Efficient Motif Discovery in Spatial Trajectories Using Discrete Fréchet Distance

Bo Tang* Man Lung Yiu* Kyriakos Mouratidis# Kai Wang$

*The Hong Kong Polytechnic University #Singapore Management University $Zhejiang University

{csbtang,csmlyiu}@comp.polyu.edu.hk; kyriakos@smu.edu.sg; kaelwang@zju.edu.cn

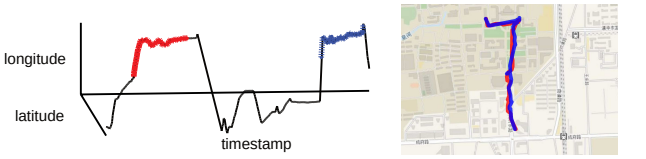
ABSTRACT

The discrete Fréchet distance (DFD) captures perceptual and geographical similarity between discrete trajectories. It has been successfully adopted in a multitude of applications, such as signature and handwriting recognition, computer graphics, as well as geographic applications. Spatial applications, e.g., sports analysis, traffic analysis, etc. require discovering the pair of most similar subtrajectories, be them parts of the same or of different input trajectories. The identified pair of subtrajectories is called a motif. The adoption of DFD as the similarity measure in motif discovery, although semantically ideal, is hindered by the high computational complexity of DFD calculation. In this paper, we propose a suite of novel lower bound functions and a grouping-based solution with multi-level pruning in order to compute motifs with DFD efficiently. Our techniques apply directly to motif discovery within the same or between different trajectories. An extensive empirical study on three real trajectory datasets reveals that our approach is 3 orders of magnitude faster than a baseline solution.

1. INTRODUCTION

Spatial trajectories are prevalent in many applications, e.g., moving object analysis, traffic estimation and prediction systems. In this paper, we study motif discovery on spatial trajectories (i.e., finding the pair of most similar subtrajectories). Trajectory motifs are used in many applications, e.g., sports sense analysis [11], traffic analysis [15], or used as a building block for other trajectory mining and analysis methods [16, 31, 12]. As an example, Figure 1(a) visualizes a pedestrian's GPS trajectory from the GeoLife trajectory dataset [32], by a 3D plot with timestamp number at the horizontal axis. The motif corresponds to the most similar pair of subtrajectories (in red and blue). Figure 1(b) illustrates the motif (i.e., the two subtrajectories) on a map, which could be used in human behavior analysis.

It is important to choose a suitable similarity measure for motif discovery. The Fréchet metric is amongst the most popular measures for trajectory similarity [24, 10]. Generally speaking, the Fréchet distance between two spatial trajectories, S_a and S_b, is the length of the shortest leash needed to walk a dog when the person walks along S_a and the dog walks along S_b. In the geographic information handbook [10], the authors conclude that *"The most successful fundamental distance measure to this date is probably the Fréchet metric, which is one of the most natural measures to calculate the similarity between two trajectories"*. The Fréchet distance and its variants have been successfully used in a number of application domains, such as handwriting recognition [22], bioinformatics [27], computational geometry [5], as well as geographic applications [2]. In the literature, many recent systems have adopted the discrete Fréchet distance (DFD) to measure the distance between discrete trajectories (or the Fréchet distance for continuous curves) [2, 10, 3, 12, 25]. In addition, as we will elaborate in Section 2, DFD is particularly suitable for real-world spatial trajectories, which often exhibit the following properties: (i) non-uniform/varying sampling rate, and (ii) missing samples at some time points. For example, the GeoLife dataset [32], a real spatial trajectory dataset collected by Microsoft, has all the above properties.



(a) A pedestrian's trajectory April 10-12, 2009

(b) Discovered motif red: 07:33-7:48, April 10, 2009 blue: 07:33-7:50, April 12, 2009

Figure 1: Subtrajectory motif discovered in a trajectory from the GeoLife trajectory dataset

In this paper, we discover motifs in spatial trajectories with DFD as the similarity measure. This problem is computationally challenging for two reasons:

(I) The computation of DFD between two subtrajectories takes $O(\ell^2)$ time [11], where ℓ denotes the subtrajectory length. There have been attempts to speed up DFD computation by using GPUs [12] or a faster algorithm (with $O(\ell^2 \cdot \frac{\log\log \ell}{\log \ell})$ time complexity) [1]. In contrast, we take an orthogonal research direction to reduce the number of DFD computations for motif discovery, e.g., by using various types of pruning on DFD computations and subtrajectory pairs.

(II) The problem involves $O(n^4)$ pairs of subtrajectories, where n is the length of the input trajectory/ies. The fact that DFD exhibits non-monotonicity (cf. Section 4.1) precludes us from applying efficient algorithmic paradigms (like binary search) to reduce the number of candidate pairs.

To overcome these challenges, we exploit the properties of DFD and devise lower bound functions that incur low computation time.

These lower bound functions serve for two purposes: first, pruning unpromising pairs of subtrajectories without invoking expensive DFD computation; second, guiding the search to discover the motif as soon as possible. Our lower bound functions are novel; nothing similar has been used in previous work. Additionally, they can be computed in amortized O(1) time.

Furthermore, we propose a grouping-based solution with multi-level pruning. This solution (i) divides the input trajectory into groups, (ii) prunes dissimilar pairs of groups, and then (iii) aggressively processes the surviving pairs of groups until the result is found. At the heart of this approach lies a suite of lower bound functions to prune unpromising pairs of groups.

All our techniques apply directly to motif discovery within the same or between different input trajectories. Importantly, besides motif discovery, they can be incorporated readily to other applications [2, 3] which employ DFD as the similarity measure.

2. RELATED WORK

In this section, we survey previous work but, due to the strict page limit, we focus only on the most relevant pieces. First, we present alternative similarity measures and pinpoint the advantages offered by the Fréchet metric that render it the ideal choice for (sub)trajectory similarity [10]. Next, we overview existing motif discovery approaches and juxtapose them to ours. Finally, we provide an outlook of other practically relevant trajectory analysis techniques.

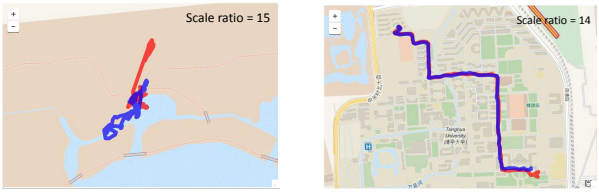
Trajectory similarity measures: Several similarity measures have been proposed for trajectories, e.g., Euclidean Distance (ED), Discrete Fréchet Distance (DFD) [8, 1], Dynamic Time Warping (DTW) [28], Longest Common Subsequence (LCSS) [26], Edit Distance on Real Sequence (EDR) [6]. Real-world trajectories (e.g., those in GeoLife dataset) exhibit two key characteristics, namely, non-uniform/varying sampling rate and missing samples for some time points. Thus, a desirable similarity measure would account for both these characteristics. In Table 1, we summarize the properties of the aforementioned trajectory similarity measures and their computation cost, expressed in terms of (sub)trajectory length ℓ. Local time shifting refers to the ability of tolerating short-term discrepancies (e.g., missing samples, measurement errors) in aligning two trajectories [6].

Distance metric	Non-uniform/varying sampling rate	Local time shifting	Computation cost
ED			$O(\ell)$
DTW		$\surd$	$O(\ell^2)$
LCSS		$\surd$	$O(\ell^2)$
EDR		$\surd$	$O(\ell^2)$
DFD	$\surd$	$\surd$	$O(\ell^2)$

Table 1: Distance measures and their characteristics

We will use examples to illustrate the advantages of discrete Fréchet Distance (DFD) over typical alternatives (e.g., ED, DTW). We first apply two different measures (ED and DFD) to compute motifs on the GeoLife trajectory dataset [32]. Figures 2(a) and 2(b) show the most similar pair of subtrajectories by ED and DFD, respectively. Observe that the result of DFD (in Figure 2(b)) captures much better a human's interpretation. The reason is that ED measures spatial proximity only, and dismisses the movement pattern.

In Figure 3, we demonstrate the effect of non-uniform sampling in real-world data using DTW and DFD between trajectories S_a, S_b and S_a, S_c. Trajectories S_a (black color) and S_b (blue color) are uniformly sampled, while trajectory S_c (red color) is



(a) Most similar pair in ED
ED: 8.71 m; DFD: 0.09 m

(b) Most similar pair in DFD
ED: 19.42 m; DFD: 0.08 m

Figure 2: ED and DFD

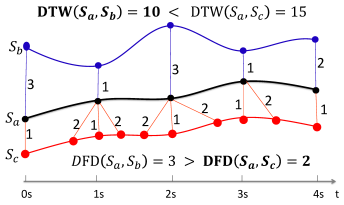


Figure 3: DTW and DFD; S_c is non-uniformly sampled

non-uniformly sampled. Intuitively, trajectory S_c is more similar to S_a than S_b, i.e., $\text{DFD}(S_a, S_c) < \text{DFD}(S_a, S_b)$, however, $\text{DTW}(S_a, S_c) > \text{DTW}(S_a, S_b)$. The reason is DTW requires each point to be matched to another (and adds up all distances between matched pairs) thus being sensitive to non-uniform sampling.

In summary, ED is the fastest metric to compute but it is not robust to local time shifting. More robust measures, such as DTW [28], LCSS [26], EDR [6], are defined as the sum of point-to-point distances, which makes them sensitive to the sampling rate. As shown in Table 1, only DFD [8, 1], also known as the "dog-man" distance, can tolerate non-uniform/varying sampling rate [11, 24, 12]. Other distance measures require that points along the trajectories are uniformly and densely sampled, which is rarely the case in real settings [11]. For more details on trajectory similarity measures, we refer the reader to surveys [10, 24, 7].

The parallel computing [12] and computational geometry [1] communities have proposed some techniques to speed up DFD computation. In contrast, in our work we take an orthogonal approach to accelerate DFD computations for trajectory motif discovery via novel pruning techniques.

