Voronoi-based Trajectory Search Algorithm for Multi-locations in Road Networks

Yu CHEN,  Jian XU*,  Ming XU,  Ning ZHENG,  Jie YANG

College of Computer, Hangzhou Dianzi University, Hangzhou 310018, China

Abstract

Trajectory search for multi-locations in a network is to find the trajectories from the historical dataset such that they can connect the given locations in the network. Existed research mainly focused on the Euclidean space, while in many cases, the track of moving object is often closely related with road networks, and it’s restricted by road networks. In order to obtain an efficient query algorithm in this case, the paper proposes a Voronoi-based trajectory search algorithm for multi-locations. First we construct Voronoi Diagram of road networks, and use this structure to simplify the historical trajectory. Then we get connected trajectories by querying the simplified trajectories dataset. Compared with previous methods, the algorithm can transform the trajectories into paths of road networks, and it greatly reduces the amount of historical trajectories and searching time. Detailed analysis is given and the efficiency of algorithm is demonstrated through experiments.

Keywords: Trajectory Search; Trajectory Simplification; Voronoi Diagram; Road Networks; Locations

1 Introduction

With the rapid development of location-acquisition technology, wireless communications and GPS-enabled mobile devices, people can record their geographic positions and movements anytime and anywhere, so that the massive historical trajectory data are coming out. The utility of trajectory relies on the trajectory’s effectiveness and efficiency in query processing among trajectory database. Trajectory search and its variants have been applied in many aspects of daily life, such as vehicle navigation, traffic flow analysis, travel behavior analysis, trajectory model mining, path recommendation and prediction, etc. Being an important form of trajectory search, trajectory search for multi-locations can return the trajectory that can connect several search locations. For instance, a traveler, in a new city, may need to plan a travel itinerary that go through several scenic spots. At this situation, the search would be quite useful because it can provide the traveler a similar itinerary that the others travelled as a reference. In addition, by using this search, the travel agency can further analyze the popular itinerary of some scenic regions; the zoologist can find the static food spots for the closest animal migration trajectory; the
transport agency can study the travel behavior of the local people in order to find out the reason of some traffic congestion.

Spatial-temporal trajectory search has been studied in [1-3], but most of them are based on the Euclidean space. In reality, the majority move objects distribute in the urban road networks, and their movements are constrained by the given road networks. The traditional one needs to scan the nearby sampling points of every query spot in order to confirm the nearest neighbor trajectory. It may lead to a situation that several sampling points in one trajectory might be over calculating, which is obviously not necessary. Considering the inefficiency of trajectory search algorithm for multi-locations based on the Euclidean space, a new method is needed to solve this problem.

Motivated by the shortcomings of existing approach, this paper proposes an efficient algorithm called Voronoi-based trajectory search for multi-locations, which avoids the above shortcomings. Firstly, the method uses the Voronoi diagram of road network nodes to simplify trajectory, its purpose is to reduce the redundancy and uniform sampling rate of trajectory. Then, using network range query based on network distance to ensure effective connectivity of query locations, which makes the algorithm more suitable for road networks. These can dramatically reduce the algorithm search space and improve search efficiency.

The rest of paper is organized as follows. Firstly, Section 2 briefly review the related work, and then algorithm and its performance analysis is presented in Section 3. Finally, we present the experimental results in Section 4 and conclude the paper in Section 5.

2 Related Work

Due to the character of trajectory data which reflects the behavior of human beings, a lot of researchers focus on trajectory query study in the past years. The researchers are mainly on similarity search on trajectory. Several typical similarity functions for different applications include Euclidean Distance [4], Dynamic Time Warping (DTW) [5], Longest Common Subsequence (LCSS) [6], Edit Distance with Real Penalty (ERP) [7], and Edit Distance on Real Sequences (EDR) [8]. As these similarity functions are used to measure the distance of the two raw trajectory sampling points, they don't apply to searching trajectories for multi-locations under the constrains of road networks.

