygons as the cells of general Voronoi diagrams. Using Manhattan distance (i.e. L_1 norm), all edges become straight lines.

Property AWV-2 The arbitrary numeric values of weights can result in empty cells [6]. We refer to the generator point of an empty cell as a *trivial* point.

Property AWV-3 The AW-Voronoi diagram of n points consists of at most $O(n)$ connected edges [7].

3 OSR and AW-Voronoi Diagrams

Assume that we are only interested in the optimal sequenced routes that follow a fixed sequence M (i.e., $Q(q, M)$ for points such as q). Suppose we can partition the space $\mathbb{R}^d$ into regions each including all the points q with a common OSR. That is, for any two points x and y inside the same region, we have $Q(x, M) = Q(y, M)$. Assume that for each region, we pre-compute this common OSR. Furthermore, we store all regions and their corresponding OSRs for the given sequence M. Therefore, for a given starting point q we can address the query $Q(q, M)$ by simply locating the region including q and reporting its corresponding optimal route.

In this section, we show that the above partitioning which facilitates processing OSR queries exists as an AW-Voronoi diagram. We prove that a unique property of optimal sequenced routes enables us to employ this partitioning. First, we prove Lemma 1 which states that the optimal route from p_i that follows a suffix of the original sequence M is the same as the corresponding suffix of the optimal route from q that follows M.

Lemma 1. *Given the sequence $M = (M_1, \ldots, M_m)$ and the starting point q, if $Q(q, M) = R = (p_1, \ldots, p_m)$, then for any $1 \leq i < m$, we have $Q(p_i, M') = sfx(R, i)$ where $M' = sfx(M, i)$.*

Proof. The proof is by contradiction. Assume that $Q(p_i, M') = R' = (p'_1, \ldots, p'_{m-i})$. Obviously $sfx(R, i) = (p_{i+1}, \ldots, p_m)$ follows sequence M', therefore we have $L(p_i, R') < L(p_i, (p_{i+1}, \ldots, p_m))$. We add $L(q, (p_1, \ldots, p_i))$ to the both sides of this inequality to get:

$$L(q, (p_1, \ldots, p_i, p'_1, \ldots p'_{m-i})) < L(q, (p_1, \ldots, p_m))$$

The above inequality shows that the answer to $Q(q, M)$ must be $(p_1, \ldots, p_i, p'_1, \ldots, p'_{m-i})$ which clearly follows sequence M. This contradicts our assumption that $Q(q, M) = (p_1, \ldots, p_m)$. □

This property of the OSRs implies that *only* the last point of the optimal sequenced route $Q(q, M)$ (i.e., p_m) must necessarily be the closest point of U_M to its previous point in the route (i.e., p_{m-1}).

Here, we utilize the special case of Lemma 1 for the first point of the route (i.e., p_1 for $i = 1$). For instance, if the optimal route from point q to a white, a black, and a grey point is (s_1, r_1, t_1), then the optimal route from the point s_1 to a black then a grey point is (r_1, t_1) (see Figure 1). Therefore, if we identify the white point s_1 with which the optimal route from q to a white, a black, then a grey point *starts* (i.e., the first point of the route) and we know that the optimal route from s_1 to a black and then a grey point is (r_1, t_1), we have found the optimal route from q (we just append s_1 to the head of (r_1, t_1)). The following lemma identifies this first point.

Lemma 2. *Given the sequence M and the starting point q, if there exists a point $p_1 \in U_{M_1}$ so that $\forall p'_1 \in U_{M_1}, p'_1 \neq p_1$*

$$D(q, p_1) + L(p_1, Q(p_1, M')) < D(q, p'_1) + L(p'_1, Q(p'_1, M')) \quad (5)$$

where $M' = sfx(M, 1)$, then the optimal route $Q(q, M)$ starts with p_1 which is followed by the points in $Q(p_1, M')$ (i.e., $Q(q, M) = p_1 \oplus Q(p_1, M')$).