Trajectory motif discovery techniques: For spatial trajectories, most of the motif discovery techniques adopt the *symbolic approach* [17, 11, 20]. This approach employs symbols to represent pre-defined movement patterns; some example symbols and patterns are illustrated in Figure 4(a). To convert a trajectory into a string of symbols, it first partitions a trajectory into fragments, and then maps each fragment to a symbol (i.e., pre-defined movement pattern). After that, it applies substring matching techniques to discover motifs [30, 14]. Unfortunately, this approach may produce similar strings even if their original trajectories are far apart. For example, we illustrate two trajectories of Uber drivers (in two different cities) in Figures 4(b) and 4(c). Although these two trajectories are geographically far apart (in two different cities), both of them are mapped to string 'RVLH'. Since this approach cannot capture the spatial distance between trajectories, we dismiss it.

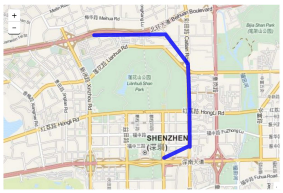
Motifs have also been studied for time series data [19, 18]. However, these techniques are tailored to time series with Euclidean distance, and are not suitable for spatial trajectories with DFD.

Other trajectory analysis techniques: Besides motif discovery, there are many other spatial trajectory analysis problems, e.g., convoy discovery [13], outlier detection [29, 17], trajectory cluster-

symbol	movement pattern
V	vertical long straight
H	horizontal long straight
L	left turn
R	right turn

(a) Pre-defined movement patterns and symbols

(b) A trajectory in Beijing; string: RVLH



(c) A trajectory in Shenzhen; string: RVLH

Figure 4: Example of the symbolic approach

ing [16, 11, 12], etc. We refer the interested reader to a recent survey [31].

3. PROBLEM STATEMENT

In this section, we introduce the problem and present a baseline solution, starting with several basic definitions.

DEFINITION 1 (SPATIAL TRAJECTORY & SUBTRAJECTORY). *A spatial trajectory $\mathcal{S} = \langle \cdots, s_i, \cdots \rangle$ is a sequence of points. We denote its trajectory length by $n = |\mathcal{S}|$.*

Given a trajectory $\mathcal{S}$, we denote a subtrajectory of $\mathcal{S}$ as $\mathcal{S}_{i,i_e} = \mathcal{S}[i..i_e]$, where $0 \le i < i_e \le n-1$.

Let $T(\mathcal{S}) = \langle \cdots, t_i, \cdots \rangle$ be a sequence of ascending timestamps, where t_i is the timestamp of location s_i in $\mathcal{S}$. The timestamps may be non-uniform.

We assume each point s_i is a latitude-longitude (φ_i, λ_i) pair. We measure the *ground distance* between two trajectory points $s_i = (\varphi_i, \lambda_i), s_j = (\varphi_j, \lambda_j)$ as the great circle distance on Earth [21]:

$$d_G(i,j) = 2R \arcsin \sqrt{\sin^2\left(\frac{\varphi_j - \varphi_i}{2}\right) + \cos\varphi_i \cos\varphi_j \sin^2\left(\frac{\lambda_j - \lambda_i}{2}\right)}$$

where R is the radius of the earth. Nevertheless, our methods are directly applicable to higher dimensions (e.g., 3-d data points) and other types of ground distance (e.g., Euclidean).

As discussed in Section 2, we adopt the discrete Fréchet distance (DFD) to measure the distance between two subtrajectories $\mathcal{S}_{i,i_e}$ and $\mathcal{S}_{j,j_e}$, defined as:

$$d_F(i,i_e,j,j_e) = \max \begin{cases} d_G(i_e,j_e) \\ \min \begin{cases} d_F(i,i_e-1,j,j_e), \\ d_F(i,i_e,j,j_e-1), \\ d_F(i,i_e-1,j,j_e-1) \end{cases} \end{cases}$$

For $i_e = i$ and $j_e = j$, $d_F(i,i,j,j) = d_G(i,j)$, and the DFD computation recursion terminates at $i_e = i$ and $j_e = j$.

We study the motif discovery problem within a single input trajectory or between different trajectories; for simplicity, we focus presentation on single input trajectory but also elaborate on (and evaluate) the latter variant too. To produce a meaningful trajectory motif $(\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e})$, we require that: (i) subtrajectories $\mathcal{S}_{i,i_e}$ and $\mathcal{S}_{j,j_e}$ are sufficiently long (e.g., each has length at least ξ), and (ii) their timestamp intervals do not overlap.

PROBLEM 1 (TRAJECTORY MOTIF DISCOVERY PROBLEM). *Given a trajectory $\mathcal{S}$ and a minimum motif length ξ, return the pair of subtrajectories $\mathcal{S}_{i,i_e}$ and $\mathcal{S}_{j,j_e}$ with the smallest DFD distance $d_F(i,i_e,j,j_e)$ among all pairs of non-overlapping subtrajectories (that is, $i < i_e < j < j_e$) with length at least ξ (that is, $i_e > i+\xi, j_e > j+\xi$).*

As mentioned previously, a variant of Problem 1 is to discover a motif between different trajectories. I.e., considering two trajectories $\mathcal{S}$ and $\mathcal{T}$, to return the pair of subtrajectories $\mathcal{S}_{i,i_e}$ and $\mathcal{T}_{j,j_e}$ whose DFD is the smallest among all possible pairs of their subtrajectories.

With Problem 1 in mind, a straightforward solution is to enumerate all pairs of subtrajectories $(\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e})$ and then compute the DFD value for each pair. Its time complexity is $O(n^6)$, as there are $O(n^4)$ pairs of subtrajectories and each call to DFD takes $O(\ell^2) = O(n^2)$ time. Even if we implement each call to DFD by [1], the time complexity is still $O(n^6 \cdot \frac{\log\log n}{\log n})$.

We observe that, for all subtrajectory pairs $(\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e})$ with the same start point (i,j), their DFD computation can be shared via dynamic programming. By incorporating this idea into the above solution, we obtain BruteDP (Algorithm 1) – a brute force algorithm that uses dynamic programming. A further optimization is to eliminate redundant calls of the ground distance function $d_G(\cdot,\cdot)$. We propose to precompute all pairs of ground distances, and store them in matrix $d_G[\cdot][\cdot]$ for quick access.

Algorithm 1 BruteDP (Trajectory $\mathcal{S}$, minimum length ξ)

```
   Input: trajectory S, length n, minimum motif length ξ
   Output: subtrajectory pair bpair = (S_{i,i_e}, S_{j,j_e})
1: bsf ← +∞; bpair ← ∅
2: for i ← 0 to n − 2ξ + 1 do
3:     for j ← i + ξ to n − ξ + 1 do
4:         d_F[i][j] ← d_G(i, j)                          ▷ initialization
5:         for t ← i + 1 to n do
6:             d_F[i][t] ← max(d_G(i, t), d_F[i][t-1])
7:             d_F[t][j] ← max(d_G(t, j), d_F[t-1][j])
8:         for i_e ← i + 1 to j − 1 do                    ▷ share DFD computation
9:             for j_e ← j + 1 to n do
10:                tmp ← min(d_F[i_e-1][j_e-1], d_F[i_e][j_e-1], d_F[i_e-1][j_e])
11:                d_F[i_e][j_e] ← max(d_G(i_e, j_e), tmp)
12:                if i_e > i + ξ, j_e > j + ξ and d_F[i_e][j_e] < bsf then
13:                    bsf ← d_F[i_e][j_e]; bpair ← (S_{i,i_e}, S_{j,j_e})
14: return bpair
```

Algorithm 1 can be adapted to motif discovery between different trajectories easily, i.e., with $\mathcal{S}_{j,j_e}$ playing the role of a subtrajectory in the second input trajectory, and by incrementing i until $n-\xi+1$ (instead of $n-2\xi+1$) at Line 2, and j starting from 0 (instead of $i+\xi$) at Line 3 (because this variant considers separate trajectories, thus not imposing the constraint $i < i_e < j < j_e$).

Analysis: With all pairs of ground distances available in matrix d_G, the time complexity of Algorithm 1 is $O(n^4)$, which is attributed to the nested for-loops for variables i, j (at Lines 2-3) and variables i_e, j_e (at Lines 8-9). The space complexity of the algorithm is $O(n^2)$, as it employs two 2-dimensional matrices: (i) $d_F[\cdot][\cdot]$ for implementing dynamic programming, and (ii) $d_G[\cdot][\cdot]$ for holding all-pair ground distances.

Before we proceed to our advanced techniques, we summarize frequently used notation in Table 2.

4. BOUNDING-BASED SOLUTION

We first analyze the properties of DFD (in Section 4.1). Then we exploit these properties to devise novel lower bound functions for DFD (in Section 4.2). Our lower bounds can be computed in amortized $O(1)$ time, and guarantee no false negatives (in Section

Symbol	Meaning
$\mathcal{S}$	input trajectory
$\mathcal{S}[i]$	(φ_i, λ_i), the i^{th} point of $\mathcal{S}$
$\mathcal{S}_{i,i_e}$	the subtrajectory of $\mathcal{S}$ starting at $\mathcal{S}[i]$ and ending at $\mathcal{S}[i_e]$
n	the length of trajectory $\mathcal{S}$
ξ	the minimum motif length
$d_G(i,j)$	the ground distance between $\mathcal{S}[i]$ and $\mathcal{S}[j]$
$d_F(i,i_e,j,j_e)$	the DFD between subtrajectories $\mathcal{S}_{i,i_e}$ and $\mathcal{S}_{j,j_e}$

Table 2: Notation

4.3). Finally, we propose a bounding-based solution that applies our lower bound functions to prune unpromising pairs of trajectories and reduce the number of DFD computations (in Section 4.4).

4.1 Properties of DFD

4.1.1 Non-monotonicity

Typical sequence/string mining algorithms exploit the monotone property to develop efficient Apriori-style algorithms. An example of the monotone property would be: "given a string S, if q_α is a substring of q_β, then the frequency of q_α in S cannot be smaller than the frequency of q_β in S." It would be tempting to adapt such an idea to solve our problem efficiently. Unfortunately, the DFD metric does not satisfy the monotone property. Formally:

DEFINITION 2 (CONTAINMENT $\subseteq$). *$\mathcal{S}_{i,i_e}$ is said to contain $\mathcal{S}_{i',i'_e}$, denoted as $\mathcal{S}_{i',i'_e} \subseteq \mathcal{S}_{i,i_e}$, iff $i' \geq i$ and $i'_e \leq i_e$.*

LEMMA 1 (NON-MONOTONICITY). *Let $(\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e})$ be a subtrajectory pair of S. Let $\mathcal{S}_{i',i'_e}, \mathcal{S}_{j',j'_e}$ be subtrajectories that satisfy $\mathcal{S}_{i',i'_e} \subseteq \mathcal{S}_{i,i_e}, \mathcal{S}_{j',j'_e} \subseteq \mathcal{S}_{j,j_e}$. It holds that, $d_F(i,i_e,j,j_e)$ is neither monotone increasing nor monotone decreasing with respect to $d_F(i',i'_e,j',j'_e)$.*

We provide a counter-example to demonstrate the non-monotonicity as follows.

