APPROXIMATING THE GEOMETRIC EDIT DISTANCE

by

Xinyi Li

APPROVED BY SUPERVISORY COMMITTEE:

Kyle Fox, Chair

Ovidiu Daescu

Benjamin Raichel

Copyright © 2019

Xinyi Li

All rights reserved

APPROXIMATING THE GEOMETRIC EDIT DISTANCE

by

XINYI LI, BE

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2019

ACKNOWLEDGMENTS

I would first like to express my special thanks of gratitude to my advisor Kyle Fox for his careful and patient advising and help during my research. I believe he will always be my role model during my future research and teaching work.

I would also like to thank Ovidiu Daescu and Benjamin Raichel as the second readers of this thesis, and the members of my supervisory committee. Thanks to my labmates who helped me a lot when I met a problem during my research. I could always get some ideas by discussing with them when I was stuck on problems.

Finally, I must feel very grateful for the support from my parents and my friends throughout these years. Thanks for comforting and accompanying me when I was suffering, even at night or when they were busy with their own work. Thanks for your patience and listening when I called or texted you at night when I was disappointed and sad. This accomplishment would not have been possible without them. Thank you.

April 2019

APPROXIMATING THE GEOMETRIC EDIT DISTANCE

Xinyi Li, MSCS
The University of Texas at Dallas, 2019

Supervising Professor: Kyle Fox, Chair

Edit distance is a measurement of similarity between two sequences such as strings, point sequences, or polygonal curves. Because many matching problems from a variety of areas, such as signal analysis, bioinformatics, etc., need to be solved in a geometric space, the geometric edit distance (GED) has been studied.

In this thesis, we focus on approximating the GED between two point sequences. Previous work has proved that there is no $O(n^{2-\delta})(\delta > 0)$ exact algorithm unless SETH fails. We present a randomized $O(n \log^2 n)$ time $O(\sqrt{n})$-approximation algorithm. We then generalize our result to give a randomized α-approximation algorithm for any $\alpha \in [1, \sqrt{n}]$, running in time $\tilde{O}(n^2/\alpha^2)$. Both algorithms are Monte Carlo and return approximately optimal solutions with high probability.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Ordered sequences are often studied objects in the context of similarity measurements, because sequence alignment plays a vital role in trajectory comparison and pattern recognition. As a consequence, several metrics have been developed to measure the similarity of two sequences, e.g., Fréchet distance, dynamic time warping, and their variations. Geometric edit distance, a natural extension of the string metric to geometric space, is the focus of this paper. This concept is formally introduced by Agarwal *et al.* [1]; however, the similar idea (extending string edit distance to a geometric space) has been applied in other ways during the past decade. Examples include an l^p-type edit distance for biological sequence comparison [2], ERP (Edit distance with Real Penalty) [3], EDR (Edit Distance on Real sequence) [4], TWED (Time Warping Edit Distance) [5] and a matching framework from Swaminathan *et al.* [6] motivated by computing the similarity of time series and trajectories. See also a survey by Wang *et al.* [7].

Problem statement.

Geometric Edit Distance (GED) is the minimum cost of any matching between two geometric point sequences that respects order along the sequences. The cost includes a constant penalty for each unmatched point.

Formally, let $P =< p_1, ..., p_m >$ and $Q =< q_1, ..., q_n >$ be two point sequences in $\mathbb{R}^d$ for some constant d. A monotone *matching* $\mathcal{M}$ is a set of index pairs $\{(i_1, j_1), ..., (i_k, j_k)\}$ such that for any two elements (i, j) and (i', j') in $\mathcal{M}$, $i < i'$ if $j < j'$. See Fig. 1.1 as an example.

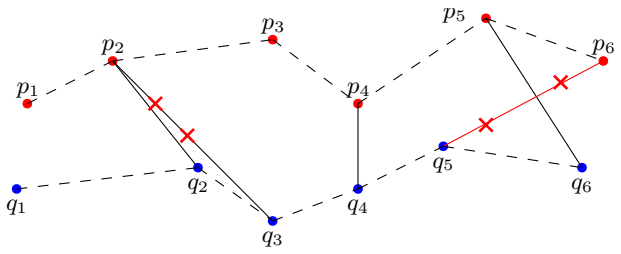


Figure 1.1: An example of a monotone matching. Notice that we do not allow one-to-many matching to multiple points or "crossing" pairs. Unmached points are permitted.

We call every unmatched point a *gap point*. Let $\Gamma(\mathcal{M})$ be the set of all gap points. The *cost* of $\mathcal{M}$ is defined as

$$\delta(\mathcal{M}) = \sum_{(i,j)\in\mathcal{M}} dist(p_j, q_j) + \rho(\Gamma(\mathcal{M})) \tag{1.1}$$

where $dist(p, q)$ is the distance between point p and q (i.e. the Euclidean norm), and $\rho(\Gamma(\mathcal{M}))$ is function about all gap points, which is known as a *gap penalty function*. The use of gap points and the gap penalty function allow us to recognize good matchings even in the presence of outlier points. For geometric edit distance, we use a linear gap function. That is to say, $\rho((\mathcal{M})) = |\Gamma(\mathcal{M})| \cdot \ell$, where ℓ is a constant parameter called the *gap penalty*.
The gap penalty or more general gap penalty function can depend upon the specific application one wishes to address.

Definition 1. The geometric edit distance is the minimum cost of a monotone matching.

We denote the GED between two sequences P, Q as:

$$GED(P, Q) = \min_{\mathcal{M}} \delta(\mathcal{M}) = \min_{\mathcal{M}} \left(\sum_{(i,j)\in\mathcal{M}} dist(p_i, q_j) + |\Gamma(\mathcal{M})| \cdot \ell \right)$$

where the minimum is taken over all monotone matchings. Without loss of generality, we assume $\ell = 1$ throughout the paper.

Prior work.

It is trivial to compute $GED(P, Q)$ in $O(mn)$ time by simply changing the cost of substitution in the original string edit distance (Levenstein distance) dynamic programming algorithm [8]. However, while there is a slightly subquadratic $O(n^2/\log n)$ time algorithm [9] for the string version, it cannot be applied to the geometric case. Very recently, Gold and Sharir [10] proposed an algorithm which can compute GED as well as the closely related dynamic time warping distance (dynamic time warping and the similarity defined discrete Fréchet do not allow unmatched points but may admit one to many matchings in contrast to GED) in $O(n^2 \log\log\log n/\log\log n)$ time in polyhedral metric spaces. Recent papers have shown a conditional lower bound for several sequence distance measures even with some restrictions. Particularly, there is no $O(n^{2-\delta})(\delta > 0)$ algorithm for Fréchet distance [11], dynamic time warping over a constant size alphabet [12] or restricted to one-dimensional curves [13] and string edit distance on the binary alphabet [14, 13]. This latter result implies the same lower bound for our problem, even assuming the sequences consist entirely of $0, 1$-points on the line.

