Short Paper

Fast Trajectory Range Query with Discrete Fréchet Distance

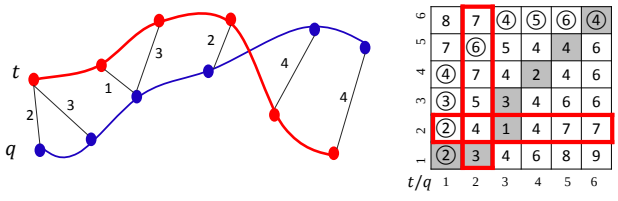
Jiahao Zhang[†‡] Bo Tang[†] Man Lung Yiu [‡]

[†] Department of Computer Science and Engineering, Southern University of Science and Technology
tangb3@sustc.edu.cn

[‡]Department of Computing, The Hong Kong Polytechnic University
{csjhzhang,csmlyiu}@comp.polyu.edu.hk

ABSTRACT

The discrete Fréchet distance (DFD) is widely used to measure the similarity between two trajectories. Trajectory range query has been extensively studied in trajectory analytical applications, e.g., outlier detection, movement pattern analysis. With the discrete Fréchet distance, the above applications are computation bound rather than disk I/O bound. In this work, we propose new lower and upper bound functions to speedup the evaluation of trajectory range queries. Experimental studies on three real datasets demonstrate the superiority of our proposal.



(a) DFD illustration (b) DFD computation

Figure 1: DFD illustration and computation

1 INTRODUCTION

The range query problem on trajectory dataset is a core subroutine in trajectory analytical applications, e.g., anomaly monitoring [3], traffic analysis [4]. In particular, given a query trajectory q and trajectory database T, the trajectory range query problem returns a subset $S \subseteq T$ such that for each $t \in S$, the distance between t and q is within given threshold θ, i.e., $dist(t, q) \leq \theta$. Specifically, we use the discrete Fréchet distance (DFD) d_F, a widely used trajectory similarity measure [1, 5]. Given two trajectories $t = \langle t[1], t[2], \cdots, t[n]\rangle$ and $q = \langle q[1], q[2], \cdots, q[m]\rangle$, a coupling L between t and q is a sequence $\langle (t[a_1], q[b_1]), (t[a_2], q[b_2]), \cdots, (t[a_l], q[b_l])\rangle$, where $a_1 = 1, b_1 = 1, a_l = n, b_l = m$, and for all $i = 1, \cdots, l$, we have $a_{i+1} = a_i$ or $a_{i+1} = a_i + 1$, and $b_{i+1} = b_i$ or $b_{i+1} = b_i + 1$. The discrete Fréchet distance between t and q is defined as

$$d_F(t, q) = \min_{L \in \Omega} \max_{(t[a_i], q[b_i]) \in L} ||t[a_i] - q[b_i]||,$$

where Ω is the set of all possible couplings between t and q.

Figure 1(a) illustrates the discrete Fréchet distance (DFD) between two trajectories t and q. Computing discrete Fréchet distance between t and q is equivalent to find a path from $(t[1], q[1])$ to $(t[n], q[m])$ in a free space diagram such that (i) the path travels along non-decreasing positions, and (ii) the maximum $||t[a_i] - q[b_i]||$ along the path is minimized [5]. For instance, the value of $d_F(t, q)$ in Figure 1 is determined by the gray path in Figure 1(b). Since it can be computed via dynamic programming, the time complexity of DFD computation is $O(mn)$.

In the literature, several lower and upper bounds have been proposed for DFD [2, 5]. We will introduce these bounds in Section 2. Nevertheless, the trajectory range query problem is still a computationally intensive problem, taking $O(|T|mn)$ time, where $|T|$ is the cardinality of trajectory dataset. To improve the query performance, we devise new lower and upper bound functions in this paper. Our experimental studies on real datasets reveal that our proposal improves the range query performance on DFD by up to an order of magnitude when compared to a baseline approach based on existing techniques.

The remainder of this paper is organized as follows. Section 2 formulates trajectory range query problem and presents the baseline solution adapted from existing works. We present our novel techniques in Section 3. Section 4 demonstrates the efficiency of our proposal with experiments on real datasets. Finally, we conclude the paper in Section 5.

2 PRELIMINARIES

In this section, we first define the trajectory range query problem in Section 2.1, then we present the baseline solution for it by adapting existing techniques in Section 2.2.