Example: We illustrate Lemma 1 using a trajectory $\mathcal{S}$ with length $n = 12$. Figure 5 shows the ground distance for each pair $(\mathcal{S}[i], S[j])$. Consider three subtrajectories $\mathcal{S}_{0,2} \subseteq \mathcal{S}_{0,3} \subseteq \mathcal{S}_{0,4}$ and their DFD distances from $\mathcal{S}_{6,9}$. Using Algorithm 1, we can compute these DFD values: $d_F(0,2,6,9) = 4$, $d_F(0,3,6,9) = 1$, $d_F(0,4,6,9) = 7$. When comparing $\mathcal{S}_{0,2}$ and $\mathcal{S}_{0,3}$, the DFD value (from $\mathcal{S}_{6,9}$) decreases from 4 to 1. However, when comparing $\mathcal{S}_{0,3}$ and $\mathcal{S}_{0,4}$, the DFD value (from $\mathcal{S}_{6,9}$) increases from 1 to 7. I.e., DFD does not satisfy the monotone property.

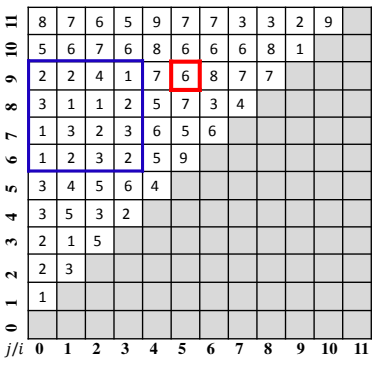


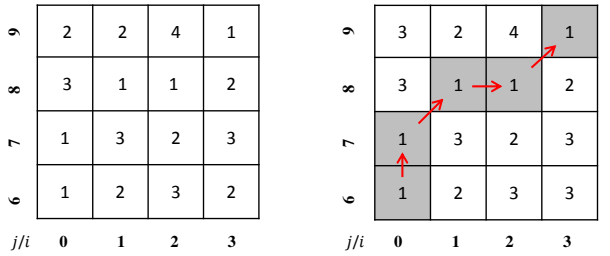
Figure 5: Example of d_G matrix

4.1.2 Crucial Observation

Non-monotonicity aside, we make a crucial observation which is quintessential to our approach. Specifically, the computation of DFD by recurrence is equivalent to a path finding problem in the d_G matrix.

OBSERVATION 1. *The DFD between $\mathcal{S}_{i,i_e}$ and $\mathcal{S}_{j,j_e}$ must be contributed by a path from (i,j) to (i_e,j_e) such that: (i) the path travels along non-decreasing positions, (ii) the worst-case ground distance along the path is minimized.*

We illustrate using two subtrajectories $\mathcal{S}_{0,3}$ and $\mathcal{S}_{6,9}$. Figure 6(a) shows the ground distance d_G for each pair of points from $\mathcal{S}_{0,3}$ and $\mathcal{S}_{6,9}$ (note that only the relevant part of the d_G matrix from Figure 5 is shown). We compute the d_F value for each pair of points, as illustrated in Figure 6(b). The DFD distance is $d_F(0,3,6,9) = 1$, which is contributed by the path of gray cells from $(0,6)$ to $(3,9)$, that minimizes the maximum ground distance among the cells it visits.



(a) Relevant part of d_G (b) d_F computation as a path in d_G

Figure 6: DFD computation for $\mathcal{S}_{0,3}$ and $\mathcal{S}_{6,9}$

4.2 Pattern-based Lower Bounds

Based on Observation 1, we devise novel lower bound functions for DFD by accessing/traversing the d_G matrix according to different patterns (e.g., a single cell, cells in a cross, cells in a band).

Specifically, assuming that matrix d_G is precomputed and that bsf is the DFD of the best subtrajectory pair encountered so far in the search process, we propose a set of lower bound functions that apply to candidate subtrajectory pairs, or entire groups of candidate pairs, such that if the bound is greater than bsf, the candidates are safe to prune, i.e., to disqualify without further consideration, because they are guaranteed not to be the motif.

4.2.1 Cell-based Lower Bound

We refer to a subtrajectory pair $(\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e})$ as candidate (i,i_e,j,j_e). We define a candidate subset $CS_{i,j}$ to represent all candidates with the same start positions i and j. This compact notation, using a pair (i,j), allows us to represent $O(n^2)$ candidates.

DEFINITION 3 (CANDIDATE SUBSET). *Given two start positions i and j, the candidate subset is defined as $CS_{i,j} = \{(i,i_e,j,j_e) : i_e > i \wedge j_e > j\}$.*

The following holds for any $CS_{i,j}$.

OBSERVATION 2. *For every $(i,i_e,j,j_e) \in CS_{i,j}$, the path leading to $d_F(i,i_e,j,j_e)$ must start from cell (i,j).*

For example, in Figure 6(a), for each candidate in $CS_{i,j}$, the path leading to DFD must start at cell $(0,6)$. We thus derive our first bound, which applies to any candidate in $CS_{i,j}$:

$$LB_{cell}(i,j) = d_G(i,j) \quad (1)$$

For every $(i, i_e, j, j_e) \in CS_{i,j}$, $LB_{cell}(i,j) \leq d_F(i, i_e, j, j_e)$.

Example: In Figure 5, for candidate subset $CS_{5,9}$ (i.e., for all candidate pairs that start at the red cell), we obtain $LB_{cell}(5,9) = d_G(5,9) = 6$. This is a lower bound for the DFD of any candidate pair in $CS_{5,9}$. E.g., for pair $(\mathcal{S}_{5,6}, \mathcal{S}_{9,11})$, the exact DFD is $d_F(5,6,9,11) = 7$.

4.2.2 Cross-based Lower Bound

If a candidate subset is not pruned using LB_{cell}, we attempt to prune it with tighter lower bounds.

OBSERVATION 3. *For every $(i, i_e, j, j_e) \in CS_{i,j}$, the path leading to $d_F(i, i_e, j, j_e)$ must pass through the $(i+1)$-th column and $(j+1)$-th row.*

Example: In Figure 5, consider candidate $(4, 6, 8, 10)$ in the candidate subset $CS_{4,8}$. For this candidate, any path from the start-cell (4,8) to the end-cell (6,10) must pass through the 5-th column and 9-th row; otherwise, the path cannot reach the end-cell (6,10). We thus define the following lower bounds.

$$LB_{row}(i,j) = \min_{i' \in [i, j-1]} \{d_G(i', j+1)\} \quad (2)$$

$$LB_{col}(i,j) = \min_{j' \in [j, n-1]} \{d_G(i+1, j')\} \quad (3)$$

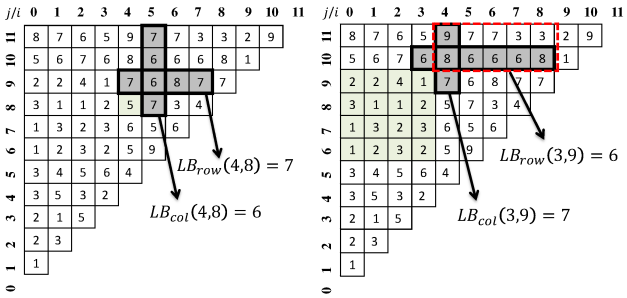
For every $(i, i_e, j, j_e) \in CS_{i,j}$, it holds that $LB_{row}(i,j) \leq d_F(i, i_e, j, j_e)$ and that $LB_{col}(i,j) \leq d_F(i, i_e, j, j_e)$. Thus, we combine the two into the cross-based lower bound below:

$$LB^{start}_{cross}(i,j) = \max\left(LB_{row}(i,j), LB_{col}(i,j)\right) \quad (4)$$

For every $(i, i_c, j, j_c) \in CS_{i,j}$, $LB^{start}_{cross}(i,j) \leq d_F(i, i_c, j, j_c)$.

Example: Consider cell (4,8) in Figure 7(a), and assume that $n = 12$. $LB^{start}_{cross}(4,8)$ is computed over the gray cells as follows:

$$LB^{start}_{cross}(4,8) = \max(LB_{row}(4,8), LB_{col}(4,8))$$
$$= \max\left(\min_{i' \in [4,7]} \{d_G(i', 9)\}, \min_{j' \in [8,11]} \{d_G(5, j')\}\right)$$
$$= \max(6,6) = 6$$



(a) $LB^{start}_{cross}(4,8)$ (b) $LB^{end}_{cross}(3,9)$

Figure 7: Examples of cross-based bounds

4.2.3 Band-based Lower Bound

Our problem definition considers only subtrajectories with length at least ξ. Based on that, we extend Observation 3 to:

OBSERVATION 4. *For every $(i, i_e, j, j_e) \in CS_{i,j}$ that satisfies the constraint $i_e > i + \xi$ and $j_e > j + \xi$, the path leading to $d_F(i, i_e, j, j_e)$ must pass through columns $i+1$ to $i+\xi$ and through rows $j+1$ to $j+\xi$.*

Hence, we define the following band-based lower bounds:

$$LB^{row}_{band}(i,j) = \max_{j' \in [j, j+\xi-1]} \{LB_{row}(i,j')\} \quad (5)$$

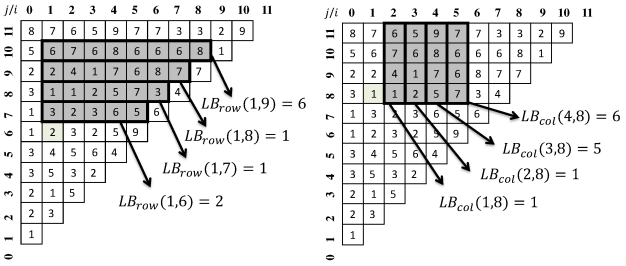
$$LB^{col}_{band}(i,j) = \max_{i' \in [i, i+\xi-1]} \{LB_{col}(i',j)\} \quad (6)$$

For every $(i, i_e, j, j_e) \in CS_{i,j}$ where $i_e > i + \xi$ and $j_e > j + \xi$ it holds that:

$$LB^{row}_{band}(i,j) \leq d_F(i, i_e, j, j_e) \quad (7)$$

$$\text{and} \quad LB^{col}_{band}(i,j) \leq d_F(i, i_e, j, j_e) \quad (8)$$

If $LB^{row}_{band}(i,j) \geq bsf$ or $LB^{col}_{band}(i,j) \geq bsf$ we can safely prune $CS_{i,j}$.