Due to these limitations and difficulties, many researchers have turned to approximation algorithms for these distances. Much work has been done to explore approximate algorithms for Fréchet distance and DTW, including variations of them [15, 16, 17, 18, 19, 1]. For GED, a simple linear time $O(n)$-approximation algorithm was observed by Agarwal *et al.* [1]. Specifically, they first test if matching $\{(1, 1), ..., (n, n)\}$ costs less than the gap penalty; if not, they just unmatch all points. In the same paper, they also offered a subquadratic time (near-linear time in some scenarios) approximation scheme on several well-behaved families of sequences. Using the properties of these families, they reduced the search space to find the optimal admissible path in the dynamic programming graph [1].

A natural approach to approximating GED without assumptions on the sequences is to extend the approximation algorithms for string edit distance to geometric edit distance.

There are numerous approximation algorithms for the string version. In particular, an $O(\sqrt{n})$-approximation algorithm can be acquired easily from an $O(n + k^2)$ time exact algorithm where k is the edit distance [20]. The current best results include papers with $(\log n)^{O(1/\epsilon)}$ [21] and constant approximation ratios [22] with different running time trade-offs; see Gonzalo's survey [23] for more information. However, none of these approaches can be directly utilized in GED since their cost for matched points are in a continuous domain, while the string edit distance is based on the finite alphabet and discrete cost values.

Our results.

Inspired by the above applications and prior work, we commit to finding a faster approach to approximate GED between general point sequences. Here we give the first near-linear time algorithm to compute GED with a sublinear approximation factor. We then generalize our result to achieve a trade-off between the running time and approximation factor. Both of these algorithms are Monte Carlo algorithms, returning an approximately best matching with high probability[1]. To simplify our exposition:

1. We assume all the points located in a plane (i.e., $d = 2$).

2. We assume the input sequences are the same length (i.e., $m = n$).

We can easily extend our results to the unequal case. In fact, our analysis shows that the running time increases by a factor polynomial in d and the approximation ratio is dependent on d.

Theorem 1. *Given two point sequences P and Q in $\mathbb{R}^2$, each with n points, there exists an $O(n \log^2 n)$-time randomized algorithm that computes an $O(\sqrt{n})$-approximate monotone matching for geometric edit distance with high probability.*

[1]We say an event occurs with high probability if it occurs with probability at least $1 - \frac{1}{n^c}$ for some constant $c > 0$.

The intuitive idea behind this algorithm is very simple: check if the GED is less than each of several geometrically increasing g that are less than $O(\sqrt{n})$. For each value of g, we transform the geometric sequences into strings and use the exact algorithm for string edit distance. Precisely, we group points using a randomly shifted grid, and then regarding the points in one cell as an identical character, we use an exact string edit distance algorithm [20] to decide whether the GED is less than g. If it is, we get an $O(\sqrt{n})$ approximate matching. If we never find a matching of cost $O(\sqrt{n})$, we simply leave all points unmatched as this empty matching is an $O(\sqrt{n})$-approximation for GED with high probability. We need to generalize the exact string edit distance algorithm for our next result, so we begin by introducing it in Chapter 2. In Chapter 3, we give the details for our $O(\sqrt{n})$-approximation algorithm.

Theorem 2. *Given two point sequences P and Q in $\mathbb{R}^2$, each with n points, there exists an $O(n \log^2 n + \frac{n^2}{\alpha^2} \log n)$-time randomized algorithm that computes an $O(\alpha)$-approximate monotone matching for geometric edit distance with high probability for any $\alpha \in [1, \sqrt{n}]$.*

The second algorithm uses similar techniques to the former, except we can no longer use the string edit distance algorithm as a black box. In particular, we cannot achieve our desired time-approximation trade-off by just directly altering some parameters in our first algorithm. We discuss why in Chapter 4. Instead, we develop a constant-factor approximation algorithm to compute the edit distance using what we refer to as *simplified* geometric costs. The algorithm for simplified geometric costs is based on the exact algorithm for string edit distance [20] but necessarily more complicated to handle geometric distances. In Chapter 4, we present the algorithm for Theorem 2 assuming the simplified geometric edit distance algorithm is given as a black box. The algorithm for simplified geometric edit distance is presented in Chapter 5.

CHAPTER 2

PRELIMINARIES

2.1 Fast computation of string edit distance

We first describe a known exact algorithm for the string edit distance that runs in $O(n+k^2)$ time if the edit distance is at most k [20]. We refer to the procedure as $SED(S,T,k)$.

2.1.1 Dynamic programming matrix and its properties

Let $S =< s_1, s_2, ...s_n >$ and $T =< t_1, t_2, ..., t_n >$ be two strings of length n. Let D denote a $(n+1) \times (n+1)$ matrix where $D(i,j)$ is the edit distance between substrings $S_i =< s_1, s_2, .., s_i >$ and $T_j =< t_1, t_2, ..., t_j >$. We give a label h to every diagonal in this matrix such that for any entry (i,j) in this diagonal $j = i + h$. See Fig. 2.1 (a). To better serve us when discussing geometric edit distance, we aim to minimize the number of insertions and deletions to turn S into T *only*; we consider substitution to have infinite cost.[1] There are four important properties in this matrix which are used in the $O(n+k^2)$ time algorithm.

Property 1.

$$D(i,j) = \min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+|s_i t_j| \end{cases}$$

where

$$|s_i t_j| = \begin{cases} 0, & \text{if } s_i = t_j \\ \infty, & \text{otherwise} \end{cases}.$$

Property 2. $D(i,0) = i$, and $D(0,j) = j$.

[1]Computing this variant of the string edit distance is morally closer to computing the longest common subsequnce length of the strings than to traditional Levenshtein distance, but we stick with "edit distance" for simplicity.

Property 3. $D(i, i+h)$ is even if and only if h is even.

Property 4. $D(i,j) - D(i-1, j-1) \in \{0, 2\}$.

Property 4 can be easily derived from Property 3 and induction on i + j (see Lemma 3 of [24]). From Property 4, we know all the diagonals are non-decreasing. That is to say, all values on diagonal k are greater than $|k|$ considering Property 2. So, we can just search the band from diagonal $-k$ to k if the edit distance between S and T is at most k.

2.1.2 Algorithm for edit distance at most k

We use a greedy approach to fill the entries along each diagonal. For each value $e \in \{0, ...k\}$ (the outer loop), we locate the elements whose value is e by inspecting diagonal $-e$ to e (the inner loop). Finally, we return the best matching if $D(n, n)$ is covered by the above search. Otherwise, the edit distance is greater than k.

The key insight is that we can implicitly find all entries containing e efficiently in each round. We first define $L_{h,e}$ as the row index of a *farthest* e entry in diagonal h.