Chen et al [2] have proposed the problem of searching trajectories by locations, in which context the query is only a small set of locations, while the target is to find the \( k \) Best-Connected Trajectories (\( k \)-BCT) from a database such that the \( k \)-BCT best connect the designated locations geographically. It uses R-tree to index trajectory sampling points, and finds the nearest candidate trajectory set by using the \( k \)-Nearest Neighbor search in each query locations, and the intersection of them is BCT. Zheng et al [3] studied the problem of efficient similarity search on activity trajectory database. It considered both the trajectory connectivity and desired activities. But these studies above deal with near neighbor query based on Euclidean distance, and the result couldn't satisfy the most users' need, such as the point which is nearest neighbor in Euclidean space, does not satisfy people's need in the real world. This is because the river, highway and others stop it to arrive directly. With the increase of trajectory scale, the time and space costs of trajectory indexes will increase quickly, because the trajectory data is massive and redundant. And it comes along the computing cost largely for query.
3 Query Processing

3.1 Problem formulation

In this section, we clarify some terms first and then briefly introduce the idea of the algorithm. A premise of the Voronoi-based trajectory search algorithm for multi-locations is that a Voronoi diagram of road network nodes is constructed, and then trajectory dataset is simplified according to the Voronoi diagram. Finally, using the node of road networks which is the nearest neighbor of query location for network range query, and getting trajectories which can connect query locations.

**Definition 1** A raw trajectory $T$ is defined as a finite sequence of GPS sampling points, $T = \{p_1, p_2, \ldots, p_n\}$, each sampling point $p_i$ contains latitude, longitude and timestamp, and it can be represented in the form of a 3-tuple $(x, y, t)$.

**Definition 2** Given a road network $N = (V, E)$, which $V = \{v_1, \ldots, v_n\} \in R^2(2 < n < \infty)$ is the node set, $E = \{e_1, \ldots, e_m\}$ is the edge set, and edge doesn’t contain any other node. Then, the Voronoi polygon (VP) of $v_i$ is defined as:

$$VP(v_i) = \{o|o \in R^2, \text{dist}(o, v_i) \leq \text{dist}(o, v_j), i \neq j\}$$

(1)

where $\text{dist}(o, v_i)$ is the minimum Euclidean distance between object $o$ and $v_i$. $v_i$ is called the generator of polygon. The set of Voronoi polygon given by $VD(V) = \{VP(v_1), \ldots, VP(v_n)\}$ is called the Voronoi Diagram (VD) generated by $V$.

**Property 1** A Voronoi diagram of a set of discrete objects $V$ is unique.

**Property 2** If an object $o$ is located the $VP(v_i)$, the Euclidean distance of the object $o$ to $v_i$ must smaller than to any other generator.

**Property 3** The relationship between generator points and Voronoi edges is $n_e \leq 3n - 6$.

**Property 4** The average number of Voronoi edges per Voronoi polygon is at most 6. Because $2(3n - 6)/n = 6 - 12/n \leq 6$.

**Definition 3** Given a raw trajectory $T = \{p_1, p_2, \ldots, p_n\}$ and the Voronoi diagram of road network node $VD(V)$, trajectory simplification for $T$ is a process that transforms $T$ into $T' = \{v_1, v_2, \ldots, v_k\}$ where $v_i \in V(1 < i < k)$. $T'$ is called the simplified trajectory of $T$.

**Definition 4** Given the simplified trajectory dataset $D = \{T'_1, T'_2, \ldots, T'_n\}$, the Voronoi diagram of road network node $VD(V)$ and query locations set $Q$, Voronoi-based connected trajectory search (VBCT) is to find trajectories which can connect multi-locations at the given network distance in road networks.

