Approximating the Geometric Edit Distance

Kyle Fox
The University of Texas at Dallas, USA
kyle.fox@utdallas.edu

Xinyi Li
The University of Texas at Dallas, USA
Xinyi.Li2@utdallas.edu

Abstract

Edit distance is a measurement of similarity between two sequences such as strings, point sequences, or polygonal curves. Many matching problems from a variety of areas, such as signal analysis, bioinformatics, etc., need to be solved in a geometric space. Therefore, the geometric edit distance (GED) has been studied. In this paper, we describe the first strictly sublinear approximate near-linear time algorithm for computing the GED of two point sequences in constant dimensional Euclidean space. Specifically, we present a randomized $O(n \log^2 n)$ time $O(\sqrt{n})$-approximation algorithm. Then, we 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.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Computational geometry

Keywords and phrases Geometric edit distance, Approximation, Randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2019.23

Related Version A full version of the paper is available at `https://arxiv.org/abs/1910.00773`.

Acknowledgements The authors would like to thank Anne Driemel and Benjamin Raichel for helpful discussions.

1 Introduction

Ordered sequences are frequently 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. [2]; however, a 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 [19], ERP (Edit distance with Real Penalty) [10], EDR (Edit Distance on Real sequence) [11], TWED (Time Warping Edit Distance) [16] and a matching framework from Swaminathan et al. [18] motivated by computing the similarity of time series and trajectories. See also a survey by Wang et al. [22].

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 $j < j'$.

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

$$\delta(\mathcal{M}) = \sum_{(i,j)\in\mathcal{M}} dist(p_i, q_j) + \rho(\Gamma(\mathcal{M})) \tag{1}$$

where $dist(p, q)$ is the distance between points p and q (i.e. the Euclidean norm), and $\rho(\Gamma(\mathcal{M}))$ is a function of 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. The distance is sensitive to scaling, so, we can only match points pairs that are sufficiently close together based on the current position. 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*.

▶ **Definition 1.** *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

To simplify the presentation of prior work, we assume $n \geq m$. 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 [21]. Assuming k is the GED, we can achieve an $O(nk)$ time algorithm by restricting our attention to the middle k diagonals of the dynamic programming table (see also Ukkonen [20]). There is a slightly subquadratic $O(n^2/\log n)$ time algorithm [17] for the string version, but it appears unlikely we can apply it to the geometric case. Accordingly, Gold and Sharir [12] proposed a different algorithm which can compute GED as well as the closely related dynamic time warping (DTW) distance in $O(n^2 \log\log\log n/\log\log n)$ time in polyhedral metric spaces. Recent papers have shown conditional lower bounds for several sequence distance measures even with some restrictions. In particular, there is no $O(n^{2-\delta})$ time algorithm for any constant $\delta > 0$ for Fréchet distance [5], DTW over a constant size alphabet [1] or restricted to one-dimensional curves [6], and string edit distance on the binary alphabet [4, 6].[1] The latter of the above results implies the same lower bound for GED, even assuming the sequences consist entirely of $0, 1$-points in $\mathbb{R}$.

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, DTW, and string edit distance [2, 3, 7–9, 14]. In particular, Bringmann

[1] The (discrete) Fréchet and DTW distances are defined similarly to GED; however, they use one-to-many correspondences instead of one-to-one matchings, and they disallow the use of gap points. As in GED, DTW aims to minimize the sum of distances between corresponding points, while discrete Fréchet distance aims to minimize the maximum distance over corresponding points.

and Mulzer [7] describe an α-approximation algorithm for the discrete Fréchet distance that runs in time $O(n \log n + n^2/\alpha)$ for any $\alpha \in [1, n]$. Chan and Rahmati [9] improved this running time to $O(n \log n + n^2/\alpha^2)$. Very recently, Kuszmaul [14] provided $O(\alpha)$-approximation algorithms with $O((n^2/\alpha) \operatorname{polylog} n)$ running times for edit distance over arbitrary metric spaces and DTW over well separated tree metrics. Another $O(n^2/\alpha)$ time algorithm with an $O(\alpha)$ approximation factor for *string* edit distance is to run Ukkonen's [20] $O(nk)$ time algorithm letting k be n/α, and unmatch all characters if this algorithm cannot return the optimal matching. Similarly, we can obtain a different $O(\alpha)$-approximation algorithm for GED running in $O(n^2/\alpha)$ time by making use of the $O(nk)$ time exact algorithm mentioned above. There are many other approximation algorithms specialized for the string version of edit distance. In particular, an $O(\sqrt{n})$-approximation algorithm can be acquired easily from an $O(n + k^2)$ time exact algorithm [15]. The current best results include papers with $(\log n)^{O(1/\epsilon)}$ [3] and constant approximation ratios [8] with different running time tradeoffs.