Definition 2. $L_{h,e} = max\{i | D(i, i+h) = e\}$

It is straightforward to recover an optimal matching by $L_{h,e}$.

Observe that all values on diagonal h are at least $|h|$, which means that we can define our initial values as:

$$L_{h,h-2} = \begin{cases} |h| - 1, & \text{if } h < 0; \\ -1, & \text{otherwise} \end{cases}$$

where $h \in [-k, k]$.

Observe, $L_{h,e} \geq r$, where $r = max\{L_{h-1,e-1}, L_{h+1,e-1} + 1\}$, because the farthest entry of value e must reach at least row r on diagonal h by Properties 1 and 4. Further, if $D(r, r+h) = e$, then $D(r+1, r+1+h) = e$ as well if $s_{r+1} = t_{r+1+h}$. In each inner loop,

Algorithm 1: Computing $L_{h,e}$ in each inner loop

```
1 r = max{L_{h-1,e-1}, L_{h+1,e-1} + 1};
2 while s_{r+1} == t_{r+1+h} and r ≤ n and r + h ≤ n do
3 |   r = r + 1 ;                                          /* slide */
4 end
5 if r > n or r + h > n then
6 |   L_{h,e} = ∞;
7 else
8 |   L_{h,e} = r;
9 end
```

we can compute $L_{h,e}$ with Algorithm 1. We call lines 2 through 4 "the slide". Fig. 2.1 (b) demonstrates this process. We can perform slides in constant time each after some $O(n)$-time preprocessing at the beginning of the algorithm [20], so the overall running time is $O(n+k^2)$.

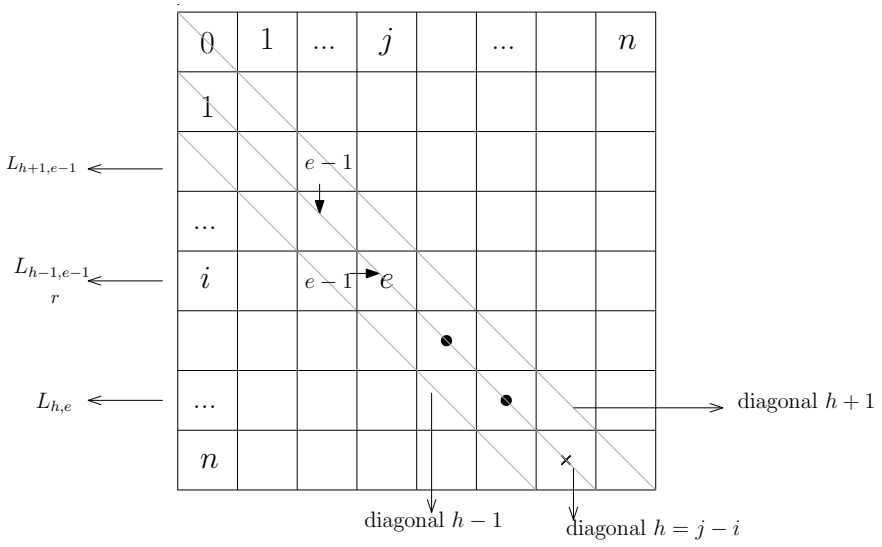


Figure 2.1: (a) The diagonal containing any entry $(i, i+h)$ is diagonal h. (b) Let $r = max\{L_{h-1,e-1}, L_{h+1,e-1}+1\}$. Then, $D(r, r+h) = e$. The algorithm keeps sliding until finding an entry representing distinct characters. A circle means the corresponding two characters are the same; a cross means they are different.

2.2 Random partition by shifted grid

In this section, we introduce a random partition technique, which we used in our algorithms. This technique partitions nearby points into common groups and distant points into different groups. We use a randomly shifted grid to realize this ideal, see for example [25].

We cover the space $\mathbb{R}^d$ with a grid. Let the side length of each grid cell be Δ, and b be a random vector chosen uniformly from $[0, \Delta]^d$. Starting from an arbitrary position, the grid shifts b_i units in each dimension i. For a point p, let $id_{\Delta,b}(p)$ denote the cell which contains p in this configuration. We consider two points $p = (x_1, x_2, ...x_d)$, and $q = (y_1, y_2, ...y_n)$ in this space. We want to know the probability that p and q lie in two different cells, i.e., $P(id_{\Delta,b}(p) \neq id_{\Delta,b}(q))$. We first consider the one-dimension case.

Lemma 1. *If $d = 1$, $P(id_{\Delta,b}(p) \neq id_{\Delta,b}(q)) = \min\{\frac{|x-y|}{\Delta}, 1\}$.*

Proof. If $d = 1$, space is a real line, and p, q are two points in this line. Let the coordinates of p, q is x, y respectively. We can regard each cell as an interval with length Δ; see Fig.2.2. Imagine pulling this line to the left or right for b distance, then p and q can be distributed into one cell or different cells.

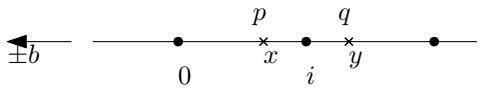


Figure 2.2: Random partition in one dimension

1. If $|x - y| > \Delta$, then p, q can never be in the same interval. In this case, $\frac{|x-y|}{\Delta} > 1$ and $P(id_{\Delta,b}(q) \neq id_{\Delta,b}(q)) = \min\{\frac{|x-y|}{\Delta}, 1\} = 1$.

2. If $|x - y| \leq \Delta$, the probability that p and q are not in the same cell is probability that one boundary point i falls in the region $[x, y]$. Because b is uniformly selected from $[0, \Delta]$, we get $P(id_{\Delta,b}(q) \neq id_{\Delta,b}(q)) = \min\{\frac{|x-y|}{\Delta}, 1\} = \frac{|x-y|}{\Delta}$.

□

We can extend this result to higher dimensions easily because two points are apart as long as the projections of points in any dimension are grouped into different cells. So,

Lemma 2. *For any d, have* $P(id_{\Delta,b}(p) \neq id_{\Delta,b}(q)) \leq \min\{\sum_{j=1}^{d} \frac{|x_j - y_j|}{\Delta}, 1\}$.

We use this observation in our algorithms and set different Δ in our two algorithms.

CHAPTER 3

$O(\sqrt{n})$-APPROXIMATION FOR GED

Recall that the main part of our algorithm is a decision procedure to check if the GED is less than a guess variable g. There are two steps in this process:

1. Transform the point sequences into strings. To be specific, we partition nearby points into common groups and distant points into different groups to simulate the identical characters and different characters in the string version.

2. Perform the string algorithm.

We have explained how to partition the points in Section 2.2. As we claimed before, we consider our problem in plane, i.e., $d = 2$. In this algorithm, we set $\Delta = \frac{g}{\sqrt{n}}$. We then do transformation by this partition as we state above in plane. We give an outline of our algorithm.[1] Here, c is a sufficiently large constant. We will later analyze the approximation factor and running time in Sections 3.1 and 3.2.