(a) $LB^{row}_{band}(1,6)$ (b) $LB^{col}_{band}(1,8)$

Figure 8: Example of band-based bound

Example: Consider candidate subset $CS_{1,6}$ in Figure 8(a). Suppose the minimum motif length is $\xi = 4$ and $n = 12$. By the definition of $LB_{row}(i,j)$, the minimum values in the 7-th, 8-th, 9-th and 10-th row are 2, 1, 1 and 6, respectively. Hence, $LB^{row}_{band}(1,6) = \max(2,1,1,6) = 6$. Similarly, consider candidate subset $CS_{1,8}$ in Figure 8(b). By the definition of $LB_{col}(i,j)$, the minimum value of the 2-nd, 3-rd, 4-th and 5-th column are 1, 1, 5 and 6, respectively, as shown in Figure 8(b). Hence, $LB^{col}_{band}(1,8) = \max(1,1,5,6) = 6$.

4.2.4 Pruning within Candidate Subset

The bounds presented so far prune entire candidate subsets. If a candidate subset $CS_{i,j}$ survives these bounds, we need to consider candidate pairs inside of it. To avoid considering all candidate pairs in $CS_{i,j}$, here we introduce a cross-based bound that prunes candidate pairs within $CS_{i,j}$.

As introduced in Section 3, for all candidate pairs (i.e., $\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e}$) in candidate set $CS_{i,j}$, their DFD computation can be shared via dynamic programming. Assume that at some point, the dynamic programming reaches end-cell (i_e, j_e), where $i_e - i > \xi$, $j_e - j > \xi$ and $bsf = d_F(i, i_e, j, j_e)$. We define the following cross-based lower bound for the end-cell:

$$LB^{end}_{cross}(i_e, j_e) = \max\left(LB_{row}(i_e, j_e), LB_{col}(i_e, j_e)\right) \quad (9)$$

If (i, i_c, j, j_c) is a candidate in $CS_{i,j}$ where $i_c > i_e$ and $j_c > j_e$, it holds that $LB^{end}_{cross}(i_e, j_e) \leq d_F(i, i_c, j, j_c)$. Hence, if $LB^{end}_{cross}(i_e, j_e) \geq bsf$, we can safely avoid expanding cell (i_e, j_e), i.e., eliminate paths within $CS_{i,j}$ that pass via cell (i_e, j_e).

Example: In Figure 7(b), suppose $\xi = 2$, $i = 0$, $j = 6$, $i_e = 3$

and $j_e = 9$. $LB^{end}_{cross}(i_e, j_e)$ is computed over the gray cells:

$$LB^{end}_{cross}(3,9) = \max(LB_{row}(3,9), LB_{col}(3,9))$$
$$= \max\Big(\min_{i'_e \in [3,8]}\{d_G(i'_e, 10)\}, \min_{j'_e \in [9,11]}\{d_G(4, j'_e)\}\Big)$$
$$= \max(6,7) = 7$$

If $LB^{end}_{cross}(3,9) \geq bsf$, we prune the candidates (i.e., subtrajectory pairs) in $CS_{0,6}$ whose end-cells fall in the red dotted box.

4.3 Relaxed Lower Bounds

If we follow the aforementioned equations directly, a cross-based bound takes $O(n)$ time to compute and a band-based bound takes $O(\xi n)$ time. Although both of them are more efficient than raw DFD computation (i.e., $O(n^2)$), in this section, we drop their amortized time complexity to $O(1)$ by relaxing them slightly. These relaxed bounds incur no false negatives, i.e., they are guaranteed not to miss the motif. Due to the space limit, we illustrate our relaxation approach for band-based bounds only. The relaxation of cross-based bounds follows the same lines.

The key idea is to employ one parameter per bound, and keep them in matrices for rapid access. First, we compute the minimum value for each column i and each row j:

$$C_{min}[i] = \min_{j' \in [0, j-1]}(d_G(i+1, j')) \quad (10)$$

$$R_{min}[j] = \min_{i' \in [i, n-1]}(d_G(i', j+1)) \quad (11)$$

This step takes $O(2 \cdot n \cdot n) = O(n^2)$ time.

We define the relaxed version of cross-based bounds as:

$$rLB^{start}_{cross}(i,j) = \max\{C_{min}[i], R_{min}[j]\} \quad (12)$$

$$rLB^{end}_{cross}(i_e, j_e) = \max\{C_{min}[i_e], R_{min}[j_e]\} \quad (13)$$

In turn, the relaxed band-based bounds are defined as:

$$rLB^{row}_{band}(j) = \max_{j' \in [j, j+\xi-1]}\{R_{min}[j']\} \quad (14)$$

$$rLB^{col}_{band}(i) = \max_{i' \in [i, i+\xi-1]}\{C_{min}[i']\} \quad (15)$$

We compute the relaxed version of cross-based bounds by computing $C_{min}[i]$ and $R_{min}[j]$ for each column i and each row j. This step takes $O(n)$ time per column/row. Similarly, we compute relaxed band-based bounds for each column i and each row j. This step takes $O(\xi n)$ time per row/column. Thus, the total computation time of cross-based and band-based lower bounds is $O(n \cdot n) = O(n^2)$ and $O(\xi n \cdot n) = O(n^2)$, respectively. By amortizing the computation time over all candidate subsets $CS_{i,j}$ (i.e., $O(n^2)$ of them), the computation time per $CS_{i,j}$ for each relaxed bound is only $O(n^2/n^2) = O(1)$.

The following lemma proves the correctness of the relaxed band-based bounds. The proof for the relaxed cross-based bounds follows the same lines and is omitted for brevity.

LEMMA 2. *It holds that:*

$$rLB^{row}_{band}(j) \leq LB^{row}_{band}(i,j) \quad \textit{and} \quad rLB^{col}_{band}(i) \leq LB^{col}_{band}(i,j)$$

PROOF.

$$\min_{i' \in [0, j-1]}(d_G(i', j+1)) \leq \min_{i' \in [i, j-1]}(d_G(i', j+1))$$
$$\Rightarrow R_{min}[j] \leq LB_{row}(i,j)$$
$$\Rightarrow \max_{j' \in [j, j+\xi-1]}\{R_{min}[j']\} \leq \max_{j' \in [j, j+\xi-1]}\{LB_{row}(i,j')\}$$
$$\Rightarrow rLB^{row}_{band}(j) = LB^{row}_{band}(i,j)$$

Similarly, $rLB^{col}_{band}(i) \leq LB^{col}_{band}(i,j)$. □

In the experiments, we compare the effectiveness of the original bounds with the relaxed ones. We summarize the time requirements of all lower bounds in Table 3.

Lower bound	Time	Relaxed bound	Time
$LB_{cell}(i,j)$	$O(1)$		
$LB^{start}_{cross}(i,j)$	$O(n)$	$rLB^{start}_{cross}(i,j)$	$O(1)$
$LB^{end}_{cross}(i_e,j_e)$	$O(n)$	$rLB^{end}_{cross}(i_e,j_e)$	$O(1)$
$LB^{row}_{band}(i,j)$	$O(\xi n)$	$rLB^{row}_{band}(j)$	$O(1)$
$LB^{col}_{band}(i,j)$	$O(\xi n)$	$rLB^{col}_{band}(i)$	$O(1)$

Table 3: Summary of lower bounds

4.4 Optimized Solution

Combining all bounds: Given a candidate subset $CS_{i,j}$, we compute a tighter lower bound for $CS_{i,j}$, denoted by $CS_{i,j}.LB$, using:

$$\max\{LB_{cell}(i,j), rLB^{start}_{cross}(i,j), rLB^{row}_{band}(j), rLB^{col}_{band}(i)\}.$$

This lower bound takes $O(1)$ time because each term can be obtained in $O(1)$ time, as shown in Table 3.

Prioritizing search order: To support effective pruning of $CS_{i,j}$ by lower bounds, it is desirable to obtain a small bsf (i.e., a good temporary motif) as early as possible. Intuitively, a candidate subset with small $CS_{i,j}.LB$ tends to contain a candidate with small DFD value. Thus, we propose to process $CS_{i,j}$ in ascending order of $CS_{i,j}.LB$.

Putting it all together: Algorithm 2 presents the pseudocode for *bounding-based trajectory motif* (BTM), which incorporates all above ideas to solve the trajectory motif discovery problem.

Algorithm 2 BTM (Trajectory $\mathcal{S}$, minLength ξ)

```
   Input: trajectory S, length n, minimum motif length ξ
   Output: subtrajectory pair bpair = (S_{i,ie}, S_{j,je})
1: bsf ← +∞; bpair ← ∅; j_end ← n
2: Compute { LB_cell, rLB^start_cross, rLB^end_cross, rLB^row_band, rLB^col_band }
3: Construct a list A with one element a per candidate subset
4: Sort A in ascending order of a.LB
5: for each a in A with bsf > a.LB do
6:     for i_e ← a.i + 1 to a.j do
7:         for j_e ← a.j + 1 to j_end do
8:             tmp ← min(d_F[i_e-1][j_e-1], d_F[i_e][j_e-1], d_F[i_e-1][j_e])
9:             d_F[i_e][j_e] ← max(d_G(i_e, j_e), tmp)
10:            if i_e > a.i + ξ, j_e > a.j + ξ and d_F[i_e][j_e] < bsf then
11:                bsf ← d_F[i_e][j_e]; bpair ← (S_{i,i_e}, S_{j,j_e})
12:        if bsf ≤ rLB^end_cross(bpair.i_e, bpair.j_e) then
13:            j_end ← bpair.j_e      ▷ Pruning by LB^end_cross(i_e, j_e) from
   Equation 9
14: return bpair
```

At Line 2, we first compute all lower bounds (and store them in matrices). Then, we insert each candidate subset $CS_{i,j}$ with its bound $CS_{i,j}.LB$ into a list (at Line 3), and sort that list (at Line 4). Next, we process the elements of the list in the sorted order. For each candidate subset, we examine its candidates via nested loops (at Lines 6-7), and compute the DFD of each candidate (at Line 8-9). Finally, we update bsf and the temporary motif pair (at Lines 10-11). Note that Lines 12-13 implement pruning by $LB^{end}_{cross}(i_e, j_e)$, as defined in Equation 9; this essentially performs pruning within the candidate subset currently considered, by disqualifying some of the candidate pairs it contains.