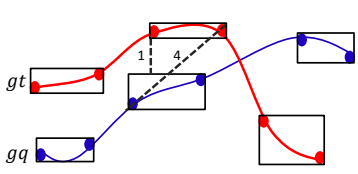
2.1 Problem Definition

Definition 2.1 (Trajectory Range Query Problem). Given a query trajectory q, and a trajectory dataset T and distance threshold θ, the trajectory range query returns a subset $S \subseteq T$ such that for each trajectory $t \in S$, the discrete Fréchet distance (DFD) between t and q is at most θ, i.e., $d_F(t, q) \leq \theta$.

A straightforward approach for this problem (cf. Definition 2.1) is to compute the exact discrete Fréchet distance (DFD) between each trajectory $t \in D$ and the query trajectory q. Its time complexity is $O(|T|mn)$, rendering it impractical for a large trajectory dataset.

2.2 Baseline Solution

In this section, we briefly introduce a baseline approach for the trajectory range query problem (cf. Definition 2.1) which employs existing techniques in the literature. It follows the filter-and-refinement paradigm. Let $LB(t, q)$ and $UB(t, q)$ denote the lower and upper bounds of $d_F(t, q)$ respectively, i.e., $LB(t, q) \leq d_F(t, q) \leq UB(t, q)$. A candidate t cannot be a result if $LB(t, q) > \theta$. A candidate t is definitely a result if $UB(t, q) \leq \theta$. In these two cases, we save an expensive exact DFD computation.



(a) Group-based trajectory

	1	2	3
3	5, 10	2, 6	2, 6
2	3, 8	1, 4	2, 6
1	1, 4	0.5, 6	7, 10

gt/gq

(b) Group-based bounds

Figure 2: Group-based DFD illustration and computation

We introduce existing lower and upper bounds as follows.

Cell-based lower bound LB_{cell} (from [5]): The cell-based lower bound is defined as

$$LB_{cell}(t, q) = \max(||t[1] - q[1]||, ||t[n] - q[m]||).$$

The idea of LB_{cell} is that any coupling L between q and t must start from $(1, 1)$ and end at (n, m). Its computation cost is $O(1)$.

Row-based lower bounds LB_{row} and LB_{cross} (from [5]): The row-based and column-based lower bounds are defined as follows:

$$LB_{row}(t, q) = \min_{b_i \in [1,m]}(||t[2] - q[b_i]||).$$
$$LB_{col}(t, q) = \min_{a_i \in [1,n]}(||t[a_i] - q[2]||).$$

The path leading to DFD pass through the second column and row certainly contribute these two bounds. Both of them have $O(n)$ time complexity.

For example, the row-based and column-based lower bounds for trajectory t and q are $LB_{row}(t, q) = \min\{2, 4, 1, 4, 7, 7\} = 1$ and $LB_{col}(t, q) = \min\{7, 6, 7, 5, 4, 3\} = 3$, respectively.

By combining row-based and column-based lower bounds, we obtain a cross-based lower bound as follows.

$$LB_{cross}(t, q) = \max(LB_{row}(t, q), LB_{col}(t, q)).$$

Group-based bounds LB_g and UB_g (from [5]): The idea is to partition a trajectory t (in Figure 1(a)) to a grouped trajectory $gt = \{gt[1], gt[2], gt[3]\}$ (in Figure 2(a)), which can be represented by a sequence of minimum bounding rectangles (MBRs). The length of group-based trajectory is denoted as τ. The minimum and maximum distance between $gt[a_i]$ and $gq[b_i]$ (two MBRs) are denoted as $d_G^{lb}(gt[a_i], gq[b_i])$ and $d_G^{ub}(gt[a_i], gq[b_i])$.

They can be derived in $O(1)$ cost. For example, in Figure 2 $d_G^{lb}(gt[2], gq[2])$ and $d_G^{ub}(gt[2], gq[2])$ are 1 and 4, respectively.

The DFD computing procedure with group-based trajectory adopts the minimum or maximum distances between MBRs leads a lower or upper bound of $d_F(t, q)$, as follows:

$$LB_g(t, q) = \min_{L \in g\Omega} \max_{(gt[a_i], gq[b_i]) \in L} d_G^{lb}(gt[a_i], gq[b_i]),$$

$$UB_g(t, q) = \min_{L \in g\Omega} \max_{(gt[a_i], gq[b_i]) \in L} d_G^{ub}(gt[a_i], gq[b_i])$$

As shown in Figure 2(b), the lower and upper bound DFD distance between group-based trajectory gt and gq are $LB_g(t, q) = 2$ and $UB_g(t, q) = 6$, respectively. It reduces computation cost as $O(\tau^2) < O(mn)$.