3.2 Algorithm

Voronoi-based trajectory search algorithm for multi-locations in road networks mainly consists of two steps: one is Voronoi-based trajectory simplification; the other one is Voronoi-based connected trajectory search. The following is the detail explanation.
3.2.1 Voronoi-based trajectory simplification

Su et al [9] has proposed that using trajectory calibration to solve trajectory heterogeneous and redundancy, it needs to find a stable reference system to transform a heterogeneous trajectory to one with unified sampling strategies. As a reference system, road network nodes are natural choices, it saves the path of trajectory and reduces redundant information at the same time. To find the nearest neighbor road node of sampling points quickly, this paper uses Delaunay triangulation network to build the Voronoi Diagram [10] of road nodes, and to index by an R-tree [11]. It’s able to efficiently find the closest road node with respect to a sampling point by using the nearest neighbor search, then to finish trajectory simplification. Here, not every sampling point will be shifted. Instead, it decreases the search times by saving the latest used node’s Voronoi polygon, and combines continuous same road network nodes to minish the simplified trajectory dataset. All these procedures only needs to be done one time off-line. Algorithm 1 shows the pseudo-code of Voronoi-Based Trajectory Simplification (VBTS).

3.2.2 Voronoi-based connected trajectory search

After the trajectory simplification, it comes out a simplified trajectory set of the original trajectory. We create an index from a road network node to the simplified trajectories and store it in HashMap. The purpose is to accelerate trajectory query in every road node. Since the query locations of user input may be arbitrary, we will first map every query point to nearest road node based on the Voronoi diagram of road network node, which is called query node. Then, using each query node to find out the trajectories which closed to query node in the given distance. Finally, the resulting set will be the intersection of the candidate trajectory set of each query node. The paper based on the observation of literature [2]: the number of query location will not be too large in each query, basically no more than ten. When it comes to more locations, the method in this paper can still be useful. Algorithm 2 shows the pseudo-code of Voronoi-based Connected Trajectory Search (VBCT).

3.3 Performance analysis

This paper proposes the Voronoi-based trajectory search algorithm for multi-locations in the road networks, which is based on simplified trajectory to query. Given a road network graph containing $V$ vertices and $E$ edges, trajectory dataset contains $N$ sampling points, $Q$ query locations, each trajectory has $i$ sampling points and $j$ roads segment on average. Traditional method needs to index all sampling points in an R-tree, and its height is $h \in \lfloor \log_M N - 1 \rfloor, \lfloor \log_m N - 1 \rfloor$ ($m, M$ are R-tree nodes entity’s the minimum and maximum number respectively). We assume that the number of sampling points and road nodes which are accessed by each query location on average is $k$, $f \in [1, j * k/i]$ within a given distance, the time complexity of the traditional algorithm is $O(Q * i * h)$. Then, the time complexity of the method in this paper is $O(Q * \log_m V + f)$. Because the $V$ is far less than the sampling points of trajectory dataset $N$, so $h$ is smaller than $\log_m V$. During the spatial range query in every query location, $i$ is greater than $f$ by network traversal. Because there may exist two cases: one is the multiple sampling points of a trajectory are transformed into the same road network node, the other one is the multiple sampling points of different trajectories are transformed into the same road network nodes.
Algorithm 1 VBTS

Input: \( V = \{v_1, v_2, \ldots, v_n\} \), \( T = \{p_1, p_2, \ldots, p_m\} \)

Output: The simplified trajectory \( T' = \{v_1, v_2, \ldots, v_k\} \)

1: for all \( v_i \in V \) do
2:    use a Delaunay triangulation method to create Voronoi Polygon (VP) for \( v_i \)
3:    use \( v_i \)'s Voronoi Polygon building R-tree
4: end for
5: for all \( p_j \in T \) do
6:   if \( \text{not-insides}(p_j, VP_{cur}) \) then
7:      the candidate set of the current sampling point \( VP_j = \text{null} \)
8:      \( VP_j = \text{search}(p_j) \)
9:      for all \( VP_{jk} \in VP_j \) do
10:         if \( \text{insides}(p_j, VP_{jk}) \) then
11:            \( VP_{cur} = VP_{jk} \), \( v_{cur} = N \)
12:               \( T'\text{List.add}(N) \)
13:         end if
14:      end for
15:   end if
16: end for
17: return \( T' \)