3.1 Time complexity

We claim the running time for Algorithm 2 is $O(n \log^2 n)$. Computing $\sum_{i=1}^{n} dist(p_i, q_i)$ takes $O(n)$ time. In the inner loop, the transformation operation (line 7) takes $O(n)$ time assuming use of a hashtable. The running time for $SED(S, T, 12\sqrt{n}+2g)$ is $O(n)$ for $g \leq \sqrt{n}$. Summing over the outer loop and inner loop, the overall running time for Algorithm 2 is

$$\sum_{i=1}^{\lceil \lg \sqrt{n} \rceil} \sum_{j=1}^{\lceil c \lg n \rceil} O(n) = O(n \log^2 n).$$

[1] We use lg to denote the logarithm of base 2.

Algorithm 2: $O(\sqrt{n})$-approximation algorithm for GED

Input: Point sequences P and Q
Output: An approximately optimal matching for GED

```
if ∑_{i=1}^{n} dist(p_i, q_i) ≤ 1 then
    return matching {(1,1), ..., (n,n)}
else
    for i := 0 to ⌈lg √n⌉ do
        g = 2^i;
        for j := 1 to ⌈c lg n⌉ do
            Transform P, Q to strings S, T using a randomly shifted grid
            out = SED(S, T, 12√n + 2g)
            if out ≠ false then
                return out
            end
        end
    end
    return the empty matching
end
```

3.2 Approximation ratio

In this section, we prove that Algorithm 2 returns an $O(\sqrt{n})$-approximate matching with high probability. For any monotone matching $\mathcal{M}$, we define $C_S(\mathcal{M})$ as the cost of edit operations for this matching in the string case and $C_G(\mathcal{M})$ is $\delta(\mathcal{M})$ as defined in (1.1) for the geometric case.

Let $\mathcal{M}_G^*$ be the optimal matching for geometric edit distance, and $\mathcal{M}_S^*$ be the optimal matching for string edit distance during one iteration of the loop. Our final goal is to establish the relationship between $C_G(\mathcal{M}_G^*)$ and $C_G(\mathcal{M}_S^*)$.

Lemma 3. *If $GED(S,T) \le g$, with a probability at least $1 - \frac{1}{n^c}$, at least one of the $\lceil c \lg n \rceil$ iterations of the inner for loop will return a matching $\mathcal{M}_S^*$, where $C_S(\mathcal{M}_S^*) \le 12\sqrt{n} + 2g$.*

Proof. Let $\mathcal{M}$ be a monotone matching, and let $UM_{\mathcal{M}}$ be the set of unmatched indices. There are four subsets of pairs in $\mathcal{M}$:

- $OC_{\mathcal{M}}$: In each pair, both indices' points fall into One cell, and the distance between the two points is less and equal to $\frac{g}{\sqrt{n}}$ (Close).
- $OF_{\mathcal{M}}$: In each pair, both indices' points fall into One cell, and the distance between the two points is larger than $\frac{g}{\sqrt{n}}$ (Far).
- $DC_{\mathcal{M}}$: In each pair, the indices' points are in Different cells, and the distance between the two points is less and equal to $\frac{g}{\sqrt{n}}$ (Close).
- $DF_{\mathcal{M}}$: In each pair, the indices' points are in Different cells and the distance between the two points is larger than $\frac{g}{\sqrt{n}}$ (Far).

These sets are disjoint, so

$$\begin{aligned} C_G(\mathcal{M}_G^*) =& |UM_{\mathcal{M}_G^*}| + \sum_{(i,j)\in OC_{\mathcal{M}_G^*}} dist(p_i, q_j) + \sum_{(i,j)\in OF_{\mathcal{M}_G^*}} dist(p_i, q_j) \\ &+ \sum_{(i,j)\in DC_{\mathcal{M}_G^*}} dist(p_i, q_j) + \sum_{(i,j)\in DF_{\mathcal{M}_G^*}} dist(p_i, q_j). \end{aligned} \tag{3.1}$$

Recall that there is no substitution operation in our version of the string case. So to understand optimal matchings for string edit distance, we must unmatch all the pairs in $DC_{\mathcal{M}_G^*}$ and $DF_{\mathcal{M}_G^*}$, forming a new matching $\mathcal{M}_G^{*'}$. Points in one cell are regarded as identical characters while those in different cells are different characters. Therefore,

$$\begin{aligned} C_S(\mathcal{M}_G^{*'}) &= |UM_{\mathcal{M}_G^*}| + 0 \cdot (|OC_{\mathcal{M}_G^*}| + |OF_{\mathcal{M}_G^*}|) + 2 \cdot (|DC_{\mathcal{M}_G^*}| + |DF_{\mathcal{M}_G^*}|) \\ &= |UM_{\mathcal{M}_G^*}| + 2 \cdot (|DC_{\mathcal{M}_G^*}| + |DF_{\mathcal{M}_G^*|}). \end{aligned}$$

Observe that there are at most $\frac{g}{g/\sqrt{n}} = \sqrt{n}$ pairs in $DF_{\mathcal{M}_G^*}$ if $C_G(\mathcal{M}_G^*) \leq g$. Therefore,

$$\begin{aligned} C_S(\mathcal{M}_S^*) &\leq C_S(\mathcal{M}_G^{*'}) \\ &= |UM_{\mathcal{M}_G^*}| + 2|DC_{\mathcal{M}_G^*}| + 2|DF_{\mathcal{M}_G^*}| \\ &\leq g + 2\sqrt{n} + 2|DC_{\mathcal{M}_G^*}| \end{aligned} \tag{3.2}$$

For any two close points p_i, q_j, let $P_C(i,j)$ be the probability that p_i and q_j are assigned into different cells. From Lemma 2, we can infer

$$P_C(i,j) \leq \frac{2dist(p_i,q_j)}{g/\sqrt{n}}.$$

Then,

$$\begin{aligned} E(|DC_{\mathcal{M}_G^*}|) &= \sum_{(i,j)\in\mathcal{M}_G^*} P_C(i,j) \\ &\leq \sum_{(i,j)\in\mathcal{M}_G^*} \frac{2dist(p_i,q_j)}{g/\sqrt{n}} \\ &\leq 2\sqrt{n}. \end{aligned} \tag{3.3}$$

Therefore,

$$E(C_S(\mathcal{M}_S^*)) \leq 6\sqrt{n} + g.$$

By Markov's inequality,

$$P[C_S(\mathcal{M}_S^*) \geq 12\sqrt{n} + 2g] \leq \frac{1}{2}.$$

In other words, $SED(S,T,12\sqrt{n}+2g)$ will fail with probability at most $\frac{1}{2}$ if $GED \leq g$. So, if we test $SED(S,D,12\sqrt{n}+2g)$ at least $c\lg n$ times, at least one iteration will return a value if $GED \leq g$ with a probability greater than or equal to

$$1 - \prod_{1}^{\lceil c\lg n\rceil} P[C_S(\mathcal{M}_S^* \geq 12\sqrt{n}+2g)] \geq 1 - \prod_{1}^{\lceil c\lg n\rceil} \frac{1}{2} = 1 - \frac{1}{n^c}.$$

We conclude the proof of Lemma 3. □

According to Lemma 3, if all test procedures return false, we can say $C_G(\mathcal{M}_G^*) > g$ with high probability; otherwise, we obtain a matching $\mathcal{M}_S^*$ and $C_S(\mathcal{M}_S^*) \leq 12\sqrt{n}+2g$.