Greedy-based upper bound UB_{greedy} (from [2]): In addition to the above bounds, [2] proposed a greedy algorithm to compute an upper bound for $d_F(t, q)$ with $O(n)$ cost.

The idea of greedy algorithm is simple, e.g., the discrete Fréchet distance (DFD) goes as follows: in every step, make the move that minimizes the current distance, where a "move" is a step in

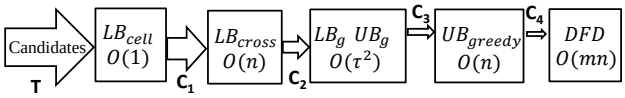


Figure 3: Baseline approach for trajectory range query

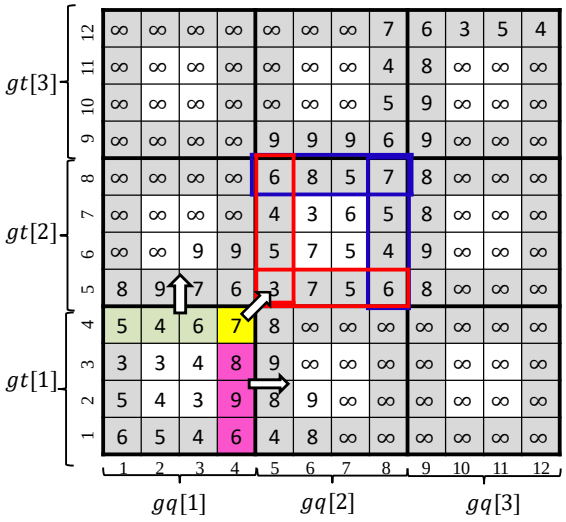


Figure 4: Running example

either one sequence or in both of them. The greedy algorithm guarantees $2^{\Theta(n)}$-approximation of the discrete Fréchet distance with linear time cost $O(n)$. As shown in Figure 1(b), the greedy algorithm produces the path shown as the circled numbers.

Baseline approach: We construct the baseline approach by exploiting existing techniques. It applies a filter-and-refinement framework, as shown in Figure 3. It computes exact DFD distance for a candidate t only if it cannot be dismissed by lower bounds (i.e., LB_{cell}, LB_{cross}, LB_g) or upper bounds. Specifically, the computation cost of employed bounds are in an order from quick-but-dirty one to slow-but-accurate one.

3 PROPOSED OPTIMIZATIONS

To improve the performance of trajectory range query (cf. Definition 2.1), we propose several novel optimizations in this section.

3.1 Group-based border bounds

The lower bound distance between two grouped trajectories gt and gq is $LB_g(t, q) = \min_{L \in g\Omega} \max_{(gt[a_i], gq[b_i]) \in L} d_G^{lb}(gt[a_i], gq[b_i])$, where $d_G^{lb}(gt[a_i], gq[b_i])$ is the lower bound of ground distance between $gt[a_i]$ and $gq[b_i]$, shown as the dashed line with the label 1 in Figure 2(a). The effectiveness of the group-based lower bound pruning depends on term $LB_g(t, q)$.

Group-based border lower bound. Instead of using $d_G^{lb}(gt[a_i], gq[b_i])$ in baseline approach, we devise a tighter lower bound for ground distance between two groups $(gt[a_i], gq[b_i])$ by the crucial observation as follows. Consider $(gt[2], gq[2])$ in Figure 4, the minimum distance of $(gt[2], gq[2])$ is determined by the gray cells, as if the DFD path of trajectory t and q passes though the group pair $(gt[2], gq[2])$, $d_F(t, q)$ must be larger than