The lower bounds presented in this section are also applicable to motif discovery between different trajectories. Hence, similarly

to Algorithm 1, Algorithm 2 is readily applicable to that problem variant too.

Analysis: The time complexity of Algorithm 2 is $O(n^4)$ in the worst case, which is attributed to the nested for-loops for variables $\mathbf{a}$ [that is, $O(n^2)$ iterations], i_e [that is, $O(n)$ iterations] and j_e [that is, $O(n)$ iterations] at Lines 5-7. The space complexity of Algorithm 2 is $O(n^2)$.

Algorithm 2 follows the best-first search paradigm with several effective lower bounds. As we will show in the experimental evaluation, it outperforms Algorithm 1 by two orders of magnitude.

5. GROUPING-BASED SOLUTION

In this section, we enhance the scalability of our techniques for long trajectories. Inspired by trajectory indexing methods [4, 9], we organize trajectory points into groups, then attempt pruning unpromising pairs of groups, before applying our solution from Section 4. To enable pruning, we design novel bounding functions for DFD on groups.

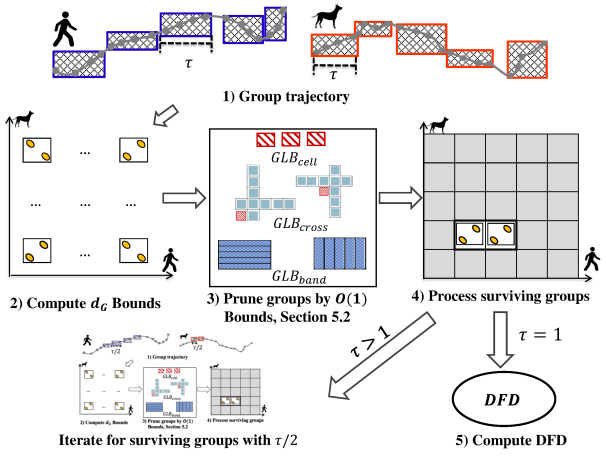


Figure 9: Grouping-based computation framework

We outline our grouping-based computation framework in Figure 9. First, we divide a trajectory into groups of τ samples (where τ is a tunable parameter), and compute a ground distance bound for each group pair (Steps 1 and 2, in Section 5.1). Next, we apply $O(1)$-time lower bounds (Step 3, in Section 5.2) to prune group pairs, before using tighter bounds for pruning (Step 4, in Section 5.3). For the surviving group pairs, we repeat the above steps by halving the group size, until τ reaches 1. Finally, we compute the exact DFD of candidates in the surviving groups (Step 5).

By combining the advantages of all techniques in Section 4 and in the current one, our grouping based computation framework outperforms the baseline solution by over 3 orders of magnitude. Importantly, all our techniques conduct only safe pruning, meaning that they produce exact answers (motifs).

5.1 Grouping Trajectory Points

We employ a group size parameter τ in order to partition a long trajectory into small groups. We proceed to define a group and the ground distances between groups.

DEFINITION 4 (τ-GROUPING). *Given the group size τ, we define the u-th group as the interval $g_u = [u\tau, (u+1)\tau - 1]$.*

For two groups g_u and g_v, we define the minimum and the maximum ground distance between them as:

$$d_G^{min}(g_u, g_v) = \min_{i \in g_u, j \in g_v} d_G(i,j) \quad (16)$$

$$d_G^{max}(g_u, g_v) = \max_{i \in g_u, j \in g_v} d_G(i,j) \quad (17)$$

By Definition 4, the ground distances between two groups satisfy the following property:

COROLLARY 1. *For every $i \in g_u$, $j \in g_v$, it holds that:*

$$d_G^{min}(g_u, g_v) \leq d_G(i,j) \leq d_G^{max}(g_u, g_v)$$

We utilize this property to devise lower bound functions in Sections 5.2, 5.3.

Example: Consider a trajectory $\mathcal{S}$ with $n = 12$ points. Given $\tau = 2$, we obtain six groups: $g_0, g_1, g_2, g_3, g_4, g_5$, as illustrated in Figure 10(a). For example, for groups $g_2 = [4,5]$ and $g_5 = [10,11]$, we compute the minimum ground distance as $d_G^{min}(g_2, g_5) = \min(d_G(4,10), d_G(4,11), d_G(5,10), d_G(5,11)) = 6$, and the maximum ground distance as $d_G^{max}(g_2, g_5) = \max(8,9,6,7) = 9$. We show the minimum and maximum ground distances for group pair g_2 and g_5 in Figure 10(b).

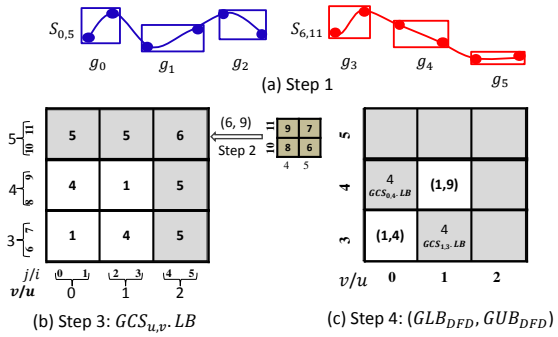


Figure 10: Example of 2-grouped trajectory

5.2 Pattern-based Bounds for Groups

To enable pruning on unpromising pairs of groups, we adapt our proposed lower bounds in Section 4 to groups. We denote the corresponding lower bounds with prefix $\mathcal{G}$, i.e., $\mathcal{GLB}_{cell}(u,v)$, $\mathcal{GLB}^{start}_{cross}(u,v)$, $\mathcal{GLB}^{end}_{cross}(u_e,v_e)$, $\mathcal{GLB}^{row}_{band}(u,v)$, and $\mathcal{GLB}^{col}_{band}(u,v)$. Later, we discuss their $O(1)$-time implementation.

Cell-based lower bound: We first define the cell-based lower bound for groups, denoted by $\mathcal{GLB}_{cell}$, as follows:

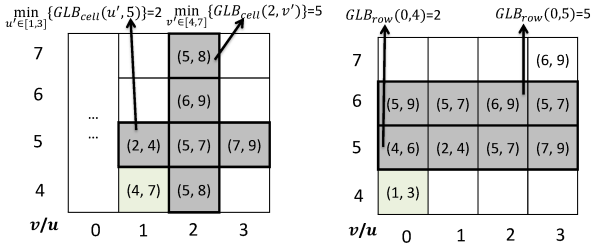
$$\mathcal{GLB}_{cell}(u,v) = d_G^{min}(g_u, g_v) \quad (18)$$

In Figure 10(b), $\mathcal{GLB}_{cell}(2,5) = d_G^{min}(2,5) = 6$. For any $i \in u$ and $j \in v$, it holds that $\mathcal{GLB}_{cell}(u,v) \leq d_F(i, i_e, j, j_e)$.

Cross-based lower bounds: Next, we show that the cross-based lower bounds for groups can be expressed in terms of $\mathcal{GLB}_{cell}(u,v)$. We demonstrate using an example, rather than presenting ugly definitions and lemmas.

We denote the row and column based lower bounds for groups as $\mathcal{GLB}_{row}(u,v)$ and $\mathcal{GLB}_{col}(u,v)$, respectively. In Figure 11(a), assuming $n = 16$ and $\tau = 2$, we obtain $\mathcal{GLB}_{row}(1,4) = \min_{u' \in [1,3]}(\mathcal{GLB}_{cell}(u',5)) = \min(2,5,7) = 2$. Similarly, $\mathcal{GLB}_{col}(1,4) = \min(5,5,6,5) = 5$. The cross-based lower bound for start-cell (1,4) is $\mathcal{GLB}^{start}_{cross}(1,4) =$

$\max(\mathcal{GLB}_{row}(1,4), \mathcal{GLB}_{col}(1,4)) = \max(2,5) = 5$. $\mathcal{GLB}^{end}_{cross}(u_e, v_e)$ is defined similarly to $\mathcal{GLB}^{start}_{cross}(u,v)$.



(a) $\mathcal{GLB}^{start}_{cross}(1,4)$ (b) $\mathcal{GLB}^{row}_{band}(0,4)$

Figure 11: Grouping based lower bounds

Band-based lower bounds: We also present band-based lower bounds for groups using an example. In Figure 11(b), we illustrate $\mathcal{GLB}^{row}_{band}(0,4)$. We compute it as $\mathcal{GLB}^{row}_{band}(0,4) = \max_{v' \in [4,5]}(\mathcal{GLB}_{row}(0,v')) = \max(2,5) = 5$. $\mathcal{GLB}^{col}_{band}(u,v)$ is defined similarly to $\mathcal{GLB}^{row}_{band}(u,v)$.

Relaxed lower bounds for groups: The concept of relaxed lower bounds, introduced in Section 4.3, can be adapted directly to the above pattern-based bounds for groups. This allows us to obtain relaxed lower bounds for groups in $O(1)$ time.

5.3 Bounding by DFD Computation

By exploiting the recurrence of DFD, we devise a tighter lower bound and a tighter upper bound for pairs of groups. While the lower bound is used to prune unpromising pairs of groups, the upper bound can be used to tighten bsf and thus improve the effectiveness of pruning.

Below we define a subtrajectory group, together with group-based DFD bounds.