Proof. The proof is by contradiction. Assume that $Q(q, M) = (p'_1, \ldots, p'_m)$ starts with $p'_1 \neq p_1$. By definition, the length of $q \oplus Q(q, M)$ is minimum over all the routes which follow sequence M. It is clear that the route $p_1 \oplus Q(p_1, M')$ follows M, so we have

$$L(q, (p_1 \oplus Q(p_1, M'))) \geq L(q, Q(q, M)) \quad (6)$$

Lemma 1 states that we have $Q(p'_1, M') = (p'_2, \ldots, p'_m)$ where $M' = sfx(M, 1)$. Hence, we get

$$Q(q, M) = p'_1 \oplus Q(p'_1, M') \quad (7)$$

Therefore, we replace $Q(q, M)$ in Equation 6 to get

$$L(q, p_1 \oplus Q(p_1, M')) \geq L(q, p'_1 \oplus Q(p'_1, M')) \quad (8)$$

Finally, we use the definition of function $L()$ to change Equation 8 as follows for the point $p'_1 \neq p_1$:

$$D(q, p_1) + L(p_1, Q(p_1, M')) \geq D(q, p'_1) + L(p'_1, Q(p'_1, M')) \quad (9)$$

The above inequality contradicts the one given in the assumption of the lemma.

□

In the example of Figure 1, U_1, U_2, and U_3 are the sets of black, white, and gray points, respectively. For the sequence $M = (2, 1, 3)$, we have $M' = (1, 3)$ and

$$Q(s_1, M') = (r_1, t_1), Q(s_2, M') = (r_3, t_3),$$

$$Q(s_3, M') = (r_1, t_1), Q(s_4, M') = (r_1, t_1)$$

$$D(q, s_1) + L(s_1, Q(s_1, M')) = 3 + (7 + 2) = 12$$

$$D(q, s_2) + L(s_2, Q(s_2, M')) = 2 + (3 + 9) = 14$$

$$D(q, s_3) + L(s_3, Q(s_3, M')) = 4 + (8 + 2) = 14$$

$$D(q, s_4) + L(s_4, Q(s_4, M')) = 5 + (9 + 2) = 16$$

According to Lemma 2, the OSR of q to a white, a black and a grey point starts with s_1 followed by $Q(s_1, M') = (r_1, t_1)$.

We now strive to find the locus of all points in space whose OSR that follows a given sequence starts with a point p_1 and hence all share a common OSR for that sequence. In Figure 1, for any white point s_i, there are points such as q whose optimal routes given the sequence $(white, black, grey)$ start with s_i. Notice that the rest of their optimal route is the same (e.g., (r_1, t_1) for s_1) and depends only on the location of point s_i and the given sequence (e.g., $(black, grey)$).

Theorem 1. *Let $M = (M_1, M_2, \ldots, M_m)$ be a given sequence and $M' = sfx(M, 1)$. The locus of all points q with a common optimal sequenced route $Q(q, M)$ which starts with p_1 is inside p_1's cell in the AW-Voronoi diagram of the set of points in U_{M_1} where the weight of each point p is $w(p) = L(p, Q(p, M'))$. Their common optimal route is $Q(q, M) = p_1 \oplus Q(p_1, M')$.*

Proof. Let the set $S = U_{M_1}$ and $w(p) = L(p, Q(p, M'))$. According to the definition of the AW-Voronoi cell of p_1 given by Equation 4, for the point q inside this cell we get $\forall p'_1 \in U_{M_1}, p'_1 \neq p_1$

$$D(q, p_1) + L(p_1, Q(p_1, M')) < D(q, p'_1) + L(p'_1, Q(p'_1, M'))$$

Therefore, according to Lemma 2 the optimal route $Q(q, M)$ is $p_1 \oplus Q(p_1, M')$ which clearly starts with p_1, the generator of the Voronoi cell including q. □

Given a sequence M, we refer to the AW-Voronoi diagram of the points p in U_{M_1} using the weights $w(p) = L(p, Q(p, M'))$ where $M' = sfx(M, 1)$, as the **OSR-Voronoi diagram** of the sequence M. Figure 5a illustrates the OSR-Voronoi diagram of sequence $(white, black, grey)$ (i.e., $M = (2, 1, 3)$). The distance function $D(.,.)$ is Manhattan distance and each white point s_i is labeled with its weight (i.e., $L(s_i, Q(s_i, (1, 3)))$). The OSR-Voronoi diagram has some interesting properties whose proofs are omitted due to lack of space:

Property OSRV-1 All points in the OSR-Voronoi diagram of the sequence M (any $p \in U_{M_1}$) have positive weights.

Property OSRV-2 Given the sequence M, if for the points p and $p' \in U_{M_1}$ we have $Q(p, M') = Q(p', M') = (p_2, \ldots, p_m)$ and $D(p, p_2) = D(p, p') + D(p', p_2)$ where $M' = sfx(M, 1)$, then p is a trivial point in OSR-Voronoi diagram of sequence M. That is, the Voronoi cell of p is empty and no OSR that follows M starts with p.