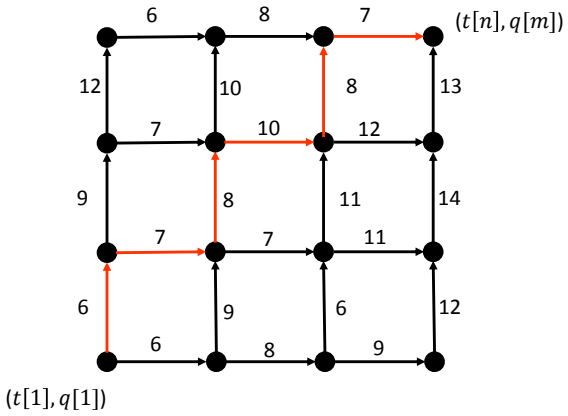


Figure 5: Group-based border upper bound

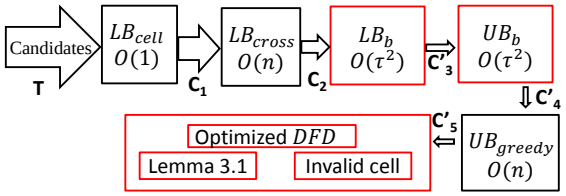


Figure 6: Advanced approach for trajectory range query

or equal to the minimum value in entrance cells (red rectangles) and exit cells (blue rectangles).

In particular, the minimum value of entrance cells and exit cells in given pair $(gt[a_i], gq[b_i])$ is defined as $d^{en}_{min}(gt[a_i], gq[b_i])$ and $d^{ex}_{min}(gt[a_i], gq[b_i])$, respectively. Thus, the entrance and exit cell based lower bound of ground distance of group pair $(gt[a_i], gt[b_i])$ is

$$d^{lb}_{ee}(gt[a_i], gq[b_i]) = \max\{d^{en}_{min}(gt[a_i], gq[b_i]), d^{ex}_{min}(gt[a_i], gq[b_i])\}.$$

Consequently, we have a tighter lower bound $LB_b(t, q)$ by

$$LB_b(t, q) = \min_{L \in g\Omega} \max_{(gt[a_i], gq[b_i]) \in L} d^{lb}_{ee}(gt[a_i], gq[b_i]).$$

There is a trade-off between the tightness of $d^{en}_{min}(gt[a_i], gt[b_i])$ (the same as $d^{ex}_{min}(gt[a_i], gt[b_i])$) and its computation cost. The quick-and-loose one is taking the smaller distance of $t[\frac{a_i n}{\tau}]$ with the MBR of $gq[b_i]$ and the MBR of $gt[a_i]$ with $q[\frac{b_i m}{\tau}]$. The slow-and-tight one is computing the smallest distance of $t[\frac{a_i n}{\tau}]$ with all points (i.e., 3, 7, 5, 6 in red rectangle) in $gq[b_i]$ and all points in $gt[a_i]$ with $q[\frac{b_i m}{\tau}]$ (i.e., 3, 5, 4, 6 in red rectangle). In terms of time complexity, the former one is $O(\tau^2)$ and the later one is $O(\tau n)$ for every group pairs in gt and gq.

Group-based border upper bound. As shown in Figure 5, the upper bound of ground distance between $t[\frac{a_i n}{\tau}]$ with the MBR of $gq[b_i]$ and the MBR of $gt[a_i]$ with $q[\frac{b_i m}{\tau}]$ for each $(gt[a_i], gq[b_i])$ pair are computed with $O(\tau^2)$ cost. The upper bound of the DFD path from $(t[1], q[1])$ to $(t[n], q[m])$ is equivalent to find a path from left-bottom corner to right-top corner, where the maximum value in that path is minimized, as the red path shown.

We denote that DFD upper bound as $UB_b(t, q)$, it is 10 in the example of Figure 5.

3.2 DFD path transfer directions

Take $(gt[1], gq[1])$ in Figure 4 as an example, the minimum distance between $gt[1]$ and $gq[1]$ is $\max\{\min\{3, 4\}, \min\{4, 6\}\} = 4$. Computing $LB_g(t, q)$ is equivalent to find a DFD path from $(gt[1], gq[1])$ to $(gt[\tau], gq[\tau])$ among the minimum distances of each group pairs, i.e., $d^{lb}_G(gt[a_i], gq[b_i])$. Even the minimum distance of each group pair is improved by entrance and exit cell bounds as above, it still can be optimized by exploiting the path transfer direction. For example, as shown in Figure 4, if the path is from $(gt[1], gq[1])$ to $(gt[1], gq[2])$, the lower bound contributes from $(gt[1], gq[1])$ is the minimum value among the magenta cells and yellow cells. Similarly, The path from $(gt[1], gq[1])$ to $(gt[2], gq[2])$ must pass the yellow cell. Obviously, the minimum value from $(gt[1], gq[1])$ to $(gt[1], gq[2])$, and from $(gt[1], gq[1])$ to $(gt[2], gq[2])$ are 6 and 7, which is larger than 4, i.e., the distance between $gt[1]$ and $gq[1]$ by entrance and exit cells bound. In summary, the transfer-based minimum values are used to tight the group-based DFD lower bound (i.e., $LB_b(t, q)$) during the DFD computation on grouped trajectories gq and gt.