DEFINITION 5 (GROUP-BASED DFD). *Let subtrajectory group $\mathcal{G}_{t,t_e}$ correspond to the interval $[t\tau, (t_e+1)\tau - 1]$, i.e., it covers group t to group t_e.*

Given two subtrajectory groups $\mathcal{G}_{u,u_e}$ and $\mathcal{G}_{v,v_e}$, we define the group-based DFD bounds $d_{Fmin}(u,u_e,v,v_e)$ and $d_{Fmax}(u,u_e,v,v_e)$ as:

$$d_{Fmin}(u,u_e,v,v_e) = \max \begin{cases} d^{min}_G(g_{u_e}, g_{v_e}) \\ \min \begin{cases} d_{Fmin}(u, u_e-1, v, v_e) \\ d_{Fmin}(u, u_e-1, v, v_e-1) \\ d_{Fmin}(u, u_e, v, v_e-1) \end{cases} \end{cases}$$

$$d_{Fmax}(u,u_e,v,v_e) = \max \begin{cases} d^{max}_G(g_{u_e}, g_{v_e}) \\ \min \begin{cases} d_{Fmax}(u, u_e-1, v, v_e) \\ d_{Fmax}(u, u_e-1, v, v_e-1) \\ d_{Fmax}(u, u_e, v, v_e-1) \end{cases} \end{cases}$$

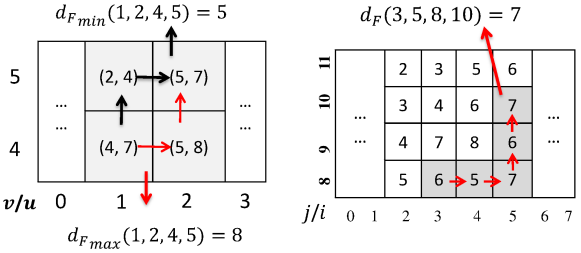
The following lemma proves the bounding property of $d_{Fmin}(u,u_e,v,v_e)$ and $d_{Fmax}(u,u_e,v,v_e)$.

LEMMA 3. *Let $\mathcal{G}_{u,u_e}$ and $\mathcal{G}_{v,v_e}$ be two subtrajectory groups. If a pair of subtrajectories $\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e}$ satisfies $i \in g_u, j \in g_v, i_e \in g_{u_e}$ and $j_e \in g_{v_e}$, it holds that:*

$$d_{Fmin}(u,u_e,v,v_e) \le d_F(i,i_e,j,j_e) \le d_{Fmax}(u,u_e,v,v_e)$$

PROOF. $\forall u' \in [u,u_e], \forall v' \in [v,v_e]$ and $i' \in g_{u'}, j' \in g_{v'}$, according to Corollary 1, we have $d^{min}_G(g_{u'}, g_{v'}) \le d_G(i',j')$. By Observation 1, $d_{Fmin}(u,u_e,v,v_e)$ is attributed to a path among $d^{min}_G(g_{u'}, g_{v'})$ values, and $d_F(i,i_e,j,j_e)$ to a path among $d_G(i',j')$ values. Hence, for $i \in g_u, j \in g_v, i_e \in g_{u_e}, j_e \in g_{v_e}$, it holds that $d_{Fmin}(u,u_e,v,v_e) \le d_F(i,i_e,j,j_e)$. $d_{Fmax}(u,u_e,v,v_e) \ge d_F(i,i_e,j,j_e)$ is proven similarly. □

Example: Consider two subtrajectory groups $\mathcal{G}_{1,2}$ and $\mathcal{G}_{4,5}$ in Figure 12(a) and assume that $n = 12$. Their DFD bounds are $d_{Fmin}(1,2,4,5) = 5$ and $d_{Fmax}(1,2,4,5) = 8$, respectively. In Figure 12(b), the pair of subtrajectories $\mathcal{S}_{3,5}, \mathcal{S}_{8,10}$ has $d_F(3,5,8,10) = 7$. In accordance with Lemma 3, this distance falls indeed into range $[5,8]$.



(a) DFD bounds on groups (b) DFD on original trajectory

Figure 12: Illustration of DFD bounds

Recall that our problem definition enforces a minimum motif length ξ. To comply with it, we define the following lower and upper bounds between two groups g_u and g_v:

$$\mathcal{GLB}_{DFD}(u,v) = \min_{u_e,v_e}\{d_{Fmin}(u,u_e,v,v_e) : u_e - u > \frac{\xi}{\tau} \wedge v_e - v > \frac{\xi}{\tau}\} \quad (19)$$

$$\mathcal{GUB}_{DFD}(u,v) = \min_{u_e,v_e}\{d_{Fmax}(u,u_e,v,v_e) : u_e - 1 - u > \frac{\xi}{\tau} \wedge v_e - 1 - v > \frac{\xi}{\tau}\} \quad (20)$$

The following lemma shows their correctness. It is derived by applying the $\min_{u_e,v_e}$ function to both sides of Lemma 3.

LEMMA 4. *$\forall i \in g_u, \forall i \in g_v$, and $i_e > i + \xi, j_e > j + \xi$ it holds that:*

$$\mathcal{GLB}_{DFD}(u,v) \le d_F(i,i_e,j,j_e) \le \mathcal{GUB}_{DFD}(u,v)$$

$\mathcal{GUB}_{DFD}(u,v)$ allows us to tighten bsf, which in turn boosts the effectiveness of pruning. Both $\mathcal{GLB}_{DFD}(u,v)$ and $\mathcal{GUB}_{DFD}(u,v)$ can be computed in $O((\frac{n}{\tau})^2)$. We can reduce their computation cost by early termination. Specifically, if at some point during the computation of $\mathcal{GLB}_{DFD}(u,v)$, it holds that $\mathcal{GLB}^{end}_{cross}(u_e,v_e) \ge \mathcal{GLB}_{DFD}(u,v)$ with $u_e - u > \frac{\xi}{\tau} \wedge v_e - v > \frac{\xi}{\tau}$, we may safely terminate the computation because $\forall u_{e'} > u_e$ and $\forall v_{e'} > v_e$ it must be that $d_{Fmin}(u,u_{e'},v,v_{e'}) > d_{Fmin}(u,u_e,v,v_e)$ (i.e., it cannot further tighten the bound). Similarly, early termination is possible in the calculation of $\mathcal{GUB}_{DFD}(u,v)$ too.

5.4 GTM Algorithm

Algorithm 3 presents the pseudocode for *grouping-based trajectory motif* (GTM), which implements the computation framework depicted in Figure 9. We first construct groups at Line 3, then we compute the pattern-based lower bounds of group pairs at Lines 4-5. Next, we insert each grouping based candidate subset $\mathcal{GCS}_{u,v}$ with its bound $\mathcal{GCS}_{u,v}.LB = \max(\mathcal{GLB}_{cell}(u,v),$

$r\mathcal{GLB}^{start}_{cross}(u,v), r\mathcal{GLB}^{row}_{band}(u,v), r\mathcal{GLB}^{col}_{band}(u,v))$ into a list. Then, we process the list in ascending order of $\mathcal{GCS}_{u,v}.LB$, and apply DFD bounds for pruning (Lines 10-11) or for tightening the bsf (Lines 12-13). After that, we halve the group size and repeat the above procedure on the set of surviving groups $\mathcal{S}_{survive}$ until the group size drops to 1. When this happens (i.e., $\tau = 1$), each element in $\mathcal{S}_{survive}$ is a candidate subset $CS_{i,j}$. We invoke Algorithm 2 on $\mathcal{S}_{survive}$ to obtain the final result.

Algorithm 3 GTM (Trajectory $\mathcal{S}$, minLength ξ, group size τ)

Input: trajectory $\mathcal{S}$, length n, minimum motif length ξ, group size τ
Output: subtrajectory pair $bpair = (\mathcal{S}_{i,i_e}, \mathcal{S}_{j,j_e})$

1: $bsf \leftarrow +\infty$; $bpair \leftarrow \emptyset$
2: **while** $\tau > 1$ **do**
3: Group trajectory $\mathcal{S}$ to $\mathcal{G}$ ▷ Section 5.1
4: Compute $\mathcal{GLB}_{cell}, r\mathcal{GLB}^{start}_{cross}, r\mathcal{GLB}^{end}_{cross}$
5: and $r\mathcal{GLB}^{row}_{band}, r\mathcal{GLB}^{col}_{band}$ ▷ Section 5.2
6: Construct a list $\mathcal{GA}$ of candidate subsets
7: Sort $\mathcal{GA}$ in ascending order of $\mathcal{GA}.LB$
8: $\mathcal{S}_{survive} \leftarrow \emptyset$ ▷ set of surviving groups
9: **for** each $\mathcal{GA}$ in $\mathcal{GA}$ with $bsf > \mathcal{GA}.LB$ **do** ▷ Section 5.3
10: **if** $bsf > \mathcal{GLB}_{DFD}(\mathcal{GA}.u, \mathcal{GA}.v)$ **then**
11: $\mathcal{S}_{survive} \leftarrow \mathcal{S}_{survive} \cup \mathcal{GA}.u \cup \mathcal{GA}.v$
12: **if** $bsf > \mathcal{GUB}_{DFD}(\mathcal{GA}.u, \mathcal{GA}.v)$ **then**
13: $bsf \leftarrow \mathcal{GUB}_{DFD}(\mathcal{GA}.u, \mathcal{GA}.v)$
14: $\tau \leftarrow \tau/2$, $\mathcal{S} \leftarrow \mathcal{S}_{survive}$
15: Invoke Lines 5-13 in Alg. 2 on $\mathcal{S}_{survive}$ to compute the result *bpair*

Example: We demonstrate the grouping-based computation framework in Figure 10. Assume that the minimum trajectory motif length is $\xi = 2$ with $bsf = 5$. We first assign these subtrajectories into groups (with $\tau = 2$), as illustrated in Figure 10(a). Consider two subtrajectories $\mathcal{S}_{0,5}$ and $\mathcal{S}_{6,11}$. We compute $O(1)$-time pattern-based bounds to prune group pairs; the pruned pairs are shown in gray in Figure 10(b). Then, we compute $\mathcal{GLB}_{DFD}, \mathcal{GUB}_{DFD}$ bounds for surviving pairs, as illustrated in Figure 10(c). The upper bounds allow us to tighten bsf (to 4), whereas the lower bounds are used to prune pairs (i.e., the gray region in Figure 10(c)). Finally, we process the two surviving cells with Algorithm 2.

The group lower bounds developed in this section are directly applicable to motif discovery between different trajectories, and the adaptation of Algorithm 3 to that variant is straightforward.