We now consider $C_G(\mathcal{M}_S^*)$. For a matching $\mathcal{M}$, let $U_{\mathcal{M}}$ be the set of unmatched indices. Observe, for all $(i,j) \in \mathcal{M}_S^*$, points p_i and q_j lie in the same grid cell. Therefore, $dist(p_i,q_j) \leq$

$\frac{\sqrt{2}g}{\sqrt{n}}$ if $(p_i, q_j) \in \mathcal{M}_S^*$. We have:

$$\begin{aligned} C_G(\mathcal{M}_S^*) &= |U_{\mathcal{M}_S^*}| + \sum_{(i,j)\in\mathcal{M}_S^*} dist(p_i, q_j) \\ &\leq 12\sqrt{n} + 2g + n \cdot (\sqrt{2} \cdot \frac{g}{\sqrt{n}}) \\ &\leq 12\sqrt{n} + 2g + \sqrt{2}g\sqrt{n} \end{aligned} \tag{3.4}$$

If $GED(P, Q) \leq \sqrt{n}$, then, with high probability, we obtain a matching $\mathcal{M}_S^*$ during an iteration where $GED(P, Q) \geq \frac{1}{2}g$. The cost of this matching is at most $12\sqrt{n}+2g+\sqrt{2}g\sqrt{n} = O(\sqrt{n})GED(P, Q)$. The same approximation bound holds if $GED(P, Q) > \sqrt{n}$, whether or not we find a matching during the outer for loop. We conclude the proof of Theorem 1.

CHAPTER 4

$O(\alpha)$-APPROXIMATION FOR GED

We now discuss our $O(\alpha)$-approximation algorithm for any $\alpha \in [1, \sqrt{n}]$. A natural approach for extending our $O(\sqrt{n})$-approximation is using the same reduction to string edit distance but let the cell's side length be a variable depending on the approximation factor α. However, this method does not appear to work well.

Let Δ_α be the cell's side length which depends on the approximation factor α. For our analysis we need

$$C_G(\mathcal{M}_S^*) \leq g \cdot O(\alpha).$$

There can be at most n matched pairs in $\mathcal{M}_S^*$. Following (3.4), we derive $n \cdot \Delta_\alpha \leq g \cdot O(\alpha)$, implying

$$\Delta_\alpha \leq O(\frac{g\alpha}{n}).$$

On the other hand, we require $C_S(\mathcal{M}_S^*) \leq g \cdot O(\alpha)$ in our analysis; in particular, we need to replace the $2\sqrt{n}$ in (3.2) with $g \cdot O(\alpha)$. We derived $2\sqrt{n}$ as $2\frac{g}{\Delta_\alpha}$. We now need $2\frac{g}{\Delta_\alpha} \leq g \cdot O(\alpha)$, implying

$$\Delta_\alpha \geq \Omega(\frac{1}{\alpha}).$$

This is fine for $\alpha = \sqrt{n}$ or for large values of g. But for small α and small g, we cannot have both inequalities be true. Therefore, we introduce simplified geometric costs that let us ignore the second inequality.

Simplified geometric cost.

Our simplified geometric costs are again based on a randomly shifted grid. We define the distance between two points using following formula.

Definition 3. Let Δ be the side length of a grid cell. For any two points p_i and q_j,

$$dist_{SG}(p_i, q_j) = \begin{cases} 0, & \text{if } p_i, q_j \text{ are in one cell;} \\ \max\{\Delta, dist(p_i, q_j)\}, & \text{otherwise.} \end{cases}$$

Then the simplified geometric edit distance between P and Q is defined as

$$SGED(P, Q) = \min_{\mathcal{M}} \left(\sum_{(i,j)\in\mathcal{M}} dist_{SG}(p_i, q_j) + |\Gamma(\mathcal{M})| \right).$$

where $\mathcal{M}$ is a monotone matching and $\Gamma(\mathcal{M})$ is the set of gap points of $\mathcal{M}$. We can obtain an $O(\alpha)$-approximation algorithm by altering the bound in the outer loop and the test procedure in Algorithm 2.

Algorithm 3: $O(\sqrt{\alpha})$-approximation algorithm

Input: Point Sequences P and Q
Output: An approximately optimal matching for GED

```
if ∑_{i=1}^{n} dist(p_i, q_i) ≤ 1 then
    return matching {(1,1), ..., (n,n)}
else
    for i := 0 to ⌈lg n/α⌉ do
        g = 2^i
        for j := 1 to ⌈c lg n⌉ do
            out = SGED(P, Q, 6g)
            if out ≠ false then
                return out
            end
        end
    end
    Return the empty matching
end
```

$SGED(P, Q, g)$ is an approximation algorithm which returns an $O(1)$-approximate matching for simplified geometric edit distance using grid cells of side length $\Delta = \frac{g\alpha}{n}$ if $SGED(P, Q) \leq g$; otherwise, it returns false.

4.1 Time complexity

We will introduce this $O(1)$-approximation algorithm in Chapter 5. Its running time is $O(n + \frac{gn}{\alpha})$. Thus, the overall running time of our second algorithm is

$$O(n) + \sum_{i=0}^{\lceil \lg \frac{n}{\alpha} \rceil} \sum_{j=1}^{\lceil c \lg n \rceil} O(n + \frac{g \cdot n}{\alpha}) = \sum_{i=0}^{\lceil \lg \frac{n}{\alpha} \rceil} O(n \log n + \frac{2^i n}{\alpha} \log n)$$
$$= O(n \log^2 n + \frac{n^2}{\alpha^2} \log n).$$

4.2 Proof of correctness

We introduce some additional notations to those used in Section 3.2.

Let $C_{SG}(\mathcal{M})$ be the cost of matching $\mathcal{M}$ using $dist_{SG}$. Let $\mathcal{M}^*_{SG}$ be the optimal matching for simplified geometric edit distance. Let $\mathcal{M}^{*'}_{SG}$ be the matching returned by $SGED(P, Q, 6g)$.

We have the following lemma.