3.3 DFD computation acceleration

As illustrated in Figure 3, for these candidate trajectories cannot be pruned by its lower bound or be detected as true results, it will incur expensive DFD computation $O(mn)$. Specifically, it computes the distance between any two points among t and q, i.e., each cell in Figure 4, then computes the $d_F(t, q)$ via dynamic programming. In the subsequential section, we devise two novel techniques, namely early termination and invalid cell ignoring, to reduce the DFD computation cost.

Early termination. We define $layer_{i,j}$ among trajectory t and q (cf. Figure 4) as either $t[i]$ or $q[j]$ are considered in that cells. Formally, the lower bound of $layer_{i,j}$ is defined as:

$$LB^{i,j}_{layer}(t, q) = \min\{\min_{k \in [1,j]}\{||t[i]-q[k]||\}, \min_{k \in [1,i]}\{||t[k]-q[j]||\}\}.$$

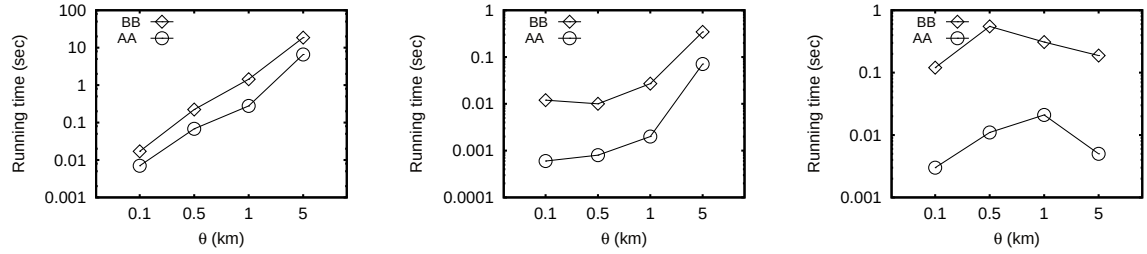
Lemma 3.1. $LB^{i,j}_{layer}(t, q) \le d_F(t, q)$

Proof. The proof is trivial as the path from $(t[1], q[1])$ to $(t[n], q[m])$ must pass through the at least one cell of $layer_{i,j}$, thus, $d_F(t, q)$ is not smaller than the minimum value in $layer_{i,j}$. □

Thus, we terminate DFD computation if $LB^{i,j}_{layer}(t, q) \ge \theta$ with Lemma 3.1. Suppose the distance threshold $\theta = 2$, consider the $layer_{3,3}$ in Figure 4, $LB^{3,3}_{layer}(t, q) = \min\{3, 3, 4, 3, 4\} = 3$. We terminate the exact DFD computation earlier as $LB^{3,3}_{layer}(t, q) > \theta$. It improves computation cost by ignoring a lot of ground distance pair computation and the exact DFD computation.

Invalid cell ignoring. Consider the example in Figure 4, suppose the distance threshold $\theta = 7$. Given cell $(t[i], q[j])$ its distance computation can be avoided if $||t[i], q[j-1]||$, $||t[i-1], q[j]||$, and $||t[i-1], q[j-1]||$ are larger than θ. We define such kind of cells as invalid cells, set their distances as ∞, as shown in Figure 4 without incurring expensive distance computation.

Put all it together: We present the framework of our advanced approach (AA) for trajectory range query (with discrete Fréchet distance) in Figure 6. Our proposed techniques are enclosed by red rectangles, e.g., $LB_b(t, q)$, $UB_b(t, q)$, and optimized exact DFD computation.



(a) **GeoLife** (b) **OSM-DE** (c) **OSM-FULL**

Figure 7: Response time vs. distance threshold θ

4 EXPERIMENTAL EVALUATION

We evaluate the performance of the baseline approach (BB, cf. Section 2) and our advanced approach (AA, cf. Section 3) for trajectory range queries. In our experiments, we report the average measurements over 10 different query trajectories.

Dataset: We used three real world trajectory datasets with diverse geographical coverage and scales of sizes.