4 OSR Query Processing using OSR-Voronoi Diagrams

In this section, we describe our OSR query processing approach that utilizes OSR-Voronoi diagrams defined in Section 3. Assume that we have already built the OSR-Voronoi diagram of a given sequence M. The user asks for the OSR $Q(q, M)$ given a starting point q. Subsequently, we locate the Voronoi cell which contains q in the OSR-Voronoi diagram of M. According to Theorem 1, the OSR of q starts with the generator of this cell (e.g., s_1 in Figure 5a). This point is then followed by the result of another OSR query (i.e., $Q(s_1, M')$). If we already knew the answer to this second query, we immediately report user's desired route as $s_1 \oplus Q(s_1, M')$. Now the problem is that the we need the OSR-Voronoi diagram of sequence M' to answer the second OSR query. Repeating the same reasoning for M' verifies that we will require the OSR-Voronoi diagrams of all suffixes of the original sequence M. That is, to find the OSR which follows $M = (M_1, \ldots, M_m)$ we need the OSR-Voronoi diagrams of all sequences $(M_i, \ldots, M_m)$ for all $1 \leq i \leq m$. We refer to these diagrams as the *OSR-Voronoi family* of M. For instance, to answer queries with sequence $(white, black, grey)$ in

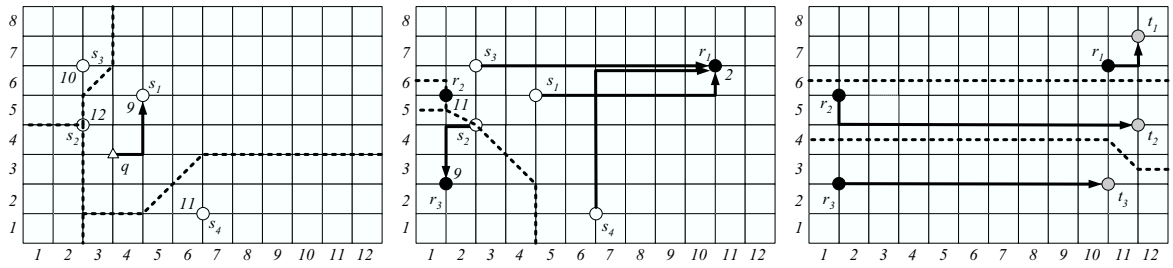


Figure 5: The OSR-Voronoi diagrams of a) the sequence $(white, black, grey)$, b) the sequence $(black, grey)$, and c) the sequence $(grey)$ using the white points in Figure 1

```
Function ComputeOSR(point q, sequence M)
    Returns route
1. V(p) = the cell containing q in the OSR-Voronoi diagram of M
2. p = the generator of V(p)
3. if |M| = 1 then
4.     return (p)
5. else
6.     return p ⊕ ComputeOSR(p, sfx(M, 1))
```

Figure 6: Computing OSR for query $Q(q, M)$

Figure 1, we require the OSR-Voronoi diagrams of sequences $(white, black, grey)$, $(black, grey)$, and $(grey)$. Notice that the OSR-Voronoi family of a sequence can also be used to answer OSR queries of any suffix of that sequence.

4.1 Query Processing

Here, we assume that we have already built the OSR-Voronoi family of a given sequence M. Subsequently, we employ this set of diagrams to answer the OSR query $Q(q, M)$. Section 4.2 shows how we recursively build all these diagrams.

We describe our query processing approach using the example of Figure 1. Notice that here the sequence is $(white, black, grey)$ and the distance function $D(.,.)$ is Manhattan distance. We have already built and loaded the OSR-Voronoi diagrams of sequences $(white, black, grey)$, $(black, grey)$, and $(grey)$ illustrated in Figures 5a, 5b, and 5c, respectively. Notice that the last diagram is the standard Voronoi diagram on grey points as the OSR of a point q which follows the sequence $(grey)$ is simply the closest grey point to q (see Lemma 1 for $i = m - 1$). First, we locate the cell containing q in the OSR-Voronoi diagram of $(white, black, grey)$. Following Theorem 1, our OSR starts with s_1, the generator of this cell (see Figure 5a). The rest of the OSR is the same as the OSR of s_1 which follows $(black, grey)$. Hence, we next find r_1, the generator of the cell containing s_1 in the OSR-Voronoi diagram of $(black, grey)$ shown in Figure 5b. Finally, we find the closest grey point to r_1 as the generator of the cell containing r_1 in the OSR-Voronoi diagram of $(grey)$ (see Figure 5c). The OSR of point q which follows sequence $(white, black, grey)$ is (s_1, r_1, t_1) built using the three subsequently retrieved points. Figure 6 shows the pseudo-code of our OSR query processing approach as a *recursive* function *ComputeOSR(point, sequence)*.