Lemma 4. *If $GED(P, Q) \leq g$, with a probability at least $1 - \frac{1}{n^c}$, at least one of the $\lceil c \lg n \rceil$ iterations will return a matching $\mathcal{M}^{*'}_{SG}$.*

Proof. Similar to (3.1), we have

$$C_{SG}(\mathcal{M}^*_G) = |UM_{\mathcal{M}^*_G}| + 0 \cdot (|OC_{\mathcal{M}^*_G}| + |OF_{\mathcal{M}^*_G}|)$$
$$+ \sum_{(i,j) \in DC_{\mathcal{M}^*_G}} dist_{SG}(p_i, q_j) + \sum_{(i,j) \in DF_{\mathcal{M}^*_G}} dist_{SG}(p_i, q_j)$$
$$= |UM_{\mathcal{M}^*_G}| + \Delta \cdot |DC_{\mathcal{M}^*_G}| + \sum_{(i,j) \in DF_{\mathcal{M}^*_G}} dist(p_i, q_j).$$

If $C_G(\mathcal{M}^*_G) \leq g$, then

$$C_{SG}(\mathcal{M}^*_{SG}) \leq C_{SG}(\mathcal{M}^*_G) = |UM_{\mathcal{M}^*_G}| + \Delta \cdot |DC_{\mathcal{M}^*_G}| + \sum_{(i,j) \in DF_{\mathcal{M}^*_G}} dist(p_i, q_j)$$
$$\leq g + \Delta \cdot |DC_{\mathcal{M}^*_G}|.$$

Using the same algebra as (3.3), we have $E(|DC_{\mathcal{M}^*_G}|) \leq \frac{2g}{\Delta}$. So,

$$E(C_{SG}(\mathcal{M}^*_{SG})) \leq g + \Delta \cdot \frac{2g}{\Delta} = 3g.$$

According to Markov's inequality, we know

$$P(C_{SG}(\mathcal{M}^*_{SG}) \geq 6g) \leq \frac{1}{2}.$$

In Chapter 5, we prove that if $C_{SG}(\mathcal{M}^*_{SG}) = SGED(P,Q) \leq 6g$, then $SGED(P,Q,6g)$ will return a constant approximate matching $\mathcal{M}^{*'}_{SG}$. So, if we test $SGED(P,Q,6g)$ $\lceil c \lg n \rceil$ times, with a probability at least $1 - \frac{1}{n^c}$, at least one $SGED(P,Q,6g)$ will return a matching $\mathcal{M}^{*'}_{SG}$. We conclude the proof of Lemma 4. □

Finally, returning to standard Euclidean distance,

$$\begin{aligned}
C_G(\mathcal{M}^{*'}_{SG}) &= |U_{\mathcal{M}^{*'}_{SG}}| + \sum_{(i,j)\in DC_{\mathcal{M}^{*'}_{SG}}} dist(p_i,q_j) + \sum_{(i,j)\in DF_{\mathcal{M}^{*'}_{SG}}} dist(p_i,q_j) \\
&\quad + \sum_{(i,j)\in OC_{\mathcal{M}^{*'}_{SG}}} dist(p_i,q_j) + \sum_{(i,j)\in OF_{\mathcal{M}^{*'}_{SG}}} dist(p_i,q_j) \\
&\leq O(1)\cdot 6g + \sum_{(i,j)\in OC_{\mathcal{M}^{*'}_{SG}}} dist(p_i,q_j) + \sum_{(i,j)\in OF_{\mathcal{M}^{*'}_{SG}}} dist(p_i,q_j) \\
&\leq O(1)\cdot 6g + n\cdot\sqrt{2}\Delta.
\end{aligned}$$

Recall, $\Delta = \frac{g\alpha}{n}$. If $GED(P,Q) = C_G(M^*_G) \geq \frac{1}{2}g$, then $C_G(\mathcal{M}^{*'}_{SG}) \leq O(g\alpha) = O(\alpha) \cdot GED(P,Q)$. Using the same argument as in Theorem 1, we conclude our proof of Theorem 2.

CHAPTER 5

O(1)-APPROXIMATION ALGORITHM FOR *SGED*

We modify the string edit distance algorithm SED(S, T) to obtain a constant-approximate matching $\mathcal{M}'^{*}_{SG}$ for simplified geometric edit distance.

5.1 Labeling the dynamic programming matrix

Notations and properties.

Similar to the string algorithm, we have a dynamic programming matrix; $D'(i, j)$ is the simplified geometric edit distance between subsequence $P_i =< p_1, ..., p_i >$ and $Q_j =< q_1, ..., q_j >$. This matrix also meets Property 1 stated earlier except that we use $dist_{SG}(p_i, q_j)$ instead of $|s_i t_j|$. In addition, we also have the following property which is a refinement of Property 4.

Property 5. $D'(i, j) - D'(i-1, j-1) \in [0, 2]$.

Clearly, the upper bound is 2 (just unmatch p_i and q_j). The lower bound can be proved by induction. Because the values in any diagonal are non-decreasing, we can still search from diagonal $-k$ to k if we know $SGED(P, Q)$ is less and equal to k.

Label rules.

To obtain an approximate matching for simplified geometric edit distance, we now label each entry in the dynamic programming matrix with an approximately tight lower bound on its value. Let $LA(i, j)$ be the label of entry (i, j) and $L'_{h,e}$ be the row index of the farthest entry whose label is e in diagonal h.

Definition 4. $L'_{h,e} = \max\{i | LA(i, i+h) = e\}$.

For each value $e = 0$ to k, for each diagonal h, we (implicitly) assign labels e to each entry on diagonal h.

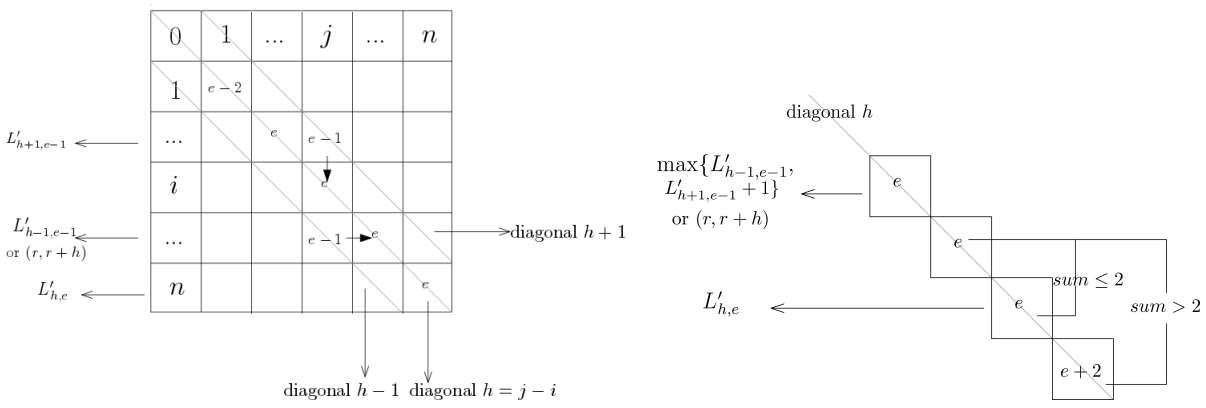
1. If $h = -e$ or e, that means we did not assign any label in this diagonal in the previous iterations. Then, we label the first entry in diagonal h as e, i.e., if $h = -e$, $LA(|h|, 0) = e$; otherwise, $LA(0, h) = e$.