Analysis: The computation cost of the while-loop (Lines 2–14) is $O(\sum_{i=1}^{log(\tau)} (\frac{c_i}{\tau})^4)$, where $c_1 = n$ and c_i is the number of surviving groups in iteration i. Line 16 takes $O(c\tau^2 n^2)$ time, where c is the number of surviving groups after the while-loop. In summary, the time complexity of Algorithm 3 is $O(\sum_{i=1}^{log(\tau)} (\frac{c_i}{\tau})^4 + c\tau^2 n^2)$. In the worst case, Algorithm 3 degenerates to Algorithm 2, with time complexity $O(n^4)$.

The space complexity of the algorithm is $O(n^2)$ as it employs two 2-dimensional matrices for precomputed ground distances (i.e., $d_G[\cdot][\cdot]$) and DFD values (i.e., $d_F[\cdot][\cdot]$). In addition, it takes $O((\frac{n}{\tau})^2)$ space for precomputed group based lower bounds in $\mathcal{GA}$ at Line 6.

5.5 Space-efficient GTM: GTM*

We present a space-efficient variant of GTM, called GTM*. It incorporates three ideas: (i) during DFD computation, we compute ground distances on-the-fly, (ii) implement DFD computation with $O(n)$ space, and (iii) execute the while-loop only once for a given τ. Idea (i) eliminates the need for precomputed ground distances (i.e., $d_G[\cdot][\cdot]$). Idea (ii) is feasible because, in Lines 8-9 of Algorithm 2, we examine at most two rows of $d_F[\cdot][\cdot]$ at the same time. Idea (iii) requires only $O((\frac{n}{\tau})^2)$ space. Thus, the space complexity of GTM* is $O(\max\{(\frac{n}{\tau})^2, n\})$.

The time complexity of GTM* is $O((\frac{n}{\tau})^4 + c'\tau^2 n^2)$, where c' is the number of group pairs that survive pruning by Idea (i). Since GTM* executes the while-loop only once for a given τ (Idea iii), the value of c' in GTM* is expected to be larger than c in GTM.

6. EMPIRICAL EVALUATION

In this section, we evaluate the performance of our solutions on real data. Section 6.1 introduces the experimental setting. Section 6.2 studies the effectiveness of our pruning techniques (e.g., lower bounds and grouping). Section 6.3 compares the performance of different methods with respect to various parameters.

6.1 Experimental Setup

We used three real trajectory datasets from moving people, vehicles and animals. We note that these datasets have different characteristics (such as sampling frequency and data distribution) thus helping us verify the generality of our findings. The details of each dataset are as follows.

GeoLife[1]**:** This GPS trajectory dataset was collected in the GeoLife project by Microsoft. The trajectories were recorded by different GPS loggers and GPS-phones, and therefore they have different sampling rates. Each trajectory is a sequence of time-stamped points, each with a latitude, a longitude and an altitude. This dataset contains 17,621 trajectories with a total distance of 1.2 million kilometers.

Truck[2]**:** This dataset contains 276 trajectories of 50 trucks moving in Athens metropolitan area in Greece. The trucks were carrying concrete to several construction sites for 33 days.

Wild-Baboon[3]**:** This dataset was collected from wild olive baboons at Mpala Research Centre in Kenya [23]. It contains 25 trajectories of baboons with a custom-designed GPS collar that recorded a location every second from 1-st August to 14-th August, 2012.

In our experiments, we report the average measurements over 10 different trajectories of the same length. The response times reported include the precomputation time of distances and lower bounds. For each dataset, we concatenate raw trajectories in order to build longer trajectories. By default, we fix the motif length threshold ξ to 100, and the trajectory length n to 5000.

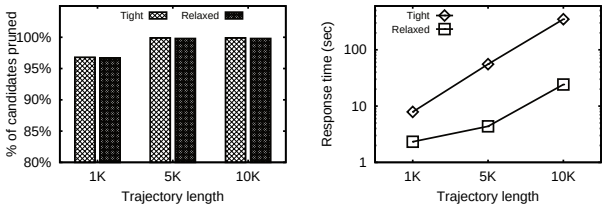
We used C++ for the implementation and conducted all experiments (with single thread) on a machine with an Intel Core i7- 4770 3.40GHz processor. We compare the following methods:

- the baseline solution BruteDP (cf. Algorithm 1)
- the bounding-based solution BTM (cf. Algorithm 2)
- the grouping-based solution GTM (cf. Algorithm 3)
- the space-efficient solution GTM* (cf. Section 5.5)

6.2 Pruning Effectiveness

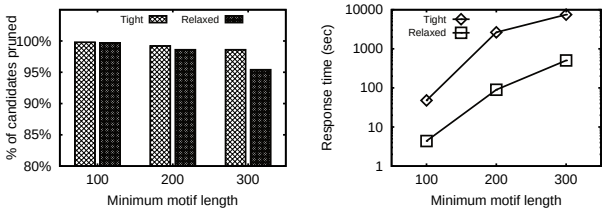
We first assess the effectiveness of our pruning techniques, particularly of our lower bounds and grouping. For the purposes of this subsection, we present results only on the GeoLife dataset. Results on Truck and Wild-Baboon are similar and are omitted in the interest of space.

[1] http://research.microsoft.com/en-us/projects/geolife/default.aspx
[2] http://chorochronos.datastories.org/
[3] https://www.datarepository.movebank.org/handle/10255/move.405



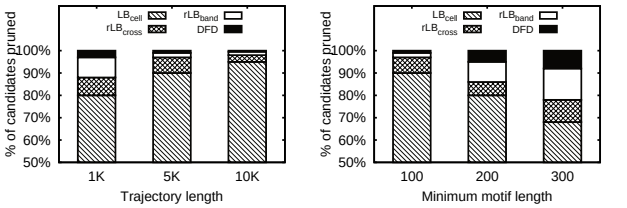
(a) Pruning ratio (b) Response time

Figure 13: BTM, effect of trajectory length n



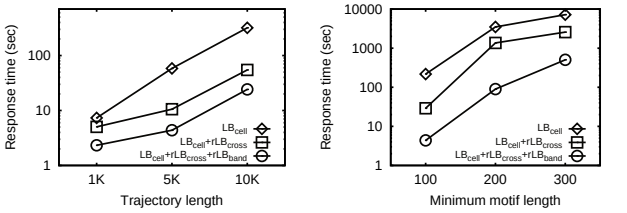
(a) Pruning ratio (b) Response time

Figure 14: BTM, effect of minimum motif length ξ



(a) Effect of trajectory length n (b) Effect of minimum motif length ξ

Figure 15: BTM, pruning ratio breakdown



(a) Effect of trajectory length n (b) Effect of minimum motif length ξ

Figure 16: BTM, response time

6.2.1 *Effectiveness of Relaxed Bounds*

We first compare two variants of BTM that use: (i) only the *tight lower bounds* from Section 4.2, and (ii) only the *relaxed lower bounds* from Section 4.3.

In Figure 13, we compare the tight with the relaxed bounds by varying the trajectory length n, with ξ fixed to 100. The pruning percentage in Figure 13(a) corresponds to the ratio of candidate pairs successfully pruned to the total number of candidate pairs. Note that because the percentage is high, and in order to show enough detail, we truncated the y-axis of the plot to start from 80%. In Figure 13(b), we show the overall response time to compute the motif. We observe that the relaxed bounds are only slightly weaker in pruning power, but they are orders of magnitude faster computation-wise.

In Figure 14, we investigate the effectiveness and performance of tight and relaxed bounds as a function of the minimum motif length ξ, with n fixed to 5000. Again, although the tight bounds have slightly higher pruning ratio (in Figure 14(a)), the relaxed bounds render motif computation 10 times faster (in Figure 14(b)). Since the relaxed bounds perform much better, we adopt them in our framework (instead of the tight ones) and use them in the subsequent experiments.

6.2.2 *Effectiveness of Lower Bounds*

In the next experiment, we compare the pruning effectiveness of the different lower bound functions ($LB_{cell}, rLB_{cross}, rLB_{band}$) using BTM. Each bar in Figure 15 corresponds to the total number of candidate pairs, broken down into the fraction pruned by each of the 3 types of bounds, and the fraction of the surviving pairs that required exact DFD computation (labeled as DFD in the bar charts). In Figures 15(a),(b) we vary the trajectory length n and the minimum motif length ξ, respectively. The bars are truncated to start at ratio 50% to retain detail, because the percentage of LB_{cell} hugely dominates the rest.

Over 92% of the candidates can be collectively pruned by our lower bounds. An interesting observation is that the bounds complement each other. For instance, when ξ increases (in Figure 15(b)), although LB_{cell} deteriorates, rLB_{band} becomes stronger, thus eliminating many of the candidates that survived LB_{cell}. This renders our methodology robust to different problem settings.

Next, we compare three variants of BTM that use: (i) LB_{cell} only, (ii) LB_{cell}, rLB_{cross} only, and (iii) $LB_{cell}, rLB_{cross}, rLB_{band}$. We vary the trajectory length n and the minimum motif length ξ in Figures 16(a),(b), respectively. The results verify that the bounds complement each other gracefully, and that the performance gains achieved are not due to just one or some of them.

6.2.3 *Effect of Group Size τ*

In GTM (Algorithm 3), the initial group size τ influences the pruning effectiveness and the computation cost of the algorithm. Generally, when τ is small, group-based pruning has a high pruning power but it requires high computation cost. In contrast, when τ is large, group-based pruning becomes faster but it becomes less effective. Figure 17 plots the response time of GTM for different values of τ (x-axis) and trajectory length n (as indicated by the label of each line). We observe that the response time is not overly sensitive to τ. In the following experiments, we set $\tau = 32$ by default as it seems to work well in all cases.

6.3 Performance Evaluation

We compare the performance of our solutions (BTM, GTM, and GTM*) with the baseline (BruteDP) on the real datasets (GeoLife, Truck, and Wild-Baboon). Recall that GTM* is the space-efficient version of GTM.