4.2 Data Structures

The OSR-Voronoi diagrams are instances of AW-Voronoi diagrams whose weights are constrained by geometry. Therefore, we build them using any algorithm for building AW-Voronoi diagrams. The algorithm proposed by Karavelas and Yvinec [4] computes the diagram of n points in $O(n \log n)$ steps. Our approach utilizes the OSR-Voronoi diagrams of all suffixes of a sequence M to find the OSRs that follow M. Therefore, to answer queries with sequence $(white, black, grey)$ in Figure 1, we require the OSR-Voronoi diagrams of this sequence together with those of sequences $(black, grey)$ and $(grey)$. In general, for a given sequence M, we require $|M|$ OSR-Voronoi diagrams of all suffixes of M to answer $Q(q, M)$. Hence, the pre-computation phase takes $O(\sum_{i=1}^{|M|} |U_{M_i}| \log |U_{M_i}|)$ computation steps.

We incrementally build the OSR-Voronoi family of $(white, black, grey)$ as follows. First, we build the OSR-Voronoi diagram of $(grey)$ shown in Figure 5c which is simply the standard Voronoi diagram of the set of grey points $T = \{t_1, t_2, t_3\}$ with respect to Manhattan distance [6]. Second, for each black point r_i we find the generator grey point of the cell containing r_i in this OSR-Voronoi diagram. In Figure 5c, each black point is connected to its corresponding grey generator by a solid line. We assign the distance between the point r_i and its corresponding grey point to the weight of r_i. For instance, we assign $w(r_1) = 2$. Third, we build the OSR-Voronoi diagram of $(black, grey)$ using the set of black points $R = \{r_1, r_2, r_3\}$ and their weights (see Figure 5b). Similar to the previous step, for each white point s_i we locate r_x, the generator of the cell which contains s_i in the diagram built in this step and assign $D(s_i, r_x) + w(r_x)$ to the weight of s_i. For example, the weight of s_1 is $7 + 2 = 9$. Finally, we build the OSR-Voronoi diagram of $(white, black, grey)$ using the set of white points $S = \{s_1, s_2, s_3, s_4\}$ with respect to their corresponding weights (see Figure 5a). Figure 7 shows the pseudo-code of the above algorithm.

Once we computed the OSR-Voronoi family of a sequence, we require to store the boundaries of the Voronoi cells corresponding to each OSR-Voronoi diagram of the family. However, with Euclidean distance, these boundaries consist of curved edges and hence cannot be stored as simple polygons. This problem can be

```
Function BuildOSRVoronois(sequence M = (M_1, ..., M_m))
    Returns OSR-Voronoi diagrams
1.  OSRV((M_m)) = the standard Voronoi diagram of U_{M_m}
2.  for each p in U_{M_m}
3.      w(p) = 0
4.  i = m − 1
5.  while i > 0 do {
6.      M' = sfx(M, i)
7.      for each p in U_{M_i} {
8.          V(p') = the cell containing p in the OSRV(M')
9.          p' = the generator of V(p')
10.         w(p) = D(p, p') + w(p')
        }
11.     OSRV(sfx(M, i − 1)) = the AW-Voronoi diagram of U_{M_i}
12.     i = i − 1
    }
13. return OSRV's
```

Figure 7: Incrementally building the data structures required to answer the OSR query $Q(q, M)$

solved in the two following ways. Each solution guarantees that given a starting point q, we can easily determine the cell containing q.

1. The cells are approximated by convex polygons and consequently these polygons are stored. The polygons are also indexed by a spatial index structure such as R-tree. Therefore, the cell containing the starting point q is easily retrieved by a spatial `contains()` query. This solution introduces an error in OSR query processing proportional to the approximation error.

2. Instead of storing the cells, the neighborhood graph of the cell generators is stored (similar to Delaunay graph for the general Voronoi diagrams [6]). Vertices of the graph are the generators of the OSR-Voronoi diagram of M. There is a graph edge connecting p and p' iff $V(p)$ and $V(p')$ have common boundaries. Now, to determine the inclusion of a starting point q in a cell, we start traversing the graph from any vertex p. We iteratively visit the vertex p', the neighbor of the current vertex p which minimizes $D(q, p') + w(p')$ among p's neighbors and p itself. When no such a vertex is found, q is inside the Voronoi cell of the current vertex p.