For GED, a simple linear time $O(n)$-approximation algorithm was observed by Agarwal et al. [2]. 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 [2].

Our results

Inspired by the above applications and prior work, we commit to finding a faster approach to approximating GED between general point sequences while also returning the approximate best matching. Here, we give the first near-linear time algorithm to compute GED with a strictly sublinear approximation factor. We then generalize our result to achieve a tradeoff between the running time and approximation factor. Both of these algorithms are Monte Carlo algorithms, returning an approximately best matching with high probability[2]. To simplify our exposition, we assume the points are located in the plane (i.e., $d = 2$), and we assume the input sequences are the same length (i.e., $m = n$). We can easily extend our results to the unbalanced case, and our analysis implies that outside the plane, the running times and approximation ratios increase only by a factor polynomial in d.

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

The intuitive idea behind this algorithm is very simple. We check if the GED is less than each of several geometrically increasing values g, each of which is less than $O(\sqrt{n})$. For each g, we transform the geometric sequences into strings using a randomly shifted grid, and run the $O(n + k^2)$ time exact algorithm for strings [15]. If the GED is less than g, then 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 give the details for this $O(\sqrt{n})$-approximation algorithm in Section 2.

▶ **Theorem 3.** *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 $\alpha \in [1, \sqrt{n}]$.*

[2] 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 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 tradeoff by just directly altering some parameters in our first algorithm. We discuss why in Section 3.1. To overcome these difficulties, we develop a constant-factor approximation algorithm to compute the GED of point sequences obtained by snapping points of the original input sequences to grid cell corners. Our algorithm for these snapped points is based on the exact algorithm for string edit distance [15] but necessarily more complicated to handle geometric distances. So, we first introduce the $O(n + k^2)$ time algorithm for strings in Section 4.1, and then describe our constant approximation algorithm for points in Section 4.2. We note that a key component of the string algorithm and our extension is a fast method for finding maximal length common substrings from a given pair of starting positions in two strings A and B. A similar procedure was needed in the discrete Fréchet distance approximation of Chan and Rahmati [9]. In Section 3, we present the algorithm for Theorem 3 using our approximation algorithm for snapped point sequences as a black box.

2 $O(\sqrt{n})$-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 value 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 of edit distance.
2. Run a modification of the exact string edit distance algorithm of Landau *et al.* [15]. 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. Details on this modified algorithm appear in Section 4.1.[3]

We explain how to transform the point sequences into strings in Section 2.1, and we analyze the approximation factor and running time in Sections 2.2 and 2.3.

For convenience, we refer to the string edit distance algorithm as $SED(S, T, k)$, where S and T are two strings with equal length. This algorithm will return a matching in $O(n + k^2)$ time if the edit distance is at most k. We give an outline of our algorithm as Algorithm 1. Here, c is a sufficiently large constant, and we use lg to denote the logarithm of base 2.

2.1 Transformation by a random grid

As stated above, the transformation technique should partition nearby points into common groups and distant points into different groups. We use a randomly shifted grid to realize this ideal, see [13] for example.

Recall P and Q lie in $\mathbb{R}^2$. We cover the space with a grid. Let the side length of each grid cell be Δ, and let b be a vector chosen uniformly at random from $[0, \Delta]^2$. 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 $p_1 = (x_1, y_1)$, and $p_2 = (x_2, y_2)$ in this space.

[3] Computing this variant of the string edit distance is really us computing the shortest common super-sequence length of the strings rather than the traditional Levenshtein distance, but we stick with "edit distance" for simplicity.

Algorithm 1 $O(\sqrt{n})$-approximation algorithm for GED.

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

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

▶ **Lemma 4.** *We have* $P(id_{\Delta,b}(p_1) \neq id_{\Delta,b}(p_2)) \leq \min\{\frac{|x_1-x_2|+|y_1-y_2|}{\Delta}, 1\}$.