Figure 18 plots the average response time for different trajectory

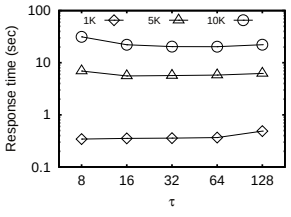
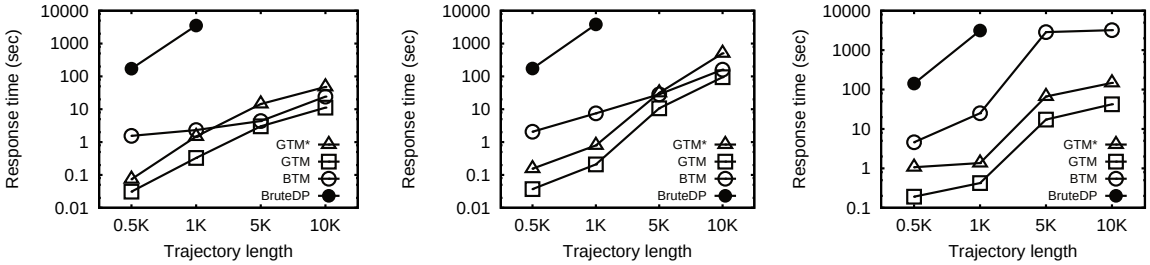
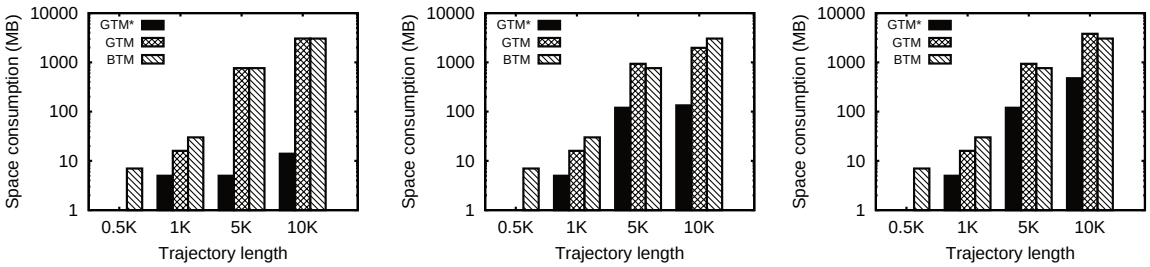


Figure 17: GTM, effect of group size τ



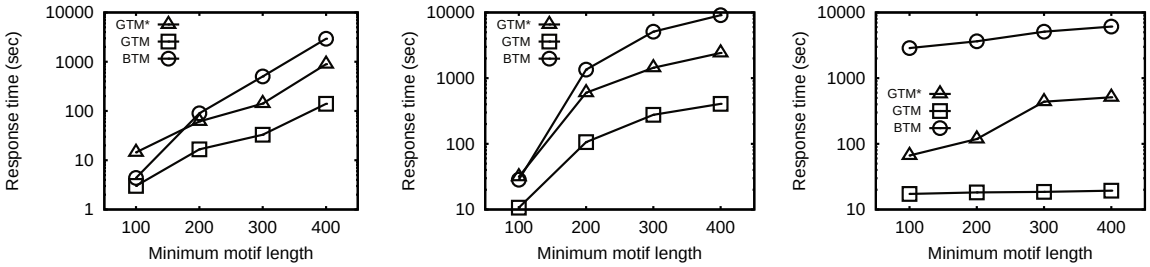
(a) GeoLife (b) Truck (c) Wild-Baboon

Figure 18: Response time vs. trajectory length n



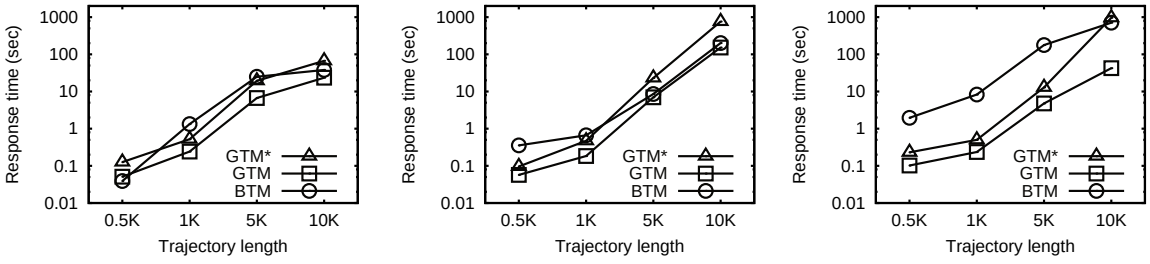
(a) GeoLife (b) Truck (c) Wild-Baboon

Figure 19: Space requirements vs. trajectory length n



(a) GeoLife (b) Truck (c) Wild-Baboon

Figure 20: Response time vs. minimum motif length ξ



(a) GeoLife (b) Truck (c) Wild-Baboon

Figure 21: Response time vs. trajectory length n, two input trajectories

lengths n while fixing $\xi = 100$. BruteDP is prohibitively slow even for small trajectories (e.g., $n = 1000$), thus, we terminate it when it exceeds 2 hours. For the settings where it does terminate within reasonable time, our advanced solutions (i.e., GTM, GTM*) outperform it by 3 orders of magnitude. GTM is the fastest algorithm, with GTM* usually the runner-up. Due to the clear inefficiency of BruteDP, we exclude it from the following experiments.

In Figure 19, we plot the space requirements of BTM, GTM, and GTM* for the same experiment as Figure 18. All methods consume more memory as the trajectory length n increases. As anticipated analytically, the space requirements of BTM and GTM increase sharply with n, but those of GTM* are linear to it. Hence, we consider GTM* as the method of choice for very long trajectories, and the method that strikes the most favourable trade-off between time and space efficiency.

In Figure 20, we measure response time as we vary the minimum trajectory motif length ξ (with n fixed to 5000). The relative performance of the methods is the same as in the previous experiment. The response time of all solutions increases with ξ. That is because a large ξ disqualifies short motifs with small DFDs, thus making it harder to identify early a small bsf that enables aggressive pruning (see also Figure 14(a)).

For completeness, we evaluate our algorithms for motif discovery between different trajectories too. In Figure 21, we randomly select 10 pairs of input trajectories (from the corresponding real dataset) and report the average response time when varying their length n (for fixed $\xi = 100$). The results demonstrate the efficiency of our approaches in this problem variant too. Their performance is very similar to the case of single input trajectory (Problem 1). The same holds when we vary ξ as well as when we measure space requirements; we omit the respective plots to avoid duplication.

7. CONCLUSION

In this paper, we study the trajectory motif discovery problem using the discrete Fréchet distance (DFD). Our contributions include (i) a suite of novel lower bound functions for DFD, (ii) a grouping-based solution that leverages on multi-level pruning to discover the trajectory motif, and (iii) a space-optimized approach that is both time and space efficient. Our fastest solution is over 3 orders of magnitude faster than the baseline solution. All our algorithms are exact. A promising direction for future work is to devise approximate solutions that trade exactness for shorter running times. Another challenging direction is to apply similar optimizations in order to accelerate other trajectory analysis operations that rely on DFD, such as similarity join, subtrajectory clustering, etc.

Acknowledgement

Man Lung Yiu and Bo Tang were supported by grant GRF 152043/15E from the Hong Kong RGC. Kyriakos Mouratidis was supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

8. REFERENCES

[1] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2), 2014.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, 2005.

[3] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *IJCGA*, 21(03), 2011.

[4] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing large trajectory data sets with SETI. In *CIDR*, 2003.

[5] E. W. Chambers and Y. Wang. Measuring similarity between curves on 2-manifolds via homotopy area. In *Annual Symposium on Computational Geometry*, 2013.

[6] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, 2005.

[7] S. Dodge. *Exploring movement using similarity analysis*. PhD thesis, 2011.

[8] T. Eiter and H. Mannila. Computing discrete fréchet distance. Technical report, Information Systems Department, Technical University of Vienna, 1994.

[9] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, 2007.

[10] J. Gudmundsson, P. Laube, and T. Wolle. Computational movement analysis. In *Springer handbook of geographic information*. 2011.

[11] J. Gudmundsson, A. Thom, and J. Vahrenhold. Of motifs and goals: mining trajectory data. In *GIS*, 2012.

[12] J. Gudmundsson and N. Valladares. A gpu approach to subtrajectory clustering using the fréchet distance. *IEEE Transactions on Parallel and Distributed Systems*, 26(4), 2015.

[13] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *PVLDB*, 1(1), 2008.

[14] C. Jia, M. B. Carson, and J. Yu. A fast weak motif-finding algorithm based on community detection in graphs. *BMC bioinformatics*, 14(1), 2013.

[15] M.-P. Kwan. Interactive geovisualization of activity-travel patterns using three-dimensional geographical information systems: a methodological exploration with a large data set. *Transportation Research Part C: Emerging Technologies*, 8(1), 2000.

[16] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, 2007.

[17] X. Li, J. Han, S. Kim, and H. Gonzalez. Roam: Rule-and motif-based anomaly detection in massive moving object data sets. In *SDM*, volume 7, 2007.

[18] Y. Li, U. Leong Hou, M. L. Yiu, and Z. Gong. Quick-motif: An efficient and scalable framework for exact motif discovery. ICDE, 2015.

[19] A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, and M. B. Westover. Exact discovery of time series motifs. In *SDM*, 2009.

[20] T. Oates, A. P. Boedihardjo, J. Lin, C. Chen, S. Frankenstein, and S. Gandhi. Motif discovery in spatial trajectories using grammar inference. In *CIKM*, 2013.

[21] R. W. Sinnott. Virtues of the haversine. *Sky and Telescope*, 68(2), 1984.

[22] R. Sriraghavendra, K. Karthik, and C. Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *ICDAR*, volume 1. IEEE, 2007.

[23] A. Strandburg-Peshkin, D. R. Farine, I. D. Couzin, and M. C. Crofoot. Shared decision-making drives collective movement in wild baboons. *Science*, 348(6241), 2015.

[24] K. Toohey and M. Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1), 2015.

[25] G. Trajcevski, H. Ding, P. Scheuermann, R. Tamassia, and D. Vaccaro. Dynamics-aware similarity of moving objects trajectories. In *GIS*. ACM, 2007.

[26] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, 2002.

[27] T. Wylie and B. Zhu. Protein chain pair simplification under the discrete fréchet distance. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(6), 2013.

[28] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, 1998.

[29] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang. Detecting moving object outliers in massive-scale trajectory streams. In *SIGKDD*, 2014.

[30] F. Zambelli, G. Pesole, and G. Pavesi. Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Briefings in bioinformatics*, 2012.

[31] Y. Zheng. Trajectory data mining: an overview. *TIST*, 6(3), 2015.

[32] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, 2009.