5 Performance Evaluation

We conducted several experiments to evaluate the performance of our proposed approach. We compared the query response time of our Voronoi-based approach (denoted by OSRV) with that of our R-LORD algorithm proposed in [8]. We also evaluated our new OSRV approach with respect to its overall time required to build the OSR-Voronoi family of a given query. We evaluated OSRV by investigating the effect of the following two parameters on its performance: 1) size of sequence M in $Q(q, M)$ (i.e., number of points in the optimal route), and 2) cardinality of the datasets (i.e., $\sum_{i=1}^{n} |U_i|$).

We used a real-world dataset which is obtained from the U.S. Geological Survey (USGS) and consists of $950,000$ locations of different businesses (e.g., schools) in the entire country. Table 2 shows the characteristics of this dataset. In our experiments, we randomly selected sets of $40K$, $70K$, $250K$ and $500K$, and $950K$ points from this dataset. We used CGAL's implementation of [4] to implement the OSR-Voronoi diagrams [1]. We ran 1000 OSR queries from random starting points on a DELL Precision 470 with Xeon 3.2 GHz processor and 3GB of RAM.

Points	*Size*	*Points*	*Size*
Hospital	5,314	Building	15,127
Summit	69,498	Cemetery	109,557
Church	127,949	School	139,523
Populated place	167,203	Institution	319,751

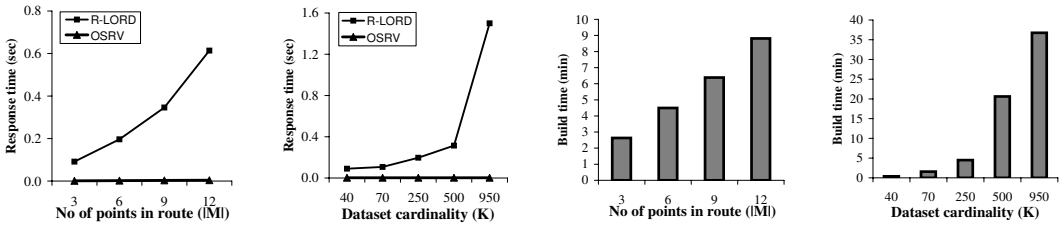
Table 2: USGS dataset

In the first set of experiments, we study the performance of OSRV in terms of query time. Figure 8a shows the query response time for our OSRV and R-LORD approaches when the number of points in optimal route (i.e., $|M|$) varies from 3 to 12. All the required diagrams fit in main memory, so the reported time represents CPU time. While the figure depicts the experimental results on 250K USGS dataset, the trend is the same with those of our other datasets with different cardinalities. Figure shows that OSRV always outperforms R-LORD. As expected, R-LORD performs slower with the bigger sequences as it requires more range queries on the R-tree. In contrast, OSR query with OSRV has almost no cost in time. OSRV's response time is unnoticeably increasing as OSRV consists of a sequence of only $|M|$ point locations on our efficient AW-Voronoi diagrams.

Figure 8b illustrates the result of our second set of experiments on the impact of the cardinality of the dataset on the efficiency of OSRV. We varied the cardinality of our real datasets from $40K$ to $950K$ and ran OSR queries of sequence size $|M| = 6$. Figure 8b shows that OSRV's performance is not much affected by the dataset size and verifies the scalability of OSRV. This is while the processing time of R-LORD moderately increases for larger datasets. For example, R-LORD's query response time is 0.09 seconds for $40,000$ points, and it increases by a factor of 16 when the number of points increases to $950,000$ by a factor of 24. Consequently, OSRV is the superior approach although R-LORD also exhibits a reasonable performance for large datasets.

The last experiment aims to measure the time required to build the diagrams in the OSR-Voronoi family of a given sequence. Note that this is a batch process which is performed off-line. Figure 8c illustrates the average build time for different sizes of the sequence M for $250K$ dataset. Moreover, Figure 8d depicts the result of the same experiment for different dataset cardinalities when the sequence size is 6. Figure 8c shows that it takes only 9 minutes to build 12 OSR-Voronoi diagrams needed for the OSR query of a sequence of size 12. However, Figure 8d shows that despite its excellent performance for small datasets, OSRV's procedure for building the OSR-Voronoi family of a sequence of size 6 takes about 36 minutes for the $950K$ dataset. This is still a reasonable performance for an off-line process.