2. We define a *start entry* $(r, r+h)$ for each diagonal h. If $h = -e$ or e, r is the row index of the first entry in diagonal h; otherwise, $r = \max\{L'_{h-1,e-1}, L'_{h+1,e-1} + 1\}$.

3. We assign the label e to the entries between $(r, r+h)$ and $(r+s, r+h+s)$ where $\sum_{i=r+1}^{s} dist_{SG}(p_i, q_{i+h}) \leq 2$ and $\sum_{i=r+1}^{s+1} dist_{SG}(p_i, q_{i+h}) > 2$. $L'_{h,e} = r+s$. These entries correspond to a slide in the string algorithm.

4. Finally, if $(r-1, r+h-1)$ is unlabeled, we go backward up the diagonal labeling entries as e until we meet a entry that has been assigned a label previously. (Again, this step is implicit. As explained below, the actual algorithm only finds the $L'_{h,e}$ entries.)

Fig. 5.1 shows our rules.



(a) Notations and labels for the boundary entries.

(b) Label entries following step 3.

Figure 5.1: Notations and rules for approximating SGED

We obtain the following properties:

Property 6. $LA(i, i+h) - LA(i+1, i+1+h) \in \{0, 2\}$.

Property 7. $LA(i, i+h) - LA(i-1, i+h) \in \{-1, 1\}$ and $LA(i, i+h) - LA(i, i+h-1) \in \{-1, 1\}$.

Assume we have set the initial values. Our algorithm only needs to compute each $L'_{h,e}$ as before. See Algorithm 4. As before, the corresponding matching can be recovered easily after computing all $L'_{h,e}$. See Section 5.3 to get a more clear idea of the matching in our algorithm.

Algorithm 4: Computing $L'_{h,e}$ for a fixed h and e

```
r := max{(L'_{h-1,e-1}), (L'_{h+1,e-1} + 1)}
r := r + 1
sum := dist_SG(p_r, q_{r+h})
while r ≤ n and r + h ≤ n and sum ≤ 2 do
    r := r + 1
    sum := sum + dist_SG(p_r, q_{r+h})
end
if r > n or r + h > n then
    L'_{h,e} := ∞
else
    L'_{h,e} := s
end
```

5.2 Time complexity

Using the same process as in [20], we can slide down maximal sequences of consecutive entries $(r, r+h)$ with $dist_{SG}(p_r, q_{r+h}) = 0$ in constant time per slide after $O(n)$ preprocessing. For $dist_{SG}(p_r, q_{r+h}) \neq 0$, we know $dist_{SG}(p_r, q_{r+h}) \geq \Delta = \frac{k\alpha}{n}$. So, we only need to manually add distances and restart faster portions of the slide $\frac{2}{\Delta} = \frac{2n}{k\alpha}$ times. Thus, the total running time is

$$O(n + \sum_{e=0}^{k} \sum_{h=-e}^{e} \frac{n}{k\alpha}) = O(n + k^2 \cdot \frac{n}{k\alpha}) = O(n + \frac{kn}{\alpha}).$$

5.3 Proof of correctness

Bounding entries by labels.

We have the following lemma.

Lemma 5. *For every entry* (i, j), $LA(i, j) \leq D'(i, j)$.

In particular $LA(n, n) \leq SGED(P, Q)$.

Proof. From Property 5, we only need to prove e is the lower bound of the first entry whose label is e in each diagonal h.

We proceed by induction on e.

1. If $e = 0$, we only label the first entry in diagonal 0 as 0. We have $0 \leq D'(0, 0) = 0$. If $e = 1$, then for diagonals 1 and -1, we have $1 \leq D'(0, 1) = D'(1, 0) = 1$.

2. Assume Lemma 5 for labels less than e. For e, we consider the diagonals $h = -e$ to e: If $h = -e$ or e, we know $e \leq D'(|h|, 0) = e$ or $e \leq D'(0, h) = e$.

   Otherwise, let $(f, f + h)$ be the first entry whose label is e. From Property 6, $f = L'_{h,e-2} + 1$. Fig. 5.2 shows the notations. From the refined Property 1, we need to discuss three cases:

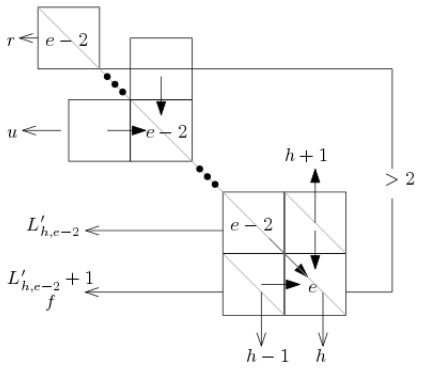


Figure 5.2: We compute the lower bound of entries which are labeled as e.

(a) $D'(f, f+h) = D'(f-1, f+h) + 1$.

From Property 7, we know $LA(f-1, f+h) = e-1$ or $e+1$.

- If $LA(f-1, f+h) = e-1$, $D'(f-1, f+h) \geq e-1$ from our assumption. So, $D'(f, f+h) = D'(f-1, f+h) + 1 \geq e-1+1 = e$.
- If $LA(f-1, f+h) = e+1$, then we know $L'_{h+1,e-1}$ is less than $f-1$. From non-decreasing property, $e-1 \leq D'(L'_{h+1,e-1}, L'_{h+1,e-1}+h+1) \leq D'(f-1, f+h-1)$.

(b) $D'(f, f+h) = D'(f, f+h-1) + 1$.

This case is similar to the above.

(c) $D'(f, f+h) = D'(f-1, f+h-1) + dist_{SG}(p_f, q_{f+h})$.

$LA(f-1, f+h-1) = e-2$, because $f-1 = L'_{h,e-2}$. Let r be the row index of the first entry to slide with label $e-2$ in diagonal h, i.e., $r = \max\{L'_{h-1,e-3}, L'_{h+1,e-3}+1\}$. See Fig. 5.2. We define u as the row index of the first entry walking backward from $(f, f+h)$ along diagonal h where $D'(u, u+h) = \min\{D'(u, u+h-1), D'(u-1, u+h-1)\} + 1$.