Algorithm 2 VBCT

Input: \( N = (V, E), Q, \text{distance} \)

Output: The result set \( VBCT \)

1: for all \( q_m \in Q \) do
2:   translate \( q_m \) to the counterpart vertex \( n_m \) in road networks
3:  \( Q\text{List.add}(n_m) \)
4: end for
5: if \( 0 == \text{distance} \) then
6:   for all \( n_i \in Q\text{List} \) do
7:      \( C_i \leftarrow \text{trajectories search by index } n_i \)
8: end for
9: \( VBCT \leftarrow C_1 \cap C_2 \cap \cdots \cap C_m \)
10: else
11:   if \( \text{distance} > 0 \) then
12:      for all \( n_i \in Q\text{List} \) do
13:         compute \( S_i \) in the range of distance
14:         for all \( v_j \in S_i \) do
15:            \( C_i \leftarrow \text{trajectories search by index } v_j \)
16:       end for
17:   end for
18: \( VBCT \leftarrow C_1 \cap C_2 \cap \cdots \cap C_m \)
19: end if
20: end if
21: return \( VBCT \)
4 Experimental Evaluation

In this section, we evaluate the efficient of the Voronoi-based trajectory search algorithm for multi-locations. Experiments use real road network map Oldenburg (Old) which consists of 6105 nodes and 7035 edges. We use Brinkoff generator simulator to generate the different sizes of synthetic dataset in this map, as shown in Table 1, dataset8 is consists of dataset3 and dataset7. The road network dataset and simulator can be shown in [16]. The experiments are executed on a PC with an Intel CPU of 3.20 GHz and 4GB memory. In the experiments, VBCT queries are collected manually by selecting a set of interesting places. We don’t use random generation for query locations as it may cause a sudden jump from one to anther far away location that probably won’t happen in real life.

We are going to evaluate the efficiency of this algorithm from two sides: trajectory simplification and connected trajectory search. At first, we consider simplification processing time and simplification rate to evaluate the efficiency of VBTS algorithm. Simplified rate is defined as ratio of the original trajectory set $D$ and the simplified $D'$ size, $|D| / |D'|$. Then, we evaluate the performance of VBCT algorithm according to query time and scalability. In order to verify the efficiency of VBCT, we slightly modify the algorithm in [2] to make it suitable for our scene where we search trajectory for multi-locations in a given distance, which is called connected trajectory search (CT). The distance threshold won’t be too big, because it will lead to our search meaningless. The default parameters are as follows:

- The number of query locations: 2 to 10, default 8.
- The distance threshold: 0 to 2000, default 500.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Trajectory Number</th>
<th>Sampling Points</th>
<th>Speed Parameter</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset1</td>
<td>2100</td>
<td>350000</td>
<td>250</td>
<td>13.9</td>
</tr>
<tr>
<td>dataset2</td>
<td>3100</td>
<td>480000</td>
<td>250</td>
<td>19.3</td>
</tr>
<tr>
<td>dataset3</td>
<td>5100</td>
<td>989000</td>
<td>250</td>
<td>39.2</td>
</tr>
<tr>
<td>dataset4</td>
<td>5050</td>
<td>1370000</td>
<td>1000</td>
<td>72.2</td>
</tr>
<tr>
<td>dataset5</td>
<td>5015</td>
<td>2920000</td>
<td>1000</td>
<td>116</td>
</tr>
<tr>
<td>dataset6</td>
<td>6100</td>
<td>3490000</td>
<td>1000</td>
<td>138</td>
</tr>
<tr>
<td>dataset7</td>
<td>9013</td>
<td>5670000</td>
<td>1000</td>
<td>223</td>
</tr>
<tr>
<td>dataset8</td>
<td>14113</td>
<td>6600000</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