[1] http://www.cgal.org/



a) Query cost vs. sequence size (250K) b) Query cost vs. cardinality ($|M| = 6$) c) Build cost vs. sequence size (250K) d) Build cost vs. cardinality ($|M| = 6$)

Figure 8: Experimental results on the performance of OSRV

We believe that upgrading main-memory AW-Voronoi diagrams to their secondary-memory versions will significantly improve OSRV's off-line performance for large datasets.

6 Related Work

In an independent research, Li et al. [5] studied Trip Planning Queries (TPQ), a class of queries similar to our OSR query. With a TPQ, the user specifies a subset (not a sequence) of location types R and asks for the optimal route from her starting location to a specified destination which passes through at least one database point of each type in R. In particular, TPQ eliminates the sequence order of OSR to define a new query. Consequently, finding accurate solutions to TPQ becomes NP-hard as the size of candidate space significantly grows. The paper proposes several approximation methods to provide near-optimal answers to TPQs. In theory, OSR algorithms are able to address address TPQ queries. This can be done by running any OSR algorithm $|R|!$ times, each time using a permutation of point types in R as the sequence M and returning the optimal route among all the resulted routes. This exponentially complex solution is inefficient in practice. The approximation algorithms of [5] are designed to handle the exponential growth of the problem's search space.

Terrovitis et al. [9] addressed k-stops shortest path problem for spatial databases. The problem seeks for the optimal path from a starting location to a given destination which passes through exactly k intermediate points of the location database. The k-stops problem is a specialized case of OSR queries. To be specific, OSR reduces a k-stops problem to query $Q(q, M)$ where q is the starting location, $M = (1, \ldots, 1)$, $|M| = k$, and U_1 is the single given point set representing the location database. The destination is considered by the variation of the above OSR query described in Section 2.1. A major drawback is that [9] only considers Euclidean space.

7 Conclusions and Future work

We studied the problem space of the OSR query in vector spaces. We exploited the geometric properties of the solution space and identified the AW-Voronoi diagrams as its corresponding geometric representation. In particular, we proved that given a sequence M the locus of all points with a common optimal sequenced route R is the cell of the first point of R in a AW-Voronoi diagram whose weights depend on the sequence M. Based on our theoretical findings, we proposed a novel OSR query processing approach using a family of pre-computed AW-Voronoi diagrams. Through experiments with a real-world dataset, we showed that our approach significantly outperforms the previous solutions to OSR query in terms of response time.

While in this paper we adapted AW-Voronoi diagrams for OSR queries, we believe that these diagrams can also be used to answer a variation of OSR query where the sequence is not important (TPQ problem [5]). An orthogonal problem is studying efficient algorithms to store the AW-Voronoi diagrams in secondary storage.

References

[1] Y. Du, D. Zhang, and T. Xia. The Optimal-Location Query. In *Proceedings of SSTD'05*, pages 163–180, 2005.

[2] C. du Mouza, P. Rigaux, and M. Scholl. Efficient Evaluation of parameterized pattern queries. In *Proceedings of CIKM'05*, pages 728–735. ACM, 2005.

[3] M. Hadjieleftheriou, G. Kollios, P. Bakalov, and V. J. Tsotras. Complex Spatio-Temporal Pattern Queries. In *Proceedings of VLDB'05*, pages 877–888, 2005.

[4] M. I. Karavelas and M. Yvinec. Dynamic Additively Weighted Voronoi Diagrams in 2d. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'02)*, pages 586–598, London, UK, 2002. Springer-Verlag.

[5] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On Trip Planning Queries in Spatial Databases. In *Proceedings of SSTD'05*, pages 273–290. Springer, August 2005.

[6] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons Ltd., 2nd edition, 2000.

[7] Y. Ostrovksy-Berman. Computing Transportation Voronoi Diagrams in Optimal Time. In *Proceedings of the 21st European Workshop on Computational Geometry (EWCG'05)*, pages 159–162, 2005.

[8] M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi. The Optimal Sequenced Route. *The VLDB Journal*, 2006. To appear.

[9] M. Terrovitis, S. Bakiras, D. Papadias, and K. Mouratidis. Constrained Shortest Path Computation. In *Proceedings of SSTD'05*, pages 181–199. Springer, August 2005.