We use this observation in our algorithm and set $\Delta = \frac{g}{\sqrt{n}}$ as each cell's side length.

2.2 Time complexity

We claim the running time for Algorithm 1 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 hash table. The running time for $SED(S, T, 12\sqrt{n} + 2g)$ is $O(n)$ for $g = O(\sqrt{n})$. Summing over the outer loop and inner loop, the overall running time for Algorithm 1 is

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

2.3 Approximation ratio

In this section, we show that Algorithm 1 returns an $O(\sqrt{n})$-approximate matching with high probability.

Notation

For any monotone matching $\mathcal{M}$, we define $C_S(\mathcal{M})$ as the cost of the corresponding edit operations for $\mathcal{M}$ in the string case and $C_G(\mathcal{M})$ to be $\delta(\mathcal{M})$ as defined in (1) for the geometric case (as stated, there is no substitution operation in our modified string case). Let $\mathcal{M}_G^*$ be the optimal matching for geometric edit distance, and $\mathcal{M}_S^*$ be the optimal matching under the string configuration 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 5.** *If* $GED(S, T) \leq g$*, 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* $C_S(\mathcal{M}_S^*) \leq 12\sqrt{n} + 2g$.

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

$$\begin{aligned} C_G(\mathcal{M}_G^*) =& |UM_{\mathcal{M}_G^*}| + \sum_{(i,j)\in OC_{\mathcal{M}_G^*}} dist(p_i,q_j) + \sum_{(i,j)\in OF_{\mathcal{M}_G^*}} dist(p_i,q_j) \\ &+ \sum_{(i,j)\in DC_{\mathcal{M}_G^*}} dist(p_i,q_j) + \sum_{(i,j)\in DF_{\mathcal{M}_G^*}} dist(p_i,q_j). \end{aligned} \tag{2}$$

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 $C_G(\mathcal{M}_G^*) \le g$. Therefore,

$$\begin{aligned} C_S(\mathcal{M}_S^*) &\le C_S(\mathcal{M}_G^{*'}) \\ &= |UM_{\mathcal{M}_G^*}| + 2|DC_{\mathcal{M}_G^*}| + 2|DF_{\mathcal{M}_G^*}| \le g + 2\sqrt{n} + 2|DC_{\mathcal{M}_G^*}| \end{aligned} \tag{3}$$

For any two points p_i, q_j, let $P_D(i,j)$ be the probability that p_i and q_j are assigned into different cells. From Lemma 4, we can infer $P_D(i,j) \le \frac{2dist(p_i,q_j)}{g/\sqrt{n}}$.

Then,

$$\begin{aligned} E(|DC_{\mathcal{M}_G^*}|) &\le \sum_{(i,j)\in\mathcal{M}_G^*} P_D(i,j) \le \sum_{(i,j)\in\mathcal{M}_G^*} \frac{2dist(p_i,q_j)}{g/\sqrt{n}} \\ &\le 2\sqrt{n}. \end{aligned} \tag{4}$$

Therefore,

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

By Markov's inequality,

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

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

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

We conclude the proof of Lemma 5. ◀

According to Lemma 5, 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 $C_G(\mathcal{M}_S^*)$. Again, $UM_{\mathcal{M}}$ is the set of unmatched indices for a matching $\mathcal{M}$. 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 $(i,j) \in \mathcal{M}_S^*$. We have:

$$C_G(\mathcal{M}_S^*) = |UM_{\mathcal{M}_S^*}| + \sum_{(i,j)\in\mathcal{M}_S^*} dist(p_i, q_j) \tag{5}$$

$$\leq 12\sqrt{n} + 2g + n \cdot (\frac{\sqrt{2}g}{\sqrt{n}}) = 12\sqrt{n} + 2g + \sqrt{2}g\sqrt{n}$$

If $GED(P,Q) \leq \sqrt{n}$, then, with high probability, we obtain a matching $\mathcal{M}_S^*$ during the iteration where $g \geq GED(P,Q) \geq \frac{1}{2}g$. 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 2.

3 $O(\alpha)$-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.

3.1 Flaws in $O(\sqrt{n})$-algorithm to achieve tradeoff

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 (5), 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) 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 do grid-snapping that let us ignore the second inequality.