- If $u > r$, like Fig. 5.2, then $u > L'_{h-1,e-3}$ and $u-1 > L'_{h+1,e-3}$. Combining our assumption, we have

$$D'(u, u+h-1) \geq D'(L'_{h-1,e-3}+1, L'_{h-1,e-3}+h) \geq e-1$$

and

$$D'(u-1, u+h) \geq D'(L'_{h+1,e-3}+1, L'_{h+1,e-3}+h+2) \geq e-1.$$

So,

$$D'(u, u+h) = \min\{D'(u, u+h-1), D'(u-1, u+h-1)\} + 1 \geq e-1+1$$

implying $D'(u, u+h) \geq e$. Recall $f \geq u$, so $D'(f, f+h) \geq e$.

- If $u \leq r$, then

$$D'(f, f+h) = D'(r, r+h) + \sum_{i=r+1}^{f} dist_{SG}(p_i, q_{i+h})$$
$$> e - 2 + 2 = e.$$

Examining all cases, we conclude the proof of Lemma 5.

□

The bounds for the approximate matching $C_{SG}(\mathcal{M}^{*'}_{SG})$.

From Algorithm 4, we note the label increases only if we decide not to match a point in Line 1, and slide only if do match points. So,

$$C_{SG}(\mathcal{M}^{*'}_{SG}) = |U_{\mathcal{M}^{*'}_{SG}}| + \sum_{(i,j)\in\mathcal{M}^{*'}_{SG}} dist_{SG}(p_i, q_j)$$
$$\leq LA(n,n) + 2 \cdot LA(n,n) \leq 3LA(n,n) \leq 3SGED(P,Q).$$

We obtain an $O(1)$-approximation algorithm to compute $SGED(P,Q)$.

CHAPTER 6

CONCLUSION

We gave two new randomized algorithms to compute the approximate geometric edit distance. One has a sublinear approximation factor and runs in near linear time, and the other one features a tradeoff between approximation factor and running time. Our running time is strictly better than the best known $O(n \log n + \frac{n^2}{\alpha})$ running time for $O(\alpha)$-approximating the discrete Fréchet distance as long as $\alpha = \omega(1)$ [16]. So, it will be good to see if such improvements are possible for other similarity measurements between sequences such as dynamic time warping.

REFERENCES

[1] P. K. Agarwal, K. Fox, J. Pan, and R. Ying, "Approximating dynamic time warping and edit distance for a pair of point sequences," in *Proceedings of the 32nd International Symposium on Computational Geometry*, 2016.

[2] A. Stojmirovic and Y.-k. Yu, "Geometric aspects of biological sequence comparison," *Journal of Computational Biology*, vol. 16, no. 4, pp. 579–611, 2009.

[3] L. Chen and R. Ng, "On the marriage of Lp-norms and edit distance," in *Proceedings of the 30th International Conference on Very Large Databases*, 2004, pp. 792–803.

[4] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, 2005, pp. 491–502.

[5] P.-F. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 306–318, 2009.

[6] S. Sankararaman, P. K. Agarwal, T. Mølhave, J. Pan, and A. P. Boedihardjo, "Model-driven matching and segmentation of trajectories," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 234–243.

[7] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[8] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[9] W. J. Masek and M. S. Paterson, "A faster algorithm computing string edit distances," *Journal of Computer and System Sciences*, vol. 20, no. 1, pp. 18–31, 1980.

[10] O. Gold and M. Sharir, "Dynamic time warping and geometric edit distance: Breaking the quadratic barrier," *ACM Transactions on Algorithms*, vol. 14, no. 4, p. 50, 2018.

[11] K. Bringmann, "Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails," in *Proceedings of the IEEE 55th Annual Symposium on Foundations of Computer Science*, 2014, pp. 661–670.

[12] A. Abboud, A. Backurs, and V. V. Williams, "Tight hardness results for LCS and other sequence similarity measures," in *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science*, 2015, pp. 59–78.

[13] K. Bringmann and M. Künnemann, "Quadratic conditional lower bounds for string problems and dynamic time warping," in *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science*, 2015, pp. 79–97.

[14] A. Backurs and P. Indyk, "Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)," in *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, 2015, pp. 51–58.

[15] C. Fan and B. Raichel, "Computing the Fréchet gap distance," in *Proceedings of the 33rd International Symposium on Computational Geometry*, 2017.

[16] K. Bringmann and W. Mulzer, "Approximability of the discrete Fréchet distance," in *Proceedings of the 31st International Symposium on Computational Geometry*, 2015.

[17] B. B. Ali, Y. Masmoudi, and S. Dhouib, "Accurate and fast dynamic time warping approximation using upper bounds," in *Proceedings of the 38th International Conference on Telecommunications and Signal Processing*, 2015, pp. 1–6.

[18] Q. Zhu, G. Batista, T. Rakthanmanon, and E. Keogh, "A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets," in *Proceedings of the 2012 SIAM international conference on data mining*, 2012, pp. 999–1010.

[19] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[20] G. M. Landau, E. W. Myers, and J. P. Schmidt, "Incremental string comparison," *SIAM Journal on Computing*, vol. 27, no. 2, pp. 557–582, 1998.

[21] A. Andoni, R. Krauthgamer, and K. Onak, "Polylogarithmic approximation for edit distance and the asymmetric query complexity," in *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science*, 2010, pp. 377–386.

[22] M. Boroujeni, S. Ehsani, M. Ghodsi, M. HajiAghayi, and S. Seddighin, "Approximating edit distance in truly subquadratic time: Quantum and mapreduce," in *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018, pp. 1170–1189.

[23] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–31, 2001.

[24] E. Ukkonen, "Algorithms for approximate string matching," *Information and Control*, vol. 64, no. 1-3, pp. 100–118, 1985.

[25] S. Har-Peled, *Geometric approximation algorithms*. American Mathematical Soc., 2011, ch. 11, Random Partition via Shifting.

BIOGRAPHICAL SKETCH

Xinyi Li was born in Chengdu, Sichuan Province, China. She completed her bachelor's degree through the joint program of Beijing University of Post and Telecommunication and Queen Mary University of London in 2016. Later, she was admitted to The University of Texas at Dallas and became a computer science master's student there. During Jan.2018 - Aug.2019, she did her thesis with Dr. Kyle Fox in theory. She plans to continue her research in the future by pursuing her PhD degree in the University of Utah.

CURRICULUM VITAE

Xinyi Li

March 15, 2019

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: xinyi.li2@utdallas.edu

EDUCATION HISTORY:

B.E., Beijing University of Post and Telecommunication, Beijing, China, 2016

RESEARCH INTEREST:

Algorithms and Theory;
Computational Geometric Algorithms;
Approximation Algorithms

ACADEMIC RELATED JOBS :

Research Assistant, the University of Texas at Dallas, May. 2018-present
Teaching Assistant for Algorithms Course, the University of Texas at Dallas, Sept. 2017-May. 2018

AWARDS & HONORS:

Honorable Mention for Mathematical Contest in Modeling Certificate of Achievement, 2015
Second Scholarship for students in BUPT, 2012–2016