4.1 Efficiency of trajectory simplification method

The experiments are simulated on synthetic trajectory dataset1 to dataset7. Firstly, to evaluate the performance of the VBTS algorithm by using these datasets. In Fig. 1, the simplified processing time is positive correlation with the size of trajectory dataset, because the bigger dataset size, the more sampling points need to be processed, but it only needs to run off-line
one time. In Fig. 2, we can see that the simplification rate is irrelevant to the dataset size. Combining with Table 1, we can find that the trajectory datasets with same sampling rate have similar simplification rate. With the increase in sampling rate, simplification rate is gradually increasing. This is because the increase of sampling rate leads to more sampling points to be simplified. Note that the sampling ratio is positive correlation with the speed parameter.

Fig. 1: The effience of VBTS

Fig. 2: The simplification rate of VBTS

4.2 Different number of query locations in VBCT

In this paper, a key assumption for this application scenario is to suppose the number of query locations is small, as it’s impractical to input tens of locations for a query. In fact, even if the number of query locations increases to 100 or more, this algorithm is proved to be executed well. But in this case, there is no great practical significance. In this part, we choose dataset8 as the experimental dataset.

In Fig. 3, with more and more query locations, the query time of two algorithms is increasing. But VBCT is obviously better than CT algorithm, because CT needs to scan more sampling points of trajectory, the VBCT method only need to access several road network nodes in the given distance. It will lead to the time complexity of the algorithm decreasing greatly.

Fig. 3: Varying the number of query points

Fig. 4: Varying distance threshold value

4.3 Different sizes of distance constraint in VBCT

This part uses dataset8 as experiment’s data to evaluate the influence of distance threshold to query algorithm. In Fig. 4, with the increasing in distance, the query time both are increasing.
But our method is better than the CT, especially, when the distance is very large. Because CT needs to access more sampling point in the Euclidean space of range query. VBCT algorithm needn’t to expend so many road nodes, thus it greatly reduces the query time. Due to people tend to get results which are closed to every query location, the distance threshold value can not be set too large in general. Otherwise, it will lose the significance of the query.

4.4 Different sizes of trajectory scale in VBCT

This part evaluates the scalability of VBCT query algorithm through the experiment. In the experiment, with the increasing number of trajectory, trajectory sampling points become very large. The CT method using R-tree to index needs to take a long time to build all the index of trajectory sampling points, and the query time also becomes longer with the increase of data objects. Here, it only shows VBCT method with different sizes of trajectory scale, as shown in Fig. 5, with the increasing of trajectory dataset, the cost of query time becomes longer, but the overall query time overhead maintains in a relative small level. The large trajectory set will make each road network node to index more trajectories, but the range query time doesn’t increase in each query node. The experiment demonstrates that our method has good scalability. When the trajectory dataset is large enough, we only need to specify the distance threshold value to 0. By this operation, the time spent in the range query will be reduced.

![Fig. 5: The scalability of VBCT](image)

5 Conclusion

This paper proposed a new approach to solve the trajectory search algorithm for multi-locations based on Voronoi diagram in road networks, considering the characters that trajectory has some redundancies and it is restricted by road network. Due to Voronoi diagram has a good performance in pre-computing the adjacent of object, this paper uses it to simplify the trajectory, and it can greatly reduce the size of trajectory dataset and search the candidate trajectory. Then, using the node of road network which is the nearest neighbor of query location for network range query, and getting trajectories which can connect query locations. Hence, our solution is not only suitable for searching trajectories for multi-locations in road networks, but also greatly improving the query efficiency. Experiments show that the proposed method is more efficient than before. In the future, we can also make query on time period, orderly query and Top-$k$ best-connected trajectory query.
References