3.2 $O(\alpha)$-algorithm based on grid-snapping

Grid-snapping

Instead of grouping points into different cells as the $O(\sqrt{n})$-approximation algorithm, we snap points to the lower left corners of their respective grid cells. Let $P' =< p'_1, ..., p'_n >$, $Q' =< q'_1, ..., q'_n >$ be the sequences after grid-snapping. We immediately obtain the following observations:

▶ **Observation 1.** *If p_i and q_j are in the same cell, $dist(p'_i, q'_j) = 0$, and $dist(p_i, q_j) \leq \sqrt{2}\Delta < 2\sqrt{2}\Delta$.*

▶ **Observation 2.** *If p_i and q_j are in different cells, $\Delta \leq dist(p'_i, q'_j) \leq dist(p_i, q_j) + 2\sqrt{2}\Delta$.*

We can then obtain our $O(\alpha)$-approximation algorithm by altering the bound in the outer loop and the test procedure of Algorithm 1. See Algorithm 2. Here, $AGED(P', Q', k)$ attempts to Approximate $GED(P', Q')$ given that P' and Q' have their points on the corners of the grid cells. If $GED(P', Q') \leq k$, then it returns an $O(1)$-approximate matching for the edit distance of the point sequences after grid-snapping. Otherwise, it either returns an $O(1)$-approximate matching or it returns false.

Algorithm 2 $O(\alpha)$-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
            Obtain P', Q' by doing grid-snapping to P, Q based on a randomly
             shifted grid
            out := AGED(P', Q', (12√2 + √2)g)
            if out ≠ false then
                return out
            end
        end
    end
    Return the empty matching
end
```

We describe how to implement $AGED(P', Q', k)$ in Section 4.2. The running time of our implementation is $O(n + \frac{k^2}{\Delta})$ where Δ is the cell side length of the grid. We do grid snapping in $O(n)$ time. For each $g = 2^i$, we use cells of side length $\frac{g\alpha}{n}$ and set k to $\left(12\sqrt{2} + 2\right) g$, so the overall running time of our $O(\alpha)$-approximation algorithm is

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

The analysis for the $O(\alpha)$-approximation algorithm is similar to the first algorithm. The major difference is that for any $g \geq GED(P, Q)$, if we compute the cost of the optimal matching for GED under the new point sequences, it will increase to only $\left(12\sqrt{2} + 2\right) g$ with constant probability despite our small choice for the grid cell side length. But as argued above, the small grid cell side length means the optimal matching of the point sequences after grid-snapping does not increase its cost much when returning the snapped points to their original positions. See Appendix A for details.

4 Constant Approximation Algorithm $AGED(P', Q', k)$

Recall that our constant factor approximation algorithm for GED of grid corner points is based on a known $O(n + k^2)$ time exact algorithm for string edit distance [15]. We first describe this exact algorithm for strings, which we refer as $SED(S, T, k)$, in Section 4.1. Then in Section 4.2, we modify this string algorithm to obtain an $O(1)$-approximate matching for edit distance between point sequences P' and Q' assuming the points lie on the corners of grid cells and $GED(P', Q') \leq k$.

4.1 The exact $O(n + k^2)$ string edit distance algorithm

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. 1 (a).

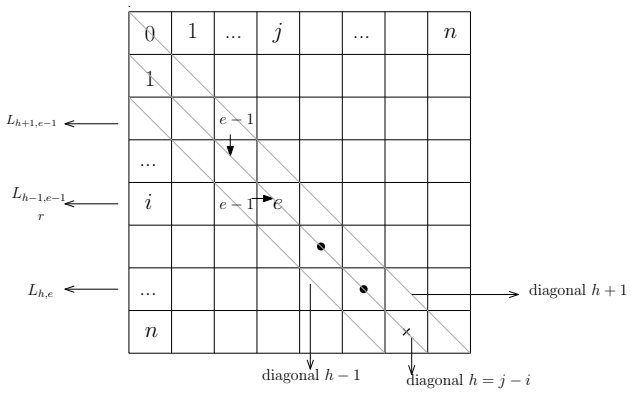


Figure 1 (a) The diagonal containing any entry $(i, i + h)$ is diagonal h. (b) The algorithm slides down the diagonal until finding an entry representing distinct characters. A circle means the corresponding two characters are the same; a cross means they are different.

Recall, we aim to minimize only the number of insertions and deletions to turn S into T. 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 $D(0, j) = j$.

▶ 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 [20]). From Property 4, we know all the diagonals are non-decreasing. In particular, all values on diagonal h are greater than $|h|$ 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.

Algorithm for edit distance at most k

We use a greedy approach to fill the entries along each diagonal. For each value $e \in \{0, \ldots, k\}$ (the outer loop), we locate the elements whose value is e by inspecting diagonals $-e$ 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 the *farthest* e entry in diagonal h.

▶ **Definition 6.** $L_{h,e} = max\{i | D(i, i+h) = e\}$.

Note by Property 3, $L_{h,e}$ is well-defined only if $h \equiv e \mod 2$. Observe that all values on diagonal h are at least $|h|$, which means that we can define our initial values as:

$$L_{h,h-2} = \begin{cases} |h| - 1, & \text{if } h < 0; \\ -1, & \text{otherwise} \end{cases}, \text{ where } h \in [-k, k].$$

Let $r = max\{L_{h-1,e-1}, L_{h+1,e-1} + 1\}$. Then, $D(r, r+h) = e$ by Properties 1 and 4. Also, if $D(r, r+h) = e$ and $s_{r+1} = t_{r+1+h}$, then $D(r+1, r+1+h) = e$. From these observations, we can compute $L_{h,e}$ in each inner loop using Algorithm 3 below.

Algorithm 3 Computing $L_{h,e}$ in each inner loop.

```
1 r := max{L_{h-1,e-1}, L_{h+1,e-1} + 1}
2 while r + 1 ≤ n, r + h + 1 ≤ n, and s_{r+1} == t_{r+1+h} 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 call lines 2 through 4 "the slide". It is straightforward to recover the optimal matching by using the $L_{h,e}$ values to trace backwards through the dynamic programming matrix. Fig. 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. In short, the length of a slide can be computed using a lowest common ancestor query in the suffix tree of a string based on S and T [15]. The overall running time is $O(n + k^2)$.

4.2 $O(1)$-approximation algorithm by modifying the string version

Notation

Similar to the string algorithm, we have a dynamic programming matrix; $D'(i, j)$ is the 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 $dist(p'_i, q'_j)$ 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 need only consider diagonals $-k$ through k.

(Implicit) label rules

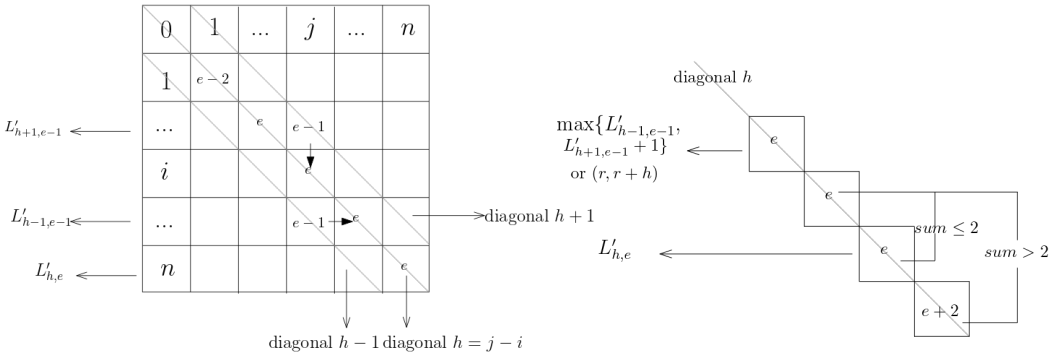
To obtain an approximate matching for the edit distance of snapped point sequences, we now label each entry in the dynamic programming matrix with an approximately tight lower bound on its value. Inspired by the string algorithm, we use non-negative integers for our labels, and the entries of any diagonal h only receive labels e where $e \equiv h \mod 2$. 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 7.** $L'_{h,e} := \max\{i | LA(i, i+h) = e\}$.

For each e from 0 to k, for each diagonal h where $h \equiv e \mod 2$, we (implicitly) assign labels e to each entry on diagonal h.

1. If $h = -e$ or e, i.e., this is the first iteration to assign labels to this diagonal, then we label the very beginning 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 one 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 entries $(r, r+h)$ to $(r+s, r+h+s)$ where $\sum_{i=r+1}^{s} dist(p'_i, q'_{i+h}) \leq 2$ and $\sum_{i=r+1}^{s+1} dist(p'_i, q'_{i+h}) > 2$. $L'_{h,e} = r + s$. 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 an 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. 2 illustrates our rules.



(a) Notations and labels for the boundary entries.

(b) Label entries following step 3.

Figure 2 Notations and rules for approximating SGED.

Computing an approximately optimal matching

Assume we have set the initial values. Our algorithm only needs to compute each $L'_{h,e}$ as before. See Algorithm 4. Then, we guarantee the following theorem:

▶ **Theorem 8.** *We can recover a matching $M_{GS}^{*'}$ using all $L'_{h,e}$ from Algorithm 4. The cost of $M_{GS}^{*'}$ for point sequences P', Q' is less and equal to $3GED(P', Q')$.*

In short, we argue each label $LA(i,j) \leq D'(i,j)$. We then follow a path through the matrix as suggested by the way we pick labels in Algorithm 4. The final matching has cost at most $3LA(n,n)$ which is less and equal to $3GED(P', Q')$. The full proof appears in Appendix B.

Algorithm 4 Computing $L'_{h,e}$ for the fixed h and e.

```
r := max{(L'_{h-1,e-1}), (L'_{h+1,e-1} + 1)}
sum := 0
while r + 1 ≤ n, r + h + 1 ≤ n, and sum + dist(p'_{r+1}, q'_{r+h+1}) ≤ 2 do
    r := r + 1
    sum := sum + dist(p'_r, q'_{r+h})
end
if r > n or r + h > n then
    L'_{h,e} := ∞
else
    L'_{h,e} := r
end
```

We conclude by discussing the time complexity for our algorithm. Using the same $O(n)$ preprocessing as in [15], we can slide down maximal sequences of consecutive entries $(r, r+h)$ with $dist(p'_r, q'_{r+h}) = 0$ in constant time per slide. Let Δ be the cell side length of the grid whose cell corners contain points of P' and Q'. For $dist(p'_r, q'_{r+h}) \neq 0$, we know $dist(p'_r, q'_{r+h}) \geq \Delta$ from Observations 1 and 2. Therefore, we only need to manually add distances and restart faster portions of each slide of distances summing to 2 a total of $\frac{2}{\Delta}$ times. Thus, the total running time is

$$O(n + \sum_{e=0}^{k} \sum_{h=-e}^{e} \frac{1}{\Delta}) = O(n + \frac{k^2}{\Delta}).$$

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78, 2015.
2. Pankaj K Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating dynamic time warping and edit distance for a pair of point sequences. In *Proceedings of the 32nd International Symposium on Computational Geometry*, pages 6:1–6:16, 2016.
3. Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 377–386, 2010.
4. Arturs Backurs and Piotr 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*, pages 51–58, 2015.
5. Karl 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*, pages 661–670, 2014.

6 Karl Bringmann and Marvin 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*, pages 79–97, 2015.

7 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *JoCG*, 7(2):46–76, 2016.

8 Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucky, and Michael Saks. Approximating edit distance within constant factor in truly sub-quadratic time. In *Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 979–990. IEEE, 2018.

9 Timothy M Chan and Zahed Rahmati. An improved approximation algorithm for the discrete Fréchet distance. *Information Processing Letters*, pages 72–74, 2018.

10 Lei Chen and Raymond Ng. On the marriage of Lp-norms and edit distance. In *Proceedings of the 30th International Conference on Very Large Databases*, pages 792–803, 2004.

11 Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 491–502, 2005.

12 Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Transactions on Algorithms*, 14(4):50, 2018.

13 Sariel Har-Peled. *Geometric approximation algorithms*, chapter 11, Random Partition via Shifting, pages 151–162. American Mathematical Soc., 2011.

14 William Kuszmaul. Dynamic Time Warping in Strongly Subquadratic Time: Algorithms for the Low-Distance Regime and Approximate Evaluation. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming*, 2019.

15 Gad M Landau, Eugene W Myers, and Jeanette P Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998.

16 Pierre-François Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):306–318, 2009.

17 William J Masek and Michael S Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

18 Swaminathan Sankararaman, Pankaj K Agarwal, Thomas Mølhave, Jiangwei Pan, and Arnold P Boedihardjo. Model-driven matching and segmentation of trajectories. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 234–243, 2013.

19 Aleksandar Stojmirovic and Yi-kuo Yu. Geometric aspects of biological sequence comparison. *Journal of Computational Biology*, 16(4):579–611, 2009.

20 Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, 1985.

21 Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

22 Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

A Analysis for $O(\alpha)$-approximation algorithm

We introduce some additional notations to those used in Section 2.3.

Let $C_{GS}(\mathcal{M})$ be the cost of any monotone matching $\mathcal{M}$ using distances between the grid-snapped points of P' and Q'. Let $\mathcal{M}^*_{GS}$ be the optimal matching for P' and Q', i.e., $C_{GS}(\mathcal{M}^*_{GS}) = GED(P', Q')$. Let $\mathcal{M}^{*'}_{GS}$ be the matching returned by $AGED(P', Q', (12\sqrt{2}+2)g)$.

We have the following lemma.

▶ **Lemma 9.** *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 $\mathcal{M}^{*'}_{GS}$.*

Proof. Similar to (2), and with Observations 1 and 2, we have

$$\begin{aligned} C_{GS}(\mathcal{M}^*_G) &= |UM_{\mathcal{M}^*_G}| + 0 \cdot (|OC_{\mathcal{M}^*_G}| + |OF_{\mathcal{M}^*_G}|) \\ &\quad + \sum_{(i,j)\in DC_{\mathcal{M}^*_G}} dist(p'_i, q'_j) + \sum_{(i,j)\in DF_{\mathcal{M}^*_G}} dist(p'_i, q'_j) \\ &\leq |UM_{\mathcal{M}^*_G}| + 2\sqrt{2}\Delta \cdot |DC_{\mathcal{M}^*_G}| + \sum_{(i,j)\in DF_{\mathcal{M}^*_G}} \left(dist(p_i, q_j) + 2\sqrt{2}\Delta \right). \\ &= |UM_{M^*_G}| + \sum_{(i,j)\in DF_{\mathcal{M}^*_G}} dist(p_i, q_j) + 2\sqrt{2}\Delta \left(|DC_{M^*_G}| + |DF_{M^*_G}| \right) \end{aligned}$$

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

$$C_{GS}(\mathcal{M}^*_{GS}) \leq C_{GS}(\mathcal{M}^*_G) \leq g + 2\sqrt{2}\Delta \cdot \left(|DC_{\mathcal{M}^*_G}| + |DF_{\mathcal{M}^*_G}| \right).$$

We have the same observation for $DF_{M^*_G}$ as before, that is there are at most $\frac{g}{\Delta}$ pairs in $DF_{M^*_G}$. Using the same algebra as (4), we have $E(|DC_{\mathcal{M}^*_G}|) \leq \frac{2g}{\Delta}$. So,

$$E(C_{GS}(\mathcal{M}^*_{GS})) \leq g + 2\sqrt{2}\Delta \cdot \left(\frac{g}{\Delta} + \frac{2g}{\Delta} \right) = 6\sqrt{2}g + g.$$

According to Markov's inequality, we know

$$P\left(C_{GS}(\mathcal{M}^*_{GS}) \geq \left(12\sqrt{2}+2\right) g \right) \leq \frac{1}{2}.$$

In Section 4.2, we prove that if $C_{GS}(\mathcal{M}^*_{GS}) = GED(P',Q') \leq (12\sqrt{2}+2)g$, then $AGED(P', Q', (12\sqrt{2}+2)g)$ will return a constant approximate matching $\mathcal{M}^{*'}_{GS}$. So, if we test $AGED(P', Q',$
$(12\sqrt{2}+2)g)$ $\lceil c \lg n \rceil$ times (using different grids each time), with a probability at least $1-\frac{1}{n^c}$, at least one $AGED(P', Q', (12\sqrt{2}+2)g)$ will return a matching $\mathcal{M}^{*'}_{GS}$. We conclude the proof of Lemma 9. ◀

Finally, from Observation 2, for every pair (i,j) in M^*_{GS}, we have $dist(p_i, q_j) \leq dist(p'_i, q'_j) + 2\sqrt{2}\Delta$. We can now return points to their original positions:

$$\begin{aligned} C_G(\mathcal{M}^{*'}_{GS}) &= |UM_{\mathcal{M}^{*'}_{GS}}| + \sum_{(i,j)\in DC_{\mathcal{M}^{*'}_{GS}}} dist(p_i, q_j) + \sum_{(i,j)\in DF_{\mathcal{M}^{*'}_{GS}}} dist(p_i, q_j) \\ &\quad + \sum_{(i,j)\in OC_{\mathcal{M}^{*'}_{GS}}} dist(p_i, q_j) + \sum_{(i,j)\in OF_{\mathcal{M}^{*'}_{GS}}} dist(p_i, q_j) \\ &\leq |UM_{\mathcal{M}^{*'}_{GS}}| + \sum_{(i,j)\in DC_{\mathcal{M}^{*'}_{GS}}} dist(p'_i, q'_j) + \sum_{(i,j)\in DF_{\mathcal{M}^{*'}_{GS}}} dist(p'_i, q'_j) + \sum_{(i,j)\in OC_{\mathcal{M}^{*'}_{GS}}} dist(p'_i, q'_j) \\ &\quad + \sum_{(i,j)\in OF_{\mathcal{M}^{*'}_{GS}}} dist(p'_i, q'_j) + 2\sqrt{2}\Delta \left(\left| DC_{\mathcal{M}^{*'}_{GS}} \right| + \left| DF_{\mathcal{M}^{*'}_{GS}} \right| + \left| OC_{\mathcal{M}^{*'}_{GS}} \right| + \left| OF_{\mathcal{M}^{*'}_{GS}} \right| \right) \\ &\leq O(1) \cdot (12\sqrt{2}+2)g + n \cdot 2\sqrt{2}\Delta. \end{aligned}$$

Recall, $\Delta = \frac{g\alpha}{n}$. If we obtain a matching $\mathcal{M}^{*'}_{GS}$ during an iteration where $g \geq C_G(M^*_G) = GED(P,Q) \geq \frac{1}{2}g$, then $C_G(\mathcal{M}^{*'}_{GS}) \leq O(g\alpha) = O(\alpha) \cdot GED(P,Q)$. Using the same argument as in Theorem 2, we conclude our proof of Theorem 3.

B Proof of Theorem 8

We have the following properties for our labels and the following lemma.

▶ 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 $LA(i, i+h) - LA(i, i+h-1) \in \{-1, 1\}$.

▶ **Lemma 10.** *For every entry (i, j), $LA(i, j) \leq D'(i, j)$.*

Note that in particular, $LA(n, n) \leq GED(P', Q')$.

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 10 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. 3 shows the notations. From the refined Property 1, we need to discuss three cases:

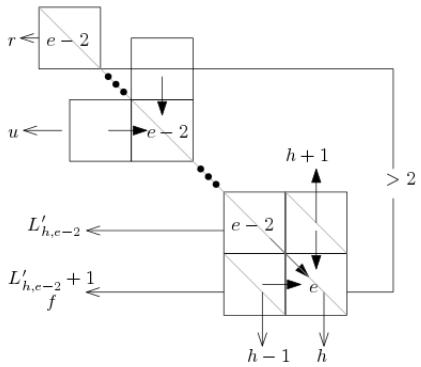


Figure 3 We compute the lower bound of entries which are labeled as e.

   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. $D'(f, f+h) = D'(f-1, f+h-1) + dist(p'_f, q'_{f+h})$.
   $LA(f-1, f+h-1) = e-2$, 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. 3. We define u as the row index of the first entry walking backward from entry $(f, f+h)$ along the 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. 3, 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

  $$\begin{aligned} D'(f, f+h) &= D'(r, r+h) + \sum_{i=r+1}^{f} dist(p'_i, q'_{i+h}) \\ &> e-2+2 = e. \end{aligned}$$

Examining all cases, we conclude the proof of Lemma 10. ◀

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

From Algorithm 4, we note the label increases correspond to not matching a point in Line 1, and slides correspond to matching points. Let $\mathcal{M}^{*'}_{GS}$ be the resulting matching. So,

$$\begin{aligned} C_{GS}(\mathcal{M}^{*'}_{GS}) &= |UM_{\mathcal{M}^{*'}_{GS}}| + \sum_{(i,j)\in\mathcal{M}^{*'}_{GS}} dist(p'_i, q'_j) \\ &\leq LA(n,n) + 2\cdot LA(n,n) \leq 3LA(n,n) \leq 3GED(P', Q'). \end{aligned}$$

We conclude the proof of Theorem 8 and obtain an $O(1)$-approximation algorithm for $GED(P', Q')$.