Exact Algorithms for Partial Curve Matching via the Fréchet Distance

Yusu Wang *

June 17, 2008

Abstract

Curve matching is a fundamental problem that occurs in many applications. In this paper, we study the problem of measuring *partial* similarity between curves. More specifically, given two curves, we wish to maximize the total length of subcurves of them that are *close* to each other, where the *closeness* is measured by the Fréchet distance, a common distance measure for curves. The resulting maximal length is called the *partial Fréchet similarity* between the two input curves. Given two polygonal curves P and Q in $\mathbb{R}^d$ of sizes n and m, respectively, we present the first exact algorithm that runs in polynomial time to compute $\mathcal{F}_\delta(P,Q)$, the partial Fréchet similarity between P and Q, under the L_1 and the L_∞ norms. Specifically, we reduce the problem of computing $\mathcal{F}_\delta(P,Q)$ to $O(nm(n+m))$ instances of a simpler problem, each of which can be solved in $O((n+m)\log(nm))$ time. Hence the partial Fréchet similarity between P and Q under the L_1 or the L_∞ norm can be computed in $O(nm(n+m)^2\log(nm))$ time. To the best of our knowledge, this is the first paper to study this natural definition of partial curve similarity between curves under the continuous setting (where all points in the curve are considered), and present a polynomial-time exact algorithm for it.

*Dept. of Comp. Sci. and Engineering, The Ohio State Univ, Columbus, OH 43016; *yusu@cse.ohio-state.edu*.

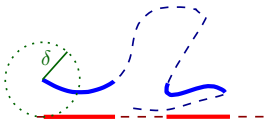
1 Introduction

Measuring similarity between curves is a fundamental problem that appears in many fields, including computer graphics, pattern recognition, geographic information systems, and structural biology. A natural measure of similarity between two curves is the *Fréchet distance*. Intuitively, imagine that a dog and its handler are walking on their respective curves with a leash between them. Both can control their speed, but they can only go forward. The Fréchet distance of these two curves is the minimal length of a leash necessary for the dog and the handler to move from the starting points of the two curves to their respective endpoints. The Fréchet distance takes the inherent order between points along the curves into consideration, making it many times a better measure of similarity for curves than alternatives such as the Hausdorff distance [5, 7].

The Fréchet distance and its variants have been widely used in many applications [20, 21, 23, 25]. Alt and Godau [4] presented an algorithm to compute the Fréchet distance between two polygonal curves with n and m vertices, respectively, in $O(nm\log(nm))$ time. Efficient approximation algorithms have been developed for special families of curves [7, 8]. However, so far, no algorithm, exact or approximate, with running time $o(nm)$ has been found for this problem for general curves.

Where the Fréchet distance falls short is when there are outliers or when the two curves share only partial similarity. An interesting recent work in [16] combined time warping to compute an integral (summed) version of the Fréchet distance, which can "smooth out" the impact of some outliers. Properties of different types of summed Fréchet distance were also studied in [11]. However, summed versions do not fully resolve the issue of partial similarity, especially when significant parts of the curves are dissimilar. To this end, we investigate the *partial curve matching problem*, where given two curves and a threshold δ, we wish to find the best matching such that the largest possible fraction of the two curves are matched within (Fréchet) distance δ.

Partial similarity between curves has previously been measured by patial similarity between their vertices, using the popular Hausdorff distance [12, 19], or (variants of) the RMSD (root-mean-square-deviation) distance. The latter is widely used in communities such as structural biology [22], which treats a curve as an ordered set of points. Since only vertices are considered in such *discrete* measures, they may sometimes fail to reflect similarity. For example, two curves in right figure are similar, but have large distance between their vertices (due to the long edges). In this paper, we aim to develop a *continuous* partial similarity measure, where every point in the polygonal curve is considered.



Alt and Godau [4] considered one natural continuous partial similarity measure by computing the Fréchet distance between *a single consecutive piece* of subcurve of P and another curve Q, and presented an efficient $O(nm\log(nm))$ algorithm to solve the decision-version problem, given an error threshold δ. However, their measure only allows to have outliers in one of the input curve, and more importantly, it does not allow outliers appearring in different (non-consecutive) locations along the input curve. In many applications, such as in structural biology, proteins may be similar only in several key locations, potentially corresponding to functional sites. Hence it is important to capture *all* pieces of matching subcurves. See the left figure for an example where we wish to capture both pairs of matching subcurves (thick subcurves in the figure). We aim to develop a more general partial similarity measure for curves.

Other related work includes minimizing the Fréchet distance under various classes of transformations [6, 14, 27], and extending it to graphs [3, 9], to piecewise smooth curves [26], to simple polygons [10], to surfaces [2], and to more general metric spaces [13, 15]. It was also used in high-dimensional approximate nearest-neighbor search [18], curve simplification [1] and morphing [17].

Our results. Currently, continuous partial curve similarity, though important, has not been extensively studied. Previous measures still have limitations in handling outliers. In this paper, we introduce a natural continuous partial curve similarity measure that allow general types of outliers, and develop an exact algorithm to compute it. Specifically, given a distance threshold δ, let $\mathcal{F}_\delta(P,Q)$ denote the partial Fréchet similarity between two polygonal curves P and Q, which is the total length of longest subcurves of P and Q that are matched with Fréchet distance at most δ (a formal definition will be introduced shortly). The Fréchet distance can be measured under any L_p norm, and we consider the L_1 and L_∞ norms in this paper.

It turns out that the partial Fréchet similarity can be considered as the length of the longest monotone path in a certain polygonal domain with weighted regions, where the weight is either 0 or 1. Hence the computation of $\mathcal{F}_\delta(P,Q)$ bears similarity with the standard shortest path queries in weighted regions. However, the properties of this domain and the monotonicity requirement of the paths considered provide more structure to the problem which our algorithm will exploit. We describe the domain and these longest paths in more detail in Section 3.

In Section 4, we present an exact algorithm to compute $\mathcal{F}_\delta(P,Q)$. Based on the properties of longest monotone paths, we show that computing $\mathcal{F}_\delta(P,Q)$ can be reduced to $O(nm(n+m))$ instances of a simpler optimization problem $\mathsf{OptpathESeq}(s,t,S)$. Each $\mathsf{OptpathESeq}(s,t,S)$ query returns the best path from a source point s to a destination point t passing through edges in S in order. Our reduction framework circumvents the typically harder question of advancing a "wavefront" of optimal path as in the continuous Dijkstra paradigm, or of constructing "shortest path map" type decomposition [24]. Instead, we maintain a superset of edge sequences that optimal paths may visit in order. This, combined with an efficient near-linear time algorithm to answer $\mathsf{OptpathESeq}(s,t,S)$ queries, produces an overall $O(nm(n+m)^2\log(nm))$ time algorithm.

2 Problem Definition and Preliminaries

A curve in $\mathbb{R}^d$ can be represented as a function $\mathsf{f}\colon [0,1]\to\mathbb{R}^d$. A (monotone) *reparametrization* α is a continuous non-decreasing function $\alpha\colon [0,1]\to[0,1]$ with $\alpha(0)=0$ and $\alpha(1)=1$. A *matching* between f and g is simply a pair of monotone reparametrizations (α,β) of f and g respectively, where the point $\mathsf{f}(\alpha(x))$ is matched to the point $\mathsf{g}(\beta(x))$, for any $x\in[0,1]$. Given two curves $\mathsf{f},\mathsf{g}:[0,1]\to\mathbb{R}^d$, the *Fréchet distance* between them under the L_p norm, $\mathcal{D}(\mathsf{f},\mathsf{g})$, is defined as

$$\mathcal{D}(\mathsf{f},\mathsf{g}) := \inf_{\alpha,\beta}\max_{t\in[0,1]} \mathrm{d_p}\big(\mathsf{f}(\alpha(\mathrm{t})),\mathsf{g}(\beta(\mathrm{t}))\big),$$

where $\mathrm{d_p}(\mathrm{x},\mathrm{y})$ denotes the distance between points x and y under the L_p norm, and α and β range over all monotone reparametrizations.

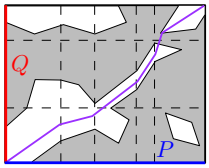
Given a distance threshold $\delta>0$ and a matching (α,β) of curves f and g, the *score* of (α,β) under the L_p norm, also referred to as the *partial similarity* between f and g for a matching (α,β), is defined as

$$\mathcal{S}_{\alpha,\beta}(\mathsf{f},\mathsf{g}) = \int_{\mathrm{d_p}(\mathsf{f}(\alpha(\mathrm{t})),\mathsf{g}(\beta(\mathrm{t}))\le\delta} (||(\mathsf{f}\circ\alpha)'(t))|| + ||(\mathsf{g}\circ\beta)'(t))||)dt,$$

where $||v||$ is the L_2 norm of a vector v; namely, the score is the total length of the portions of the two curves f and g that are matched with L_p distance smaller than δ. Naturally, we would like to maximize the length of these portions, and the *partial Fréchet similarity* between f and g for a threshold δ is defined as the maximum score of any matching of f and g; that is, $\mathcal{F}_\delta(\mathsf{f},\mathsf{g}) := \max_{\alpha,\beta}\mathcal{S}_{\alpha,\beta}(\mathsf{f},\mathsf{g})$ where α and β range over all monotone reparametrizations. Note that for curves in $\mathbb{R}^d$, the partial Fréchet similarity between P and Q under the L_1 (resp. L_∞) norm w.r.t. a

distance threshold δ is at least the optimal partial Fréchet similarity between P and Q under the L_2 norm w.r.t. a distance threshold $\delta/\sqrt{d}$ (resp. $\delta\sqrt{d}$). In this sense, the partial Fréchet similarity under the L_1 or L_∞ approximates that under the L_2 norm.