GeoLife.[1] The GeoLife project in Microsoft Research Asia collected this dataset from 182 users over five years (April 2007-August 2012). It has 18,655 trajectories and 24.9 million GPS points.

OSM-FULL.[2] This dataset contains 7.5 years of OpenStreetMap trajectory data around the world. It includes total 2.4 million trajectories constructed by 2.7 billion GPS points.

OSM-DE. It is a subset of **OSM-FULL**. We extracted trajectories that are located in Germany. **OSM-DE** has the highest data density among all regions in **OSM-FULL** [6]. It has 0.5 million trajectories and 0.5 billion GPS points in total.

We used C++ for the implementation and conducted all experiments (with single thread) on a machine with AMD A10-7850K 3.70GHz processor and 16GB main memory.

Overall performance evaluation: We compare the performance of BB and AA for trajectory range query problem on all three real world datasets by varying distance threshold in Figure 7. AA is faster than BB by 2.42 to 50.1 times. In addition, the performance gap between AA and BB becomes large with the rise of distance threshold θ. It also confirms AA is scalable. The trajectories in **GeoLife** are much denser than other two datasets, and that is the reason why range queries perform slower though **OSM-FULL** and **OSM-DE** have large sizes.

Effect of optimizations: We then evaluate the effectiveness of our proposed optimization techniques, e.g., group border-based bound, path transfer direction-aware bound, etc. Due to space limitation, we omit the experiment results on **OSM-DE** and **OSM-FULL** as they are similar to **GeoLife**. The group-size τ is 8 in all the experiments.

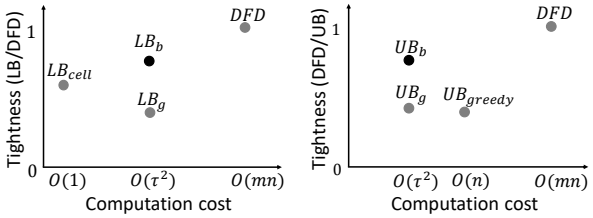
Table 1 illustrates the number of DFD executions (per query) of BB and AA with regard to distance threshold θ, respectively. Obviously, our advanced approach outperforms baseline approach.

We then investigate the effectiveness of DFD lower and upper bounds. Our proposed LB_b is much better than LB_g while both with $O(\tau^2)$ cost. In addition LB_b with $O(\tau n)$ cost is slightly better than with $O(mn)$, as shown in Figure 8(a). UB_b is better than UB_{greedy} and UB_g as illustrated in Figure 8(b).

[1]https://bit.ly/2E4sntq
[2]https://bit.ly/2zwrKVH

Table 1: DFD **executions on GeoLife dataset**

θ	0.1 km	0.5 km	1km	5 km
BB	6.60	48.97	125.33	550.01
AA	**1.87**	**21.15**	**19.32**	**88.1**



(a) Lower bounds (b) Upper bounds

Figure 8: Tightness of DFD **lower and upper bounds**

5 CONCLUSION

In this paper, we propose several novel techniques (e.g., group border-based bound, DFD computation acceleration) to speedup the trajectory range query problem on discrete Fréchet distance. Our advanced approach is up to 50 times faster than the baseline solution we construct from the literature. A promising direction for future work is to devise error-guaranteed approximate solutions for the trajectory range query problem.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation of China (NSFC No. 61802163) and Guangdong Natural Science Foundation (Grant No. 2018A030310129).

REFERENCES

[1] Pankaj K Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. 2018. Subtrajectory Clustering: Models and Algorithms. In *PODS*. 75–87.
[2] Karl Bringmann and Wolfgang Mulzer. 2015. Approximability of the discrete Fréchet distance. In *Journal of Computational Geometry*, Vol. 7. 46–76.
[3] Yingyi Bu, Lei Chen, Ada Wai-Chee Fu, and Dawei Liu. 2009. Efficient anomaly monitoring over moving object trajectory streams. In *KDD*. 159–168.
[4] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: a partition-and-group framework. In *SIGMOD*. 593–604.
[5] Bo Tang, Man Lung Yiu, Kyriakos Mouratidis, and Kai Wang. 2017. Efficient motif discovery in spatial trajectories using discrete fréchet distance. In *EDBT*. 378–389.
[6] Dong Xie, Feifei Li, and Jeff M Phillips. 2017. Distributed trajectory similarity search. *PVLDB* 10, 11 (2017), 1478–1489.