Free space diagram. For two polygonal curves $P = \langle p_1, \ldots, p_n \rangle$ and $Q = \langle q_1, \ldots, q_m \rangle$, an alternative way to view the partial Fréchet similarity $\mathcal{F}_\delta(P, Q)$ is via the following *free space diagram* $D = D_\delta(P, Q)$ originally introduced in [4]: D is an n by m diagram such that its ith column corresponds to the ith edge of P and has width $||p_i p_{i+1}||$, while its jth row corresponds to the jth edge of Q and has width $||q_j q_{j+1}||$. For a set of segments X in the plane, let $\text{len}_p(X)$ denote the total length of the segments of X under the L_p norm. Let $\mathsf{f} : [0, \text{len}_2(P)] \to \mathbb{R}^d$ and $\mathsf{g} : [0, \text{len}_2(Q)] \to \mathbb{R}^d$ represent the arc-length parametrization of P and Q, respectively. Then every point (x, y) in D corresponds to a pair of points $\mathsf{f}(x) \in P$ and $\mathsf{g}(y) \in Q$. From now on, we abuse the notation slightly and use $P(x)$ and $Q(y)$ to represent $\mathsf{f}(x)$ and $\mathsf{g}(y)$, respectively.

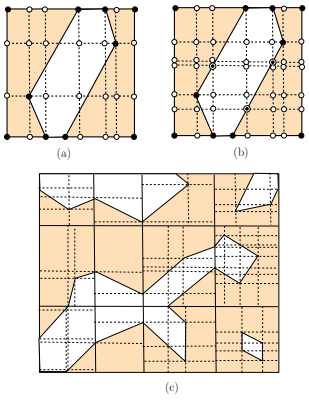


A point $(x, y) \in D$ is *white* if $\mathrm{d_p}(\mathrm{P(x)}, \mathrm{Q(y)}) \leq \delta$. By convexity of the L_p norm, white points within any cell $D[i][j]$ of D form a connected convex region. Under the L_2 norm, this *white region* is the intersection between an ellipse with the cell $D[i][j]$. For curves in $\mathbb{R}^d$, the white region is the intersection between the cell $D[i][j]$ with a convex polygon of complexity $O(d)$ under the L_∞ norm, and with a convex polygon of complexity $O(2^d)$ under the L_1 norm. The boundary edges of the white regions are referred to as *free space edges*. In this paper we consider only the L_∞ and the L_1 norms. Thus the white region in each cell is polygonal, and has constant complexity.

Finally, a *monotone path* in D is a path monotone in both the horizontal and the vertical direction. There is a one-to-one correspondence between all possible matchings of P and Q, and the set of monotone paths in D from its bottom-left corner to its top-right corner. Given any monotone path π, let π_W denote its intersection with the white regions of D. Then $\mathcal{S}_\pi(P, Q)$, the partial similarity between P and Q induced by π, also referred to as the *score of* π, is simply the L_1-norm length $\text{len}_1(\pi_W)$ of π_W. To compute $\mathcal{F}_\delta(P, Q)$, the goal is to find an *optimal monotone path* whose score is maximized. This corresponds to an *optimal partial matching* between P and Q.

3 Monotone Paths and Edge Sequences

In this section, we present some study of the monotone paths in the free space diagram.



Refined free space diagram $\widehat{D}$. Consider a cell $C = D[i][j]$ of the free space diagram D. For every vertex v of the white region in C, we extend a horizontal as well as a vertical line from v till it reaches the boundaries of C. See (a) in the right figure, where dotted segments are these *extension segments*. Next, if a *vertical* extension edge intersects a free space edge, then we extend another horizontal extension segment from the intersection point (the double-points in (b) in the right figure). We process every cell of D by this two-level extension. The resulting diagram is called the *refined free space diagram* $\widehat{D}$; see (c) in the right figure for an example. Let $\text{Arr}(\widehat{D})$ denote the arrangement of $\widehat{D}$; each region in $\text{Arr}(\widehat{D})$ is convex. Since the complexity of the white space within each cell of D is $O(1)$, the complexity of $\text{Arr}(\widehat{D})$ is still $O(nm)$.

Let G be the set of grid edges from the original diagram D as well as the extension segments from the refined diagram $\widehat{D}$. Let $\text{Arr}(G)$ be the arrangement of G — each cell in $\text{Arr}(G)$ is a rectangle.

By construction of $\widehat{D}$, every cell $C \in \mathrm{Arr}(G)$ can contain at most two free space edges that have either *both* positive or *both* negative slopes, and these edges cannot intersect the vertical boundary edges of C. The only eight possible configurations of $C \in \mathrm{Arr}(G)$ are enumerated in Figure 1.

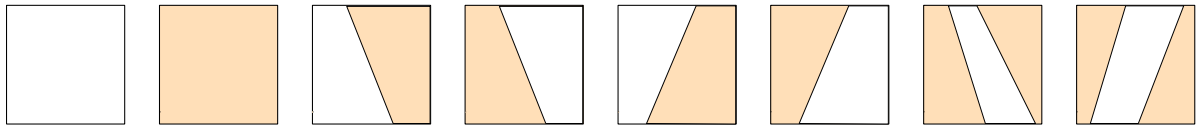
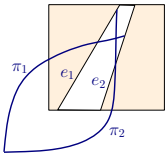
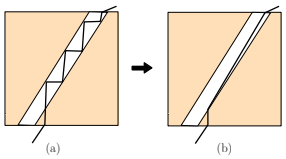


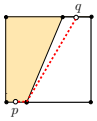
Figure 1: Eight possible configurations of a cell in $\mathrm{Arr}(G)$.



For two (rectangular) cells C_1 and C_2 from $\mathrm{Arr}(G)$, we say that $C_1 \prec C_2$ (or $C_2 \succ C_1$) if there is a monotone path visits C_1 before C_2. Easy to verify that this order relation is well-defined for C_1 and C_2. That is, either it does not exists (no monotone path connects them). Or if it exists, then there cannot exist two monotone paths π_1 and π_2, such that π_1 visits C_1 before C_2, while π_2 visits C_2 before C_1. Hence it induces a partial order over cells from $\mathrm{Arr}(G)$. One can then find a total order of cells in $\mathrm{Arr}(G)$ that is consistent with all these pairwise partial order constraints. We remark that this ordering relationship may not be well-defined for two edges in $\mathrm{Arr}(\widehat{D})$. See the right figure for an example, where path π_1 intersects e_1 and e_2 in order, while another monotone path π_2 traverses e_2 first and then e_1.



Conformal monotone paths. A path is *conformal* if it is monotone, polygonal with vertices lying on edges of $\mathrm{Arr}(\widehat{D})$, and its intersection with any region in $\mathrm{Arr}(\widehat{D})$ has at most one segment. Since the score of a path is the L_1-norm length of its intersection with the white regions, and any region in the arrangement of $\widehat{D}$ is convex, any monotone path can be converted into a conformal path without decreasing its score. Hence from now on, we consider only conformal paths. In general, a monotone path can visit the same edge in $\mathrm{Arr}(G)$ several times (see the thick path in (a) of the right figure). However, a conformal one can visit each edge at most once (see (b) in the right figure).



We now describe a simple but useful observation. Consider a cell $C \in \mathrm{Arr}(G)$. Let p be a point on the left or bottom edge of C, and q a point on the top or right edge. Then one can connect p and q optimally (i.e, with largest score) by a monotone polygonal chain with at most three pieces, which we refer to as a *three-piece configuration*. Specifically, the first (resp. last) segment will be degenerate if and only if p (resp. q) is inside the white region. The middle segment is contained in the white region. The configuration of the polychain is such that the L_1-norm length of the middle segment is maximized. See the right figure for an example where the last piece is degenerate. The optimal three-piece configuration can be computed in constant time once the entry and point points are given.

Path-score function. Now consider $\pi^*(\mathbf{b}, p)$, an optimal path from the left-bottom vertex of $\widehat{D}$, $\mathbf{b}$, to a point $p \in \widehat{D}$. Define *the path-score at* p as the score of this optimal path; that is, $\mathcal{S}_{\mathbf{b}}(p) = \mathcal{S}(\pi^*(\mathbf{b}, p))$. Althought the path-score function $\mathcal{S}_{\mathbf{b}} : \widehat{D} \to \mathbb{R}$ measures the longest length, under the L_1-norm and the monotonicity constraint, it can be converted to the shortest monotone path problem over a weighted domain. Indeed, as observed by several reserachers, since any monotone path from s to t has the same L_1-length, maximizing the L_1-length inside white

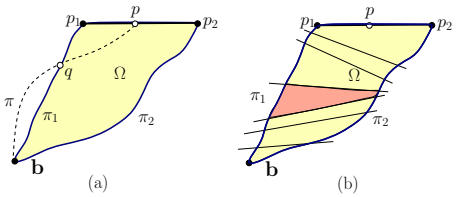
regions is the same as minimizing the L_1-length inside black regions. Hence finding an optimal path from $\mathbf{b}$ to p is the same as finding the shortest L_1-norm length monotone path from $\mathbf{b}$ to p where the white region has weight 0, and the black region has weight 1.

Observation 3.1 *The function value of $\mathbb{S}_{\mathbf{b}}$ along any horizontal (resp. vertical) edge is non-decreasing from left to right (resp. bottom to top).*

Edge-sequences. We say that a path π belongs to the class $\mathbb{C}(S)$ if S is the sequence of edges from $\text{Arr}(\widehat{D})$ visited by π in order. An edge-sequence S is *optimal at p* if there is a path $\pi \in \mathbb{C}(S)$ from the bottom-left vertex $\mathbf{b}$ of D to p, that produces the optimal score at p. We also say that S *can reach p optimally* in this case. Note that due to monotone and conformal constraints, any conformal path can visit an edge e at most once. Hence in this paper, for any of the edge sequences we consider, no edge appears more than once in it.

Claim 3.2 *Along each vertical or horizontal edge e, the region that any edge-sequence S can reach optimally is a single segment (possibly empty).*

Proof: Assume without loss of generality that e is horizontal. Suppose S is optimal at two points $p_1, p_2 \in e$. We claim that any point p in the segment p_1p_2 can be optimally reached by S as well.



Indeed, let π_1 and π_2 be two optimal paths in $\mathbb{C}(S)$ connecting the bottom-left vertex $\mathbf{b}$ of D to p_1 and p_2, respectively. Consider the region Ω bounded by π_1, π_2, the segment p_1p_2, and the vertex $\mathbf{b}$. Since π_1 and π_2 visit the same edge-sequence in order, there cannot be any vertex in the interior of Ω. In fact, any conformal monotone path completely inside Ω will have to follow the edge sequence S. To see this, imagine that the region Ω is decomposed into a set of chambers by edges in S (see (b) in the right figure). Since a conformal path intersects each edge at most once, within any chamber, the monotone path has only one choice where to leave. Hence it will follow the edge sequence S. This implies that if there is a point p in the interior of p_1p_2 such that p cannot be reached by S, then any optimal path π at p cannot lie completely inside Ω. Furthermore, this optimal path π must intersect either π_1 or π_2 — otherwise, π has to reach p from above the horizontal edge e, and thus cannot be monotone.

Assume without loss of generality that π intersects π_1 (see the dashed curve in (a) of the previous figure). Let q be the last intersection point. Obviously, the subcurves $\pi_1[\mathbf{b}, q]$ and $\pi[\mathbf{b}, q]$ are both optimal at q and have the same score. Now construct a new path π', which is the concatenation between $\pi_1[\mathbf{b}, q]$ and $\pi[q, p]$. Since both π and π_1 are conformal, π' is conformal everywhere other than around q, as q may be in the interior of some region $R \in \text{Arr}(\widehat{D})$. However, since R is convex, we can easy modify π' locally to make it conformal, by connecting the two intersection points of π' and R directly. The resulting path π'' is conformal and it has the same score as π'. Furthermore, since π'' now lies completely inside Ω, $\pi'' \in \mathbb{C}(S)$. This causes contradiction as π'' reaches S optimally. Hence our assumption is wrong and the lemma holds. ■

We refer to this segment on e that can be optimally reached by S as the *influence-inteval* $\mathbf{I}(S, e)$; e is omitted when its choice is clear. We say that S is *valid* on e if $\mathbf{I}(S, e)$ is not empty. (We remark that the above claim does not hold for a slope edge. An example is described in Appendix A. This implies that the region in $\widehat{D}$ that an edge-sequence can reach optimally is not convex. However, its intersections with horizontal and vertical edges are.) The proof of the following observation is similar to that of Claim 3.2 and omitted.

Observation 3.3 *Given a horizontal or vertical edge e,let S_1 and S_2 be two edge-sequences with non-empty influence-intevals $\mathbf{I}(S_1)$ and $\mathbf{I}(S_2)$ on e. It cannot happen thatone of the influence-intevals lie completely in the interior of the other.*

4 Exact Algorithm for Partial Fréchet Similarity

Let E be the set of edges in $\mathrm{Arr}(\widehat{D})$. Below we first describe a framework that computes the partial Fréchet similarity using the following subroutine: given an edge sequence $S \subseteq E$, $\mathsf{OptpathESeq}(p, q, S)$ returns the best path in $\mathbb{C}(S)$ from the point p to the point q. We then show how to implement the $\mathsf{OptpathESeq}()$ procedure in near linear time, thereby producing an efficient polynomial time algorithm to compute the partial Fréchet similarity between two polygonal curves exactly.

Overview of the algorithm. Our goal is to compute some longest monotone path in a weighted region (weight 1 for white regions, and 0 otherwise). Unfortunately, it appears that there is no simple characterization of these optimal paths so that one only needs to check polynomial number of discrete cases. For example, due to the monotonicity requirement, it is possible that any optimal path to a certain point has to pass through *interior points* of edges in $\widehat{D}$, and straightforward characterization of such interior points leads only to exponential number discrete cases. Our algorithm instead computes a set of edge-sequences that may produce an optimal path.

More specifically, consider a total order $\langle C_1, \ldots, C_M \rangle$ of the set of rectangular cells in $\mathrm{Arr}(G)$ where if $C_i \prec C_j$, then $i < j$. We process C_is in increasing order of i. Since the influence-inteval of an edge-sequence is well-behaved along a horizontal or vertical edge (Claim 3.2), we maintain for each *boundary grid edge* e of a cell C_i, a *candidate-set of edge sequences* such that the union of their influence-intevals covers the edge e. Note that we *do not* maintain the influence-intevals of these edge-sequences. This candidate-set does not necessarily contain all valid edge-sequences on e (the number of which can be exponential in worst case), and it may contain invalid edge-sequence whose influence-inteval on e is empty. The important observation that we will show shortly is that such a candidate-set of size $O(nm)$ can be maintained for each cell by making only a polynomial number of $\mathsf{OptpathESeq}()$ queries. Finally, after we compute the candidate-set Σ associated with the two grid edges incident to the top-right vertex $\mathbf{t}$ of D, we can identify an optimal path from $\mathbf{b}$ to $\mathbf{t}$ by checking the best path generated by each edge-sequence in Σ.

In what follows, we describe in Section 4.1 how we maintain the candidate-set for each cell in Π, and show how to answer $\mathsf{OptpathESeq}()$ queries in near linear time in Section 4.2. We summarize and analyze the entire algorithm in Section 4.3.

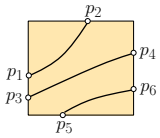
4.1 Update of Candidate-sets

Given a horizontal or vertical edge e, we assign a direction to it: *left-right* if e is horizontal and *top-down* if e is vertical. This direction induces an *ordering* between points in e. For example, $p \prec q$ if p is to the left of q or if p is above q. An interval on e can be represented by its endpoints: $\langle$ *left-endpoint, right-endpoint* $\rangle$ if e is horizontal, and $\langle$ *top-endpoint, bottom-endpoint* $\rangle$ if e is vertical. Given two non-empty intervals I_1 and I_2 on e, the *order* between I_1 and I_2, denoted by $I_1 \prec I_2$ or $I_2 \succ I_2$, is the lexicographic order between their corresponding pair-representations. Easy to verify that this definition of order induces a partial order relation over a set of intervals on e. Given two edge-sequences S_1 and S_2, if their influence-intevals $I_1 = \mathbf{I}(S_1)$ and $I_2 = \mathbf{I}(S_2)$ on e are non-empty, then $S_1 \prec S_2$ (or $S_1 \succ S_2$) if $I_1 \prec I_2$ (or $I_1 \succ I_2$); otherwise, there is no relationship between S_1 and S_2. Note that by Observation 3.3, if $I_1 \prec I_2$, then the right-endpoint (resp. bottom-endpoint) of I_1 cannot lie to the right of (resp. below) that of I_2.

Given a set of edge-sequences L, we say that L *covers* an edge e if $\bigcup_{S\in L}\mathbf{I}(S) = e$; that is, the union of the influence-intevals induced by L covers e. Our algorithm maintains for each grid edge $e \in G$ a *candidate-set, denoted by* $\mathbf{L}(e)$, which is an ordered list of edge-sequences that covers e, and whose order is compatible with the partial order constraints between edge-sequences as introduced above. The following observation is straightforward, as one can simply perform the $\mathsf{OptpathESeq}(\mathbf{b}, p, S)$ query for each edge-sequence $S \in \mathbf{L}(e)$.

Observation 4.1 *Given any edge e and the candidate-set $\mathbf{L}(e)$ associated with it, for any point $p \in e$, an optimal path $\pi^*(\mathbf{b}, p)$ can be computed by $O(|\mathbf{L}(e)|)$ number of* $\mathsf{OptpathESeq}()$ *queries.*

Now consider any rectangular cell $C \in \mathrm{Arr}(G)$. Recall that there are eight possible configurations for C, as illustrated in Figure 1. Let e_L, e_B, e_T and e_R denote the left, bottom, top, and right boundary edges of C, respectively. Since we process all cells in $\mathrm{Arr}(G)$ in order, at the time we process C, we already have the candidate-sets for e_L and e_B, and we wish to maintain such lists for e_T and e_R.



Let $\mathbf{L}_1 \oplus \mathbf{L}_2$ denote the concatenation of an ordered list $\mathbf{L}_2$ to the end of $\mathbf{L}_1$. Set $\mathbf{L}_L = \mathbf{L}(e_L) \oplus \mathbf{L}(e_B)$ and $\mathbf{L}_U = \mathbf{L}(e_T) \oplus \mathbf{L}(e_R)$. We say that an edge sequence S_2 is *an extension* of another one S_1 if S_1 is a prefix of S_2. Let X be a list of edge-sequences. Denote by $\mathrm{ExtS}(X, e)$ the extensions of edge-sequences of X so that e is the last edge, and for a set of edges E', denote by $\mathrm{ExtS}(X, E')$ the union of edge-sequences $\bigcup_{e'\in E'} \mathrm{ExtS}(X, e')$. To compute $\mathbf{L}_U$ from $\mathbf{L}_L$, we need the following results. Intuively, $\mathbf{L}_U$ can be computed by extending edge-sequences from $\mathbf{L}_L$, and this extension will roughly "preserves" the order of edge-sequences in $\mathbf{L}_L$ (see the right figure illustrating the intuition, where p_2, p_4 and p_6 are extended by optimal paths from p_1, p_3 and p_5). We use $e_L \oplus e_B$ (resp. $e_T \oplus e_R$) to denote the concatenation of e_L and e_B (resp. e_T and e_R). Points in e_L (resp. e_T) preceed points in e_B (resp. e_R).

Lemma 4.2 *Let $N = |\mathbf{L}_L|$ denote the size of $\mathbf{L}_L$.*
(1) Each edge-sequence in $\mathbf{L}_L$ can generate $O(1)$ extensions to $e_T \cup e_R$. Furthermore, the length of any extension of $S \in \mathbf{L}_L$ to $e_T \cup e_R$ is $|S| + O(1)$.
(2) For any point $q \in e_T \cup e_R$, there is an edge-sequence $S \in \mathrm{ExtS}(\mathbf{L}_L, \{e_T, e_R\})$ such that S is optimal at q.
(3) Any point $p \in e_L \cup e_B$ breaks the chain $e_L \oplus e_B$ into two portions, s_1 and s_2. Let $\mathbf{L}_L[m]$ be any edge-sequence from $\mathbf{L}_L$ that is optimal at p. Then $\{\mathbf{L}_L[1], \ldots, \mathbf{L}_L[m]\}$ covers s_1, and $\{\mathbf{L}_L[m], \ldots, \mathbf{L}_L[N]\}$ covers s_2.
(4) Any point $q \in e_T \cup e_R$ breaks the chain $e_T \oplus e_R$ into two portions, s'_1 and s'_2. Let $\mathbf{L}_L[k]$ be an edge-sequence from $\mathbf{L}_L$ such that some extension of it is optimal at q. Then $\mathrm{ExtS}(\{\mathbf{L}_L[1], \ldots, \mathbf{L}_L[k]\}, s'_1)$ covers s'_1, and $\mathrm{ExtS}(\{\mathbf{L}_L[k], \ldots, \mathbf{L}_L[N]\}, s'_2)$ covers s'_2.

Proof: Claim (1) follows immediately from the fact that there are only constant number of edges in the cell C. To show claim (2), take any optimal path π_q from $\mathbf{b}$ to q. Due to the monotonicity requirement, π_q must intersect $e_L \cup e_B$ exactly once, say, at q' in $e_L \cup e_B$. Let $S \in \mathbf{L}_L$ be any edge-sequence from $\mathbf{L}_L$ that is optimal at q' (i.e, $q' \in \mathbf{I}(S)$). Set S' be the extension of S by adding edges traversed by the subcurve $\pi_q[q', q]$ to the end of S. Obviously, S' is optimal at q.

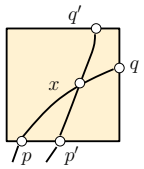
Claim (3) follows from the fact that $\mathbf{L}_L$ covers $e_L \cup e_B$, and that it is ordered. Indeed, for any $p' \in s_1$, let j be the smallest index such that $\mathbf{L}_L[j]$ is optimal at p'. We claim that $j \le m$. This is because if $j > m$, then the facts that $\mathbf{L}_L[m] \prec \mathbf{L}_L[j]$, and $\mathbf{L}_L[m]$ is optimal at $p \succ p'$ imply that the influence-inteval of $\mathbf{L}_L[m]$ contains p'. Contradiction. Similarly, for $p'' \in s_2$, let l be the largest index such that $\mathbf{L}_L[l]$ is optimal at p''. We claim that $l \ge m$. This is because that if $l < m$, then by

Observation 3.3, the influence-inteval of $\mathbf{L}_L[m]$ should contain the right endpoint of that of $\mathbf{L}_L[l]$. Hence $\mathbf{L}_L[m]$ is optimal at p''.

We now prove claim (4). Specifically, we show that $\text{ExtS}(\{\mathbf{L}_L[1], \ldots, \mathbf{L}_L[k]\}, s'_1)$ covers s'_1. Proof for the other half is symmetric (but with the use of Observation 3.3). Assume that there is a point $q' \in s'_1$ such that no extension from $\mathbf{L}[i]$, for $i \in [1, k]$, can be optimal at q'. By claim (1), there must be some $\mathbf{L}[j]$ optimal at q' with $j > k$. Let π (resp. π') be an optimal path to q (resp. to q') whose edge-sequence is an extension of $\mathbf{L}_L[k]$ (resp. $\mathbf{L}_L[j]$), and p (resp. p') is its intersection point with $e_L \cup e_B$. We claim that $p \prec p'$.

Indeed, since $\mathbf{I}(\mathbf{L}_L[k]) \prec \mathbf{I}(\mathbf{L}_L[j])$, the left (or top) endpoint of $\mathbf{L}_L[k]$ must be to the left (or above) that of $\mathbf{L}_L[j]$. Hence if $p \succ p'$, then $p' \in \mathbf{I}(\mathbf{L}_L[k])$. This implies that an optimal path to q' can be generated by extending $\mathbf{L}_L[k]$ as well, which is a contradiction.

Since $q' \prec q$ and $p \prec p'$, this means that π and π' must intersect, say, at point x. See the right figure. Now modify π to π'' by replacing $\pi[x, q]$ by $\pi'[x, q']$. Obviously, $\mathcal{S}(\pi') = \mathcal{S}(\pi'')$, implying that π'' is optimal at q'. Since π'' is generated by an extension of $\mathbf{L}_L[k]$, this is a contradiction. Thus the claim holds. ■

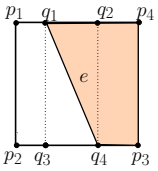


Lemma 4.3 *Given any cell $C \in \text{Arr}(G)$, and the candidate-sets $\mathbf{L}(e_L)$ and $\mathbf{L}(e_B)$, we can compute $\mathbf{L}(e_T)$ and $\mathbf{L}(e_R)$ by making $O(|\mathbf{L}(e_L)| + |\mathbf{L}(e_B)|)$ number of* OptpathESeq*() queries. Furthermore, we have that $|\mathbf{L}(e_T)| + |\mathbf{L}(e_R)| \leq |\mathbf{L}(e_L)| + |\mathbf{L}(e_B)| + O(1)$.*

Proof: By Lemma 4.2 (1), each edge-sequence in $\mathbf{L}_L = \mathbf{L}(e_L) \oplus \mathbf{L}(e_B)$ can have $O(1)$ extensions to edges $e_T \cup e_R$. Hence it is easy to construct a set of edge-sequences $\mathbf{L}_U = \mathbf{L}(e_T) \oplus \mathbf{L}(e_R)$ such that $|\mathbf{L}_U| = O(|\mathbf{L}_L|)$. To reduce its size to $|\mathbf{L}_L| + O(1)$, we will construct $\mathbf{L}_U$ in such a way that any edge-sequence from $\mathbf{L}_L$, other than constant numbers, can have *at most one* extension in $\mathbf{L}_U$. Below we show how to construct $\mathbf{L}_U$ for each configuration in Figure 1.

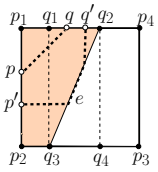
(1) Configuration (a) and (b). By Lemma 4.2 (2), there exists some edge-sequence from $\mathbf{L}_L$, say $\mathbf{L}_L[k]$, such that its extension to the top-right vertex p of C is optimal at p. To compute $\mathbf{L}_L[k]$, we simply perform $\text{OptpathESeq}(\mathbf{b}, p, \mathbf{L}_L[i] \oplus e_T)$ queries for each $i \in [1, N]$, and return the one that produces the longest monotone path to p. We set $\mathbf{L}(e_T)$ as the extensions of edge-sequences $\mathbf{L}_L[1], \ldots, \mathbf{L}_L[k]$ in order, and $\mathbf{L}(e_R)$ as the extensions of $\mathbf{L}_L[k], \ldots, \mathbf{L}_L[N]$. Specifically, since the cell is all white for Configuration (a), and all black for Configuration (b), there is *only one way* to extend an edge-sequence from $\mathbf{L}_L$ to e_T or e_R. That is, $\mathbf{L}(e_T)[i] = \mathbf{L}_L[i] \oplus e_T$ for $1 \leq i \leq k$, and $\mathbf{L}(e_R)[i] = \mathbf{L}_L[i + k - 1]) \oplus e_R$ for $1 \leq i \leq |\mathbf{L}_L| - k + 1$. It follows from Lemma 4.2 (4) that $\mathbf{L}_U$ constructed this way indeed covers $e_T \cup e_R$. Furthermore, since $\mathbf{L}_L[k]$ is the only edge-sequence from $\mathbf{L}_L$ that extends to more than one class in $\mathbf{L}_U$, we have that $|\mathbf{L}_U| = |\mathbf{L}_L| + 1$.

(2) Configuration (c), (d), and (e). We describe only the update for configuration (c). Configurations (d) and (e) can be handled similarly. See the right figure. Let e be the free space edge in the cell C. Partition e_L and e_B into a set of segments $\Pi_1 = \{p_1p_2, p_2q_3, q_3q_4, q_4p_3\}$, and e_T and e_R into $\Pi_2 = \{p_1q_1, q_1q_2, q_2p_4, p_4p_3\}$. Given an entry point $p \in w_1 \in \Pi_1$ and an exit point $q \in w_2 \in \Pi_2$, recall that p and q can be optimally connected by a three-piece configuration, such that the piece in the white region has maximal L_1 length. Straightforward case analysis shows that the combinatorial description of this configuration, that is, the sequence of edges it intersects, can be decided purely by the two segments w_1 and w_2 (regardless of the positions of p and q in these segments), and can be computed in constant time. Hence how to optimally extend an edge-sequence at p to q depends only on w_1 and w_2 as well. For example, let S_1 be an edge-sequence at p, and S_2 its optmial extension to q. If $p \in w_1 = p_2q_3$ and $q \in w_2 = q_1q_2$, then $S_2 = S_1 \oplus e \oplus e_T$; while if $w_1 = q_4p_3$ and $w_2 = p_4p_3$, then $S_2 = S_1 \oplus e_R$.



Now to construct $\mathbf{L}_U$, we first compute edge-sequences $\mathbf{L}_L[m_1], \mathbf{L}_L[m_2]$, and $\mathbf{L}_L[m_3]$ from $\mathbf{L}_L$ that are optimal to p_2, q_3, and q_4, respectively. By Lemma 4.2 (3), $\mathbf{L}_L$ is now decomposed to $\mathbf{L}(p_1p_2) = \{\mathbf{L}_L[1], \ldots, \mathbf{L}_L[m_1]\}$, $\mathbf{L}(p_2q_3) = \{\mathbf{L}_L[m_1], \ldots, \mathbf{L}_L[m_2]\}$, $\mathbf{L}(q_3q_4) = \{\mathbf{L}_L[m_2], \ldots, \mathbf{L}_L[m_3]\}$, and $\mathbf{L}(q_4p_3) = \{\mathbf{L}_L[m_3], \ldots, \mathbf{L}_L[N]\}$, where $N = |\mathbf{L}_L|$. In other words, we now assign each edge-sequence in $\mathbf{L}_L$ a segment from Π_1 (other than $\mathbf{L}_L[m_i]$'s). Next, we identify the edge-sequences $\mathbf{L}_L[k_1]$, $\mathbf{L}_L[k_2]$, and $\mathbf{L}_L[k_3]$ from $\mathbf{L}_L$ whose extensions produce optimal paths reaching q_1, q_2, and p_4, respectively. This can be done by $O(|N|)$ number of OptpathESeq() queries by Lemma 4.2 (1) and Observation 4.1. Finally, $\mathbf{L}(p_1q_1)$, $\mathbf{L}(q_1q_2)$, $\mathbf{L}(q_2p_4)$, and $\mathbf{L}(p_4p_3)$ are simply the proper extensions of edge-sequences in $\langle \mathbf{L}_L[1], \ldots, \mathbf{L}_L[k_1] \rangle$, $\langle \mathbf{L}_L[k_1], \ldots, \mathbf{L}_L[k_2] \rangle$, $\langle \mathbf{L}_L[k_2], \ldots, \mathbf{L}_L[k_3] \rangle$, and $\langle \mathbf{L}_L[k_3], \ldots, \mathbf{L}_L[N] \rangle$, respectively. Based on the identities of the segment from Π_1 associated with each edge-sequence in $\mathbf{L}_L$, and the target segment in Π_2, each extension is uniqutely decided. Hence $\mathbf{L}_U$, the union of these extensions, has size $|\mathbf{L}_U| \leq |\mathbf{L}_L| + 6$, as only $\mathbf{L}_L[m_i]$s and $\mathbf{L}_L[k_j]$s can potentially produce two extensions in $\mathbf{L}_U$.

(3) Configuration (f). See the right figure. Set $\Pi_1 = \{p_1p_2, p_2q_3, q_3q_4, q_4p_3\}$, and $\Pi_2 = \{p_1q_1, q_1q_2, q_2p_4, p_4p_3\}$. Consider extending any edge-sequence optimal to a point in a segment w_1 in Π_1 to a point in a segment w_2 in Π_2. The combinatorial description of this extension can be decided purely based on the choices of w_1 and w_2, for any combination other than *one* particular case: That is, depending on where the entry point $p \in p_1p_2 = e_L$ and exit point $q \in q_1q_2$ are located, the optimal path from p to q *may or may not* intersect the free space edge e, thus the choice for extension is not unique in this case. Two examples (the thick dashed chains from p to q and from p' to q') are shown in the right figure.



Nevertheless, we now show that the best score at each point on $e_1 = q_1q_2$ has a simple structure that helps to identify the extensions from $\mathbf{L}(e_L)$ to e_1. Specifically, for $t \in [0, 1]$, let $F(t)$ denote the best score of any path reaching the point $e_1(t) = (1-t)q_1 + tq_2$ via some point in e_L.

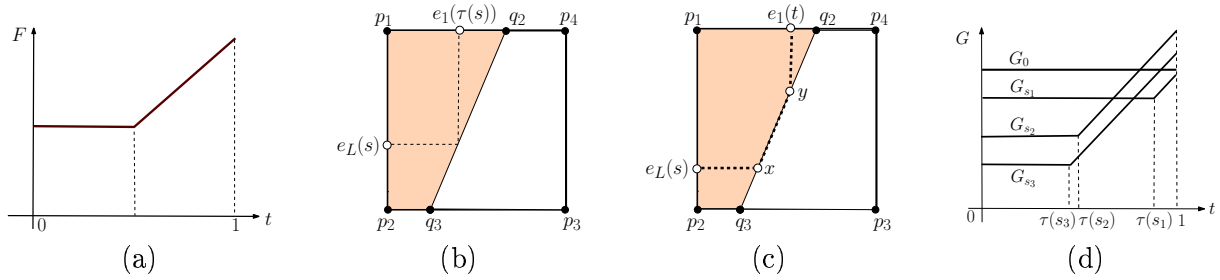


Figure 2: (a) The graph of F has an elbow shape. (b) The definition of $\tau(s)$. (c) The best three-piece configuration to connect $e_L(s)$ to $e_1(t)$ for $t \geq \tau(s)$. (d) The graph of F is the upper envelope of those of G_s, and has an elbow shape. $s_1 < s_2 < s_3$ in the figure.

Claim 4.4 *The function $F(t)$ is a piecewise-linear function with at most two pieces (see Figure 2 (a)). The horizontal piece is induced by some path passing through p_1, and the slope piece is induced by some path passing through a particular point $p^* \in e_L$.*

Proof: Consider the score function $G_s : [0, 1] \to \mathbb{R}$ where $G_s(t)$ is the best score of any path reaching $e_1(t)$ by passing through the point $e_L(s) = (1-s)p_1 + sp_2$. The graph of F is the upper envelope of the graph of G_s for all $s \in [0, 1]$. That is, $F(t) = \max_{s \in [0,1]} G_s(t)$ for any $t \in [0, 1]$.

Now shoot a ray rightward from $e_L(s)$, and change its direction upward once the ray touches the free space edge e. Denote by $e_1(\tau(s))$ the intersection between the vertical ray and e_1. See

Figure 2 (b). Obviously, the function $\tau : [0,1] \to [0,1]$ is a non-increasing function with $\tau(0) = 1$ and $\tau(1) = 0$. For $t < \tau(s)$, $G_s(t)$ is simply the best score at $e_L(s)$, i.e, $G_s(t) = \mathcal{S}_{\mathbf{b}}(e_L(s))$. For $t \geq \tau(s)$, an optimal way to connect $e_L(s)$ to $e_1(t)$ has the configuration as shown in Figure 2 (c). Indeed, the segment xy has the longest L_1-norm length among the intersection between any monotone path from $e_L(s)$ to $e_1(t)$ and the white region. Hence $G_s(t)$ is a linear function for $\tau(s) \leq t \leq 1$. Furthermore, the slope ρ of G_s depends only on the slope of the free space edge e and the ratio between the length of e_L and e_1, so it remains the same for any $s \in [0,1]$.

In other words, the graph of G_s has an *elbow-shape* with two pieces: the horizontal piece is at the height of $\mathcal{S}_{\mathbf{b}}(e_L(s))$, and the slope piece starts from $\tau(s)$ and has a fixed slope ρ independent of s. Furthermore, by Observation 3.1, $\mathcal{S}_{\mathbf{b}}(e_L(s))$ is a non-increasing function as s increases. Hence the starting height of the graph of G_s is non-increasing, and all below $\mathcal{S}_{\mathbf{b}}(e_1(0)) = \mathcal{S}_{\mathbf{b}}(e_L(0))$. Now consider the s^* such that $G_{s^*}(1)$ is highest. Easy to verify that the graph of F is the upper envelope of G_0 and G_{s^*}. See Figure 2 (d), where $s^* = s_2$. The lemma then follows. The horizontal piece of F is induced by G_0, which is a contant function at height $\mathcal{S}_{\mathbf{b}}(p_1)$. ■

By the claim above, in order to compute an edge-sequence S^* that generates the slope piece of $F(t)$, we simply check the edge-sequence $S \oplus e \oplus e_T$ for each $S \in \mathbf{L}(e_L)$ and find the one producing the largest score at q_2 (i.e, $G_{s^*}(1)$ is highest). The overall computation of $\mathbf{L}_U$ is similar to the case for Configurations (c), (d), and (e). The main difference is that when considering the extensions from $\mathbf{L}(e_L)$ to $\mathbf{L}(p_1q_2)$, we can ignore all edge-sequences in $\mathbf{L}(e_L)$ other than S^* and the one that is optimal at p_1, say S_1. We only keep two extensions $S_1 \oplus e_T$ and $S^* \oplus e \oplus e_T$ for this case. The lemma holds for this case, as again only constant number of edge-sequences from $\mathbf{L}_L$ will have two extensions in $\mathbf{L}_U$.

Finally, the update for configuration (g) is a combination of those of configurations (d) and (c), and that for configuration (h) is a combination of cases (f) and (e). In summary, we can update $\mathbf{L}_L$ to obtain $\mathbf{L}_U$ by making $O(|\mathbf{L}_L|)$ number of OptpathESeq() queries. The size of $\mathbf{L}_U$ can increase by at most $O(1)$. The lemma then follows. ■

4.2 The OptpathESeq() Query

In this section, we consider the OptpathESeq$(\mathbf{s}, \mathbf{t}, S)$ query. Specifically, given the source $\mathbf{s}$, the destination $\mathbf{t}$, and an edge sequence $S = \langle e_1, \ldots, e_{|S|} \rangle$ from the arrangement of the refined diagram $\widehat{D}$, we wish to compute an optimal monotone path $\pi^* \in \mathbb{C}(S)$ connecting $\mathbf{s}$ to $\mathbf{t}$, together with its score. Below we first describe a slower $O((n+m)^2)$ algorithm to explain the main idea. The time complexity can then be improved to $O((n+m)\log(nm))$.

4.2.1 An $O((n+m)^2)$ Algorithm

Algorithm setup. First, observe that not all points in e_i is reachable by a monotone path from the source point $\mathbf{s}$ passing through the edge sequence $S_i := \langle e_1, \ldots, e_i \rangle$. We preprocess S by keeping only the portion of e_i that is reachable from $\mathbf{s}$ via S_i. Easy to verify that the reachable portion on each e_i is a single subsegment. The preprocessing takes $O(|S|)$ time. From now on, we assume that S is the set of edges after this preprocessing.

Furthermore, since edges in S are from $\mathrm{Arr}(\widehat{D})$, we can enumerate the configurations of any two consecutive edges in S based on the eight cell-configurations shown in Figure 1. In particular, there are 17 configurations grouped into four categories in Figure 3; ab and AB denote the two consecutive edges e_i and e_{i+1}, respectively. Each configuration is decided by the sign of the slopes and the relative sidedness of ab and AB. The choices of this grouping will be made clear later when

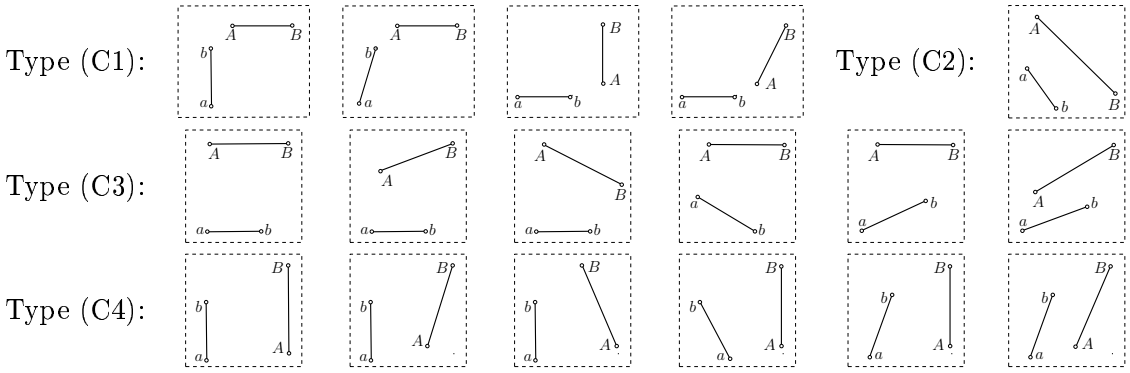
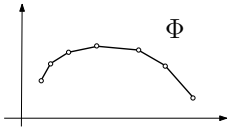


Figure 3: Configurations for two consecutive edges ab and AB in an edge sequence from $\text{Arr}(\widehat{D})$.

we describe the algorithm. Let $\mathcal{H}(ab, AB)$ denote the convex hull of ab and AB. Since all cells in $\text{Arr}(\widehat{D})$ are convex, $\mathcal{H}(ab, BA)$ is either all white or all black.



Let $\mathcal{S}(p, q, S')$ denote the optimal score of any path from p to q, passing through the edge sequence S'. For each e_i, $1 \le i \le |S|$, we maintain the score function $\Phi_i : e_i \to \mathbb{R}$, defined as $\Phi_i(p) = \mathcal{S}(\mathbf{s}, p, S_i)$ for any $p \in e_i$. In what follows, we will show that each Φ_i is a *convex piecewise-linear (PL) function*, where the slopes of the linear pieces are non-increasing. In other words, the graph of a convex PL function is a convex polygonal chain (see the right figure)[①]. Furthermore, we will show that the number of linear pieces in Φ_i, called its *descriptive complexity* $|\Phi_i|$, satisfies that $|\Phi_i| = O(i)$, and Φ_i can be computed from Φ_{i-1} in $O(i)$ time. We say that an object X is *visible* to a point p, denoted by $p \triangleleft X$, if for any point $q \in X$, there is a monotone path from p to q.

Observation 4.5 *Given a point u and an edge e such that $u \triangleleft e$, the function $h_u : e \to \mathbb{R}$, defined as $h_u(p) = \text{len}_1(pu)$ for $p \in e$, is a linear function over e. Furthermore, given two points u and v, h_u and h_v share the same slope.*

Observation 4.6 *Given any edge $e = ab$ with $a \triangleleft b$, a point u, and an edge sequence S', the function $f : e \to \mathbb{R}$, defined as $f(p) = \Phi(u, p, S')$ for $p \in ab$, is an increasing function. Furthermore, if ab is white, then $f(q) \ge f(p) + \text{len}_1(pq)$ for any two points $p \triangleleft q$ on e.*

Lemma 4.7 *For any $1 \le i \le |S|$, Φ_i is a convex PL function of $O(i)$ descriptive complexity, and it can be computed from Φ_{i-1} in $O(i)$ time. A corresponding monotone path can also be computed in the same time complexity.*

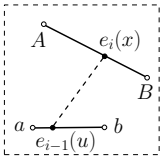
Proof: We prove the lemma by induction. The base case (i.e, $i = 1$) follows from Observation 4.5. Now assume the lemma holds for $F = \Phi_{i-1}$. To prove that it holds for $H = \Phi_i$, we verify it for each configuration of $e_{i-1} = ab$ and $e_i = AB$ in Figure 3 — the 17 configurations are classified into 4 types (C1) – (C4), and those of the same type can be handled similarly. For simplicity, we assume without loss of generality that the region $\mathcal{H}(ab, AB)$ is white. The case when $\mathcal{H}(ab, AB)$ is black is similar and simpler.

(1). Type (C1): Configurations of this type have the property that $a \triangleleft b$, and that all points in AB are visible to every point from ab. Since $a \triangleleft b \triangleleft AB$, for any point $p \in ab$ and $q \in AB$, $\text{len}_1(pq) = \text{len}_1(pb) + \text{len}_1(bq)$. It then follows from Observation 4.6 that $F(p) + \text{len}_1(pq) \le F(b) + \text{len}_1(bq)$.

[①]We remark that the function is in fact a *concave* function. We use the term convex as its graph is visually convex.

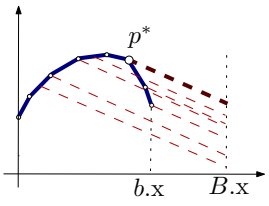
Thus for any point $q \in AB$, $H(q) = \max_{p \in ab}[F(p) + \text{len}_1(pq)] = F(b) + \text{len}_1(bq)$, and there is an optimal path from $\mathbb{C}(S_i)$ to q coming from the point b. Hence H is a convex LP function with just one piece and can be computed in $O(1)$ time.

(2). Types (C3) and (C4): We describe how to update for configurations in (C3), and those in (C4) can be handled in a symmetric manner. Given a point p, let $p.\text{x}$ and $p.\text{y}$ denote its x- and y-coordinates, respectively; p can also be written as the tuple $p = \langle p.\text{x}, p.\text{y}\rangle$. By monotonicity requirement and our preprocessing, it is necessary that $a.\text{x} \leq A.\text{x}$. We assume that $a.\text{x} = A.\text{x} = 0$ for simplicity — if $A.\text{x} > a.\text{x}$, we simply remove the unnecessary portion from the function H obtained, and the resulting function can only have a smaller complexity. Similarly, we assume without loss of generality that $B.\text{x} \geq b.\text{x}$. Now parametrize segments e_{i-1} and e_i by their projection on the x-axis and represent e_{i-1} (resp. e_i) as $e_{i-1}(x) = a + \langle x, \rho_{i-1}x\rangle$ for $x \in [0, b.\text{x}]$ (resp. $e_i(x) = A + \langle x, \rho_i x\rangle$ for $x \in [0, B.\text{x}]$), where ρ_{i-1} (resp. ρ_i) is the slope of e_{i-1} (resp. e_i). From now on, we abuse the notations slightly by using F and H to denote $F \circ e_{i-1}$ and $H \circ e_i$, respectively.



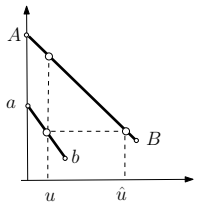
Given an edge e, let $e[x_0, x_1]$ denote the subsegment of e from $e(x_0)$ to $e(x_1)$. For any $x \in [0, b.\text{x}]$, an important property for type-(C3) configurations is that the point $e_{i-1}(x)$ can reach all points in the segment $e_i[x, B.\text{x}]$ (via a monotone path). Now consider the function $f_u : [0, B.\text{x}] \to \mathbb{R}$, where $f_u(x)$ is the best score of any path in $\mathbb{C}(S_i)$ to $e_i(x)$ via the point $e_{i-1}(u)$. For $x < u$, $f_u(x)$ does not exists and we set $f_u(x) = 0$. For $x \geq u$, then the best connection between $e_{i-1}(u)$ to $e_i(x)$ is simply connecting them by a segment (see the right figure for an example). Since we have assumed that $\mathcal{H}(ab, BA)$ is all white, we have that $f_u(x) = F(u) + (e_i(u).\text{y} - e_{i-1}(u).\text{y}) + \rho(x - u)$, where $\rho = 1 + \rho_i$ depends only on the slope ρ_i of the edge e_i, and $(e_i(u).\text{y} - e_{i-1}(u).\text{y}) + \rho(x - u)$ is the distance between $e_{i-1}(u)$ and $e_i(x)$ under the L_1 norm. Hence the graph of f_u is a ray of slope ρ, originated from the point $\langle u, F(u) + (e_i(u).\text{y} - e_{i-1}(u).\text{y})\rangle$. Furthermore, $H(x) = \max_{u \in [0, b.\text{x}]} f_u(x)$ for every $x \in [0, B.\text{x}]$, implying that the graph of H is the upper-envelope of the graphs of f_us for all $u \in [0, b.\text{x}]$.

The above observations suggest the following two-stages procedure to construct H from F. In the first *lifting stage*, we compute $F_1(u) = F(u) + (e_i(u).\text{y} - e_{i-1}(u).\text{y})$, for $u \in [0, b.\text{x}]$. That is, we lift each point in the graph of $F(u)$ vertically by $V(u) := e_i(u).\text{y} - e_{i-1}(u).\text{y} = (A.\text{y} - a.\text{y}) + (\rho_i - \rho_{i-1})u$. Obviously, the function $V : [0, b.\text{x}] \to \mathbb{R}$ is a linear function with slope $\rho_i - \rho_{i-1}$. Now, the graph of the function f_u is simply the ray γ_u originated from the point $\langle u, F_1(u)\rangle$ with slope ρ. All rays for different $u \in [0, b.\text{x}]$ share the same slope and H is the upper envelope of them. The computation of the upper envelope of such rays is the second *enveloping stage*.



Since F_1 is the addition between F and a linear function V, it is still a convex LP function with $|F_1| = |F|$. This is because adding V to any linear piece in F increases the slope of that linear function by the slope of V. Hence the order of the slopes of all linear pieces in F remains the same after adding V. In the second enveloping stage, let $\overrightarrow{\text{v}}$ denote the common direction shared by all rays. Since F_1 is convex, the only ray that can appear on the upper envelope must be originated from $p^* = \langle u^*, F_1(u^*)\rangle$, such that $\overrightarrow{\text{v}}$ is a tangent direction at p^*. See the right figure for an example, where the solid convex chain is the graph of F_1, and the graph of H is the concatenation between the portion of the convex chain before p^* and the thick dashed ray. Note that no other rays (thin dashed rays) can appear on the upper envelope. Hence we have $|H| \leq |F_1| + 1 \leq |F| + 1$ and H can be computed in $O(|F|)$ time by a linear scanning of F.

(3). Type (C2): The main difference between the configuration of type (C2) and those of types (C3) is that a point $e_{i-1}(u) \in ab$ can reach (via a monotone path) only $e_i[u, \hat{u}]$ in AB, instead of reaching $e_i[u, B.x]$ as in the previous case, where $e_i(\hat{u})$ is the intersection point between e_i and the horizontal line passing through $e_{i-1}(u)$. See the right figure. Similar to the previous case, we assume again that $a.x = A.x = 0$, and $b.x \leq B.x$ without loss of generality. Note that both ρ_{i-1} and ρ_i are necessarily negative in this case.



Consider again the function $f_u : [0, B.x] \to \mathbb{R}$, where $f_u(x)$ is the best score of any path in $\mathbb{C}(S_i)$ to $e_i(x)$ via the point $e_{i-1}(u)$. It has the following form, where $\rho = 1 + \rho_i$.

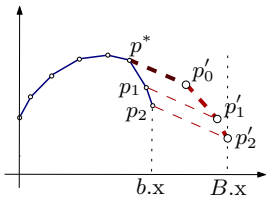
$$f_u(x) = \begin{cases} F(u) + (e_i(u).y - e_{i-1}.y) + \rho(x - u), & \text{for } x \in [u, \hat{u}] \\ 0 & \text{for } x \in (0, u) \cup (\hat{u}, B.x]. \end{cases}$$

This means that during the second enveloping stage, instead of extending a *ray* from each point $\langle u, F_1(u)\rangle$, we now extend only a *segment* from $\langle u, F_1(u)\rangle$ over the interval $[u, \hat{u}]$. All such extension-segments share the same slope ρ. However, they may be of different length.

Furthermore, since $e_{i-1}(u).y = e_i(\hat{u}).y$, we have that $u = \frac{\rho_i}{\rho_{i-1}}\hat{u} + \frac{a.y - A.y}{\rho_{i-1}} = c_1\hat{u} + c_2$. We also note that $u \leq \hat{u}$ for any $u \in [0, b.x]$ (as the two input segments do not intersect). Now assume that we have already obtained $F_1 = F + V$ after the lifting stage. In the second stage, we extend a segment of slope ρ from $\langle u, F_1(u)\rangle$ over the interval $[u, \hat{u}]$, and we trace the right endpoint of this segment. More specifically, let $g(u) = d_1u + d_2$ be one linear piece from F_1. For each point $\langle u, g(u)\rangle$, the right endpoint of the corresponding extending segment is $\langle \hat{u}, h(\hat{u})\rangle$, where

$$h(\hat{u}) \;=\; g(u) + \rho(\hat{u} - u) \;=\; (d_1 - \rho)u + d_2 + \rho\hat{u} \;=\; [c_1(d_1 - \rho) + \rho]\hat{u} + c_2(d_1 - \rho) + d_2.$$

Hence $h(\hat{u})$ is a linear function, and its slope depends only on e_{i-1}, e_i, and the slope of the linear piece g. Thus as $\langle u, F_1(u)\rangle$ changes, $h(\hat{u})$ is a piecewise-linear function with $|F_1|$ number of pieces. Furthermore, since ρ_{i-1} and ρ_i are both negative, $c_1 = \rho_i/\rho_{i-1}$ is positive. Thus given two linear pieces from F_1 with slopes d_1 and d_1', the resulting pieces in h still maintain the order between d_1 and d_1'. Hence h is also convex PL.



Finally, H is the upper envelope of all these extending segments. Let $\overrightarrow{v}$ be the common direction shared by all extending segments, and $p^* = \langle u^*, F_1(u^*)\rangle$ the point in the graph of F_1 such that $\overrightarrow{v}$ is a tangent direction at p^*. The graph of H is the concatenation between the portion of the graph of F_1 before p^*, the extending segment at p^*, and the portion of the graph of h after $\hat{u}^*$. See the right figure for an example where $p_0' = \langle \hat{u}^*, h(\hat{u}^*)\rangle$. (Note that the slope of $p_0'p_1'$ (resp. of $p_1'p_2'$) may be different from that of p^*p_1 (resp. p_1p_2). But the order between these slopes are maintained.) Hence $|H|$ is still convex PL, and $|H| = |F_1| + 1 = |F| + 1$.

Putting all cases together, the lemma then follows. ■

By the above lemma, we can compute the score function on the last edge in S, thus the optimal score to the destination t, in $O(|S|^2)$ time. We improve the time complexity to near linear in next section.

4.2.2 A Faster Algorithm

We now show that, by representing a convex PL function in an appropriate data structure, we can compute Φ_i from Φ_{i-1} in $O(\log |S|)$ *amortized time*, and thus answer the $\mathsf{OptpathESeq}(s, t, S)$ query in $O(|S| \log |S|)$ time.

A data structure for convex PL-functions. In general, a PL function H can be represented as a list of break points $b_1, \ldots, b_{l+1}$, associated with a sequence of linear functions $H_i : [b_i, b_{i+1}] \to \mathbb{R}$. We now describe a way to represent H using a balanced binary tree $T = T(H)$. The goal is to enable efficient updates of the PL function H.

More specifically, T is the following augmented balanced interval tree: From left to right, the ith leaf of T, denoted by l_i, corresponds to interval $[b_i, b_{i+1}]$. Every leaf node l_i stores a linear function $f_{l_i} : [b_i, b_{i+1}] \to \mathbb{R}$, called the *base function* at leaf l_i. Each internal node $v \in T$ is associated with an interval $I_v = [b_{v_l}, b_{v_r+1}]$ where v_l and v_r are the indices of the left-most and right-most leaves of the subtree T_v rooted at v, respectively. It also stores a *modification record (MD-record)* that specifies how to modify every base function within the subtree T_v. In general, a linear function h can be described by a pair of parameter (ρ, ω), such that $h(x) = \rho x + \omega$. An MD-record μ contains two functions $\langle \mu_s, \mu_t \rangle$ such that it modifies an input linear function $h = \rho x + \omega$ to a new linear function $h' = \mu_s(\rho, \omega)x + \mu_t(\rho, \omega)$. Each function μ_s (or μ_t) is a linear combination of ρ and ω that has the simple form that $\mu_s(\rho, \omega) = c_1\rho + c_2\omega + c_3$ for some constants c_is. Hence μ_s and μ_t are each described by three constants. As an example, if we wish to add a linear function $h'(x) = ax + b$ to function $h = \rho x + \omega$, we simply set $\mu_s(\rho, \omega) = \rho + a$ and $\mu_t(\rho, \omega) = \omega + b$. An idendity MD-record $\langle \mu_s, \mu_t \rangle$ is simply $\mu_s(\rho, \omega) = \rho$, and $\mu_t(\rho, \omega) = \omega$.

The composition of two MD-records $\mu^1 = \langle \mu_s^1, \mu_t^1 \rangle$ and $\mu^2 = \langle \mu_s^2, \mu_t^2 \rangle$ can be computed in constant time, and the result is a new MD-record $\langle \mu_s, \mu_t \rangle = \langle \mu_s^1, \mu_t^1 \rangle \circ \langle \mu_s^2, \mu_t^2 \rangle$. Note that this operation is not commutative. To obtain H_i, the ith piece of linear function of H, we first find the path from the root v_0 of T to the leaf l_i corresponding to the interval $[b_i, b_{i+1}]$. Let $A_i = \{v_0, \ldots, v_m = l_i\}$ be the set of nodes along this path. Let $h_m = \rho_m x + \omega_m$ be the base function stored at l_i, and μ^j the MD-record stored at v_j, for $j \in [0, m-1]$. The linear function $H_i : [b_i, b_{i+1}] \to \mathbb{R}$ is simply $H_i = \mu^0 \circ \mu^1 \circ \cdots \circ \mu^{m-1}(\rho_m, \omega_m)$. It is important to note that since the composition is not commutative, this implies that we apply μ^v to a base function in its subtree *only after* we apply all the MD-records along the path from v to the corresponding leaf. The entire process takes $O(m)$ time. For a PL function with complexity l, $m = O(\log l)$ since the tree is balanced.

Given such an augmented binary tree representation of a PL function with complexity l, we can insert or delete k consecutive intervals (together with corresponding linear functions) in $O(\log l + k)$ time. The main difference from standard balanced binary tree operations is to maintain the MD-record properly, especially during rotations. We omit the straightforward, but tedious details.

Given any interval $[b_i, b_j]$, one can also identify $O(\log l)$ number of canonical and disjoint intervals whose union forms $[b_i, b_j]$, and each such interval corresponds to a leaf or an internal node of the tree T. This set of intervals is called the *canonical decomposition of* $[b_i, b_j]$.

Now suppose we wish to modify all functions from H over an interval $[b_i, b_k]$ by an MD-record μ^*. We first identify the set of nodes corresponding to the canonical decomposition of $[b_i, b_k]$. For any interior node v we visit during the descending from the root to search for these canonical decomposition nodes, we push the MD-record μ^v to its children, and change the MD-record of v to identity. Let μ^l and μ^r be the MD-record associated with the left and right children of v before this pushing down operation, respectively. Afterwards, the new MD-records are $\mu^{l*} = \mu^v \circ \mu^l$ and $\mu^{r*} = \mu^v \circ \mu^r$. As a result, let u be a node corersponding to an interval in the canonical decomposition of $[b_i, b_k]$. Let $\{v_0, \ldots, v_m = u\}$ be the set of nodes along the path from the root v_0 to u, and μ^j the MD-record stored at v_j. After the descending to u, all nodes v_j's have an identity as their MD-record, for $j \in [0, m-1]$. For u, we set its MD-record $\mu = \mu^* \circ \mu^0 \circ \mu^1 \circ \cdots \circ \mu^m$. The entire process takes $O(\log l)$ time.

Similarly, we also store an *interval-modification record (IM-record)* with each interior node v, which is a linear function I_v that will apply to all break points within the subtree rooted at v. For example, $I_v = c_1 x + c_2$ means that a break point b_i will now move to $b_i' = c_1 b_i + c_2$. The IM-records

can be maintained and updated, similarly to that of MD-records.

Finally, we also augment the data structure so that at each internal node corresponding to the interval $[b_{v_l}, b_{v_r}]$, we also store the slopes of the two linear functions following b_{v_l} and b_{v_r}. If the input function H is convex, then given an angle θ, we can perform a standard binary search to locate the first interval $[b_i, b_{i+1}]$ so that the slope of H_i is at most θ in $O(\log l)$ time. We call this operation an *elbow-search operation.*

Lemma 4.8 *Given two points $s, t \in \widehat{D}$, and an edge sequence S from $\mathrm{Arr}(\widehat{D})$, a* OptpathESeq*$(s, t, S)$ query can be answered in $O(|S| \log |S|)$ time.*

Proof: Our algorithm performs $|S|$ rounds, and maintains Φ_i using the above data structure at each iteration. For each update, we only need $O(1)$ number of the following operations: inserting a new break point, deleting k consecutive break points, modifying all linear functions within an interval, modifying all break points within an interval, and performing an elbow-search query. All operations take $O(\log i)$ time each, other than the deletion operation, which takes $O(\log i + k)$ time to delete k consecutive break points. However, note that each iteration can create at most one new break point. Thus the total number of break points ever created throughout all $|S|$ iterations, which is the same as the total number of break points that can ever be deleted, is $O(|S|)$. Hence the amortized cost for the deletion operation at each iteration is $O(\log i)$. This implies that we can compute $\Phi_{|S|}$, thus answering the OptpathESeq$(\mathbf{s}, \mathbf{t}, S)$ query, in $O(|S| \log |S|)$ time and $O(|S|)$ space. Thus proves the lemma. ■

4.3 Putting Everything Together

We process cells in $\mathrm{Arr}(G)$ in order, maintaining the list of candidate-set for each cell boundary edge as described in Section 4.1. Once the last cell is processed, we can compute the optimal path from $\mathbf{b}$ to the top-right vertex $\mathbf{t}$ of D in $O(N)$ number of OptpathESeq() queries, where N is the size of the list of candidate edge-sequences associated with the horizontal and the vertical edges from $\mathrm{Arr}(G)$ incident to $\mathbf{t}$. Furthermore, since we consider conformal monotone paths, any edge-sequence we ever encounter has size $O(n + m)$.

The remaining question is to bound the total number of OptpathESeq() queries. There are altogether $O(nm)$ cells in $\mathrm{Arr}(G)$. Hence the size of the candidate-set for any grid edge in $\mathrm{Arr}(G)$ is at most $O(nm)$ by Lemma 4.3, and it makes $O(nm)$ number of OptpathESeq() queries to process each cell. This generates an $O(n^2m^2)$ upper bound on the total number of OptpathESeq() queries. We can further improve this bound to $O(nm(n + m))$ by a global argument. First, we need the following result.

Claim 4.9 *Given a grid M of k_1 columns and k_2 rows such that Lemma 4.3 holds for each cell in M, suppose the set of candidate-sets $\mathbf{L}_L$ on the left and bottom boundary edges of M is given. Process cells in M in order as described in Section 4.1. Then, the size of candidate-sets associated with the set of upper and right boundary edges of M is bounded by $O(|\mathbf{L}_L| + k_1k_2)$. The total number of* OptpathESeq*() queries involved is $O((|\mathbf{L}_L| + k_1k_2)(k_1 + k_2))$.*

Proof: Let $H[i][j]$ and $V[i][j]$ denote the (i, j)'th horizontal and vertical edges of M, respectively, for $0 \le i \le k_1$ and $0 \le j \le k_2$. Let $h[i][j]$ (resp. $v[i][j]$) be the size of the candidate-set associated with the edge $H[i][j]$ (resp. $V[i][j]$). Note that

$$|\mathbf{L}_L| = \sum_{i \in [1,k_1]} h[i][0] + \sum_{j \in [1,k_2]} v[0][j].$$

Let $\mathbf{L}_U$ denote the union of candidate-sets from all top and right edges of M. Then we have:

$$|\mathbf{L}_U| = \sum_{i\in[1,k_1]} h[i][k_2] + \sum_{j\in[1,k_2]} v[k_1][j].$$

Consider an arbitrary column, say, the Ith column. For the jth cell in this column, its left and bottom boundary edges are $V[I-1][j]$ and $H[I][j-1]$, respectively. Its right and top edges are $V[I][j]$ and $H[I][j]$, respectively. Since Lemma 4.3 holds for each cell, we have that

$$h[I][j] + v[I][j] = h[I][j-1] + v[I-1][j] + O(1), \quad \text{for any } j \in [1,k_2].$$

Since the top edge of the jth cell is the same as the bottom edge of the $j+1$th cell in this column, we have that

$$\sum_{j=1}^{k_2}(h[I][j] + v[I][j]) = \sum_{j=1}^{k_2}(h[I][j-1] + v[I-1][j]) + O(k_2)$$

$$\Rightarrow \quad h[I][k_2] + \sum_{j=1}^{k_2} v[I][j] = h[I][0] + \sum_{j=1}^{k_2} v[I-1][j] + O(k_2). \tag{1}$$

Summing up Eqn (1) for all $I \in [0,k_1]$, we have that $|\mathbf{L}_U| = |\mathbf{L}_L| + O(k_1k_2)$.

To show the second part of the claim, first, it is easy to verify that total cost of processing all cells is

$$O(\sum_{i\in[1,k_1]}\sum_{j\in[0,k_2]} h[i][j] + \sum_{i\in[0,k_1]}\sum_{j\in[1,k_2]} v[i][j]\ ).$$

Furthermore, by repeatedly applying Eqn (1), we have that the total size of candidate-sets for all the right boundary edges of the Ith column is $O(\sum_{i\in[1,I]} h[i][0] + \sum_{j\in[1,k_2]} v[0][j] + k_2I)$, implying that

$$O(\sum_{i\in[0,k_1]}\sum_{j\in[1,k_2]} v[i][j]) = O(k_1\sum_{i\in[1,k_1]} h[i][0] + k_1\sum_{j\in[1,k_2]} v[0][j] + k_2k_1^2) = O(k_1|\mathbf{L}_L| + k_2k_1^2).$$

A symmetric argument shows that

$$O(\sum_{i\in[1,k_1]}\sum_{j\in[0,k_2]} h[i][j]) = O(k_2\sum_{i\in[1,k_1]} h[i][0]k_2 + \sum_{j\in[1,k_2]} v[0][j] + k_1k_2^2) = O(k_2|\mathbf{L}_L| + k_1k_2^2).$$

It then follows that the total cost is bounded by:

$$O(\sum_{i\in[1,k_1]}\sum_{j\in[0,k_2]} h[i][j] + \sum_{i\in[0,k_1]}\sum_{j\in[1,k_2]} v[i][j]\ ) \quad = \quad O((|\mathbf{L}_L| + k_1k_2)(k_1+k_2)).$$

This proves the claim. ■

Now consider the refined diagram $\widehat{D}$, as a two-level grid. The high level is the original free-space diagram D. Each cell in D can be further considered as a second-level grid consisting of extension edges in $\widehat{D}$. Apply the above claim to both levels of the grids. First, since each cell C in D is decomposed into a $c_1 \times c_2$ grid in the refined diagram $\widehat{D}$, where c_1, c_2 are constants. Claim 4.9 implies that Lemma 4.3 holds for C as well. Hence we can apply Claim 4.9 again, this time for the $n \times m$ grid D, and the total number of OptpathESeq() queries needed to compute the candidate-sets for all edges in D is $O(nm(n+m))$.

Putting everything together, we conclude with the following main result.

Theorem 4.10 *One can build a data structure by making $O(nm(n+m))$ OptpathESeq() queries in $O(nm(n+m)^2\log(nm))$ total time, so that the shortest path from the left-bottom vertex $\mathbf{b}$ to any point in $\widehat{D}$ can be computed in $O(nm(n+m)\log(nm))$ time.*

5 Conclusion

In this paper, we propose a natural extension of the Fréchet distance to measure the partial similarity between curves. Specifically, given two polygonal curves P and Q in $\mathbb{R}^d$ of sizes n and m, respectively, the partial Fréchet similarity between them under the L_1 or L_∞ norm can be computed in $O(nm(n+m)^2 \log(nm))$ time and $O((n+m)^2)$ space. This is the first polynomial-time algorithm for computing continuous partial curve similarity that allows general types of outliers.

It is interesting to see whether the time complexity can be improved. The number of OptpathESeq() queries in our algorithm is tight — one can construct an example where the total number of edge-sequences necessary to cover all grid edges in $refinediagram$ is $O(nm(n+m))$. However, can one process each cell more efficiently by exploiting the temporal coherence existing in the OptpathESeq() queries for neighboring edge sequences?

The optimal matching under the L_1 or the L_∞ norm provide certain approximation for the optimal matching the L_2 norm. A natural next question is to compute the partial Fréchet similarity under the L_2 norm *exactly*. Under the L_2 norm, the white region in the free space diagram is no longer polygonal, and little is known for the structure of the optimal partial matching under the L_2 norm. It will be interesting to explore in this direction.

Other important questions include developing efficient approximation algorithms for the partial Fréchet similarity problem, and for the problem of finding the best partial matching under translations or rigid motions. The partial matching problem under transformations appears to be especially challenging. Current solutions for the matching problems under transformations often rely on characterizing the potentially optimal transformations combinatorially. Such characterization is usually based on the fact that some specific points should be matched. However, it is unclear whether one can find such "special points" in the case of continuous partial matching.

Acknowledgement. The author would like to thank anonymous reviewers for very helpful comments, especially one pointing out an incorrect statement in the earlier version of this paper about the "equivalent classes" (which is no longer necessary in current version of the paper). The author also thanks Sariel Har-Peled for useful discussions.

References

[1] P. K. Agarwal, S. Har-Peled, N. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification in two and three dimensions. *Algorithmica*, 42:203–219, 2005.

[2] H. Alt and M. Buchin. Semi-computability of the Fréchet distance between surfaces. In *Proc. 21th European Workshop on Computational Geometry*, 2005.

[3] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *J. Algorithms*, 49:262–283, 2003.

[4] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[5] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B. V. North-Holland, Amsterdam, 2000.

[6] H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the fréchet distance. In *Proc. 18th Internat. Sympos. Theoret. Asp. Comp. Sci.*, pages 63–74, 2001.

[7] H. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

[8] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, Revisited. In *Proc. 14th Annu. European Sympos. Algorithms*, pages 52–63, 2006.

[9] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, pages 853–864, 2005.

[10] K. Buchin, M. Buchin, and C. Wenk. Computing the Fréchet distance between simple polygons in polynomial time. In *Proc. 22st Annu. ACM Sympos. Comput. Geom.*, pages 80–87, 2006.

[11] M. Buchin. *On the Computability of the Fréchet Distance Between Triangulated Surfaces.* PhD thesis, Dept. of Comput. Sci., Freie Universitt Berlin, 2007.

[12] D. Cardoze and L. Schulman. Pattern matching for spatial point sets. In *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 156–165, 1998.

[13] E. W. Chambers, E. C. de Verdière, J. Erickson, S. Lazard, F. Lazarus, and S. Thite. Walking your dog in the woods in polynomial time. In *Proc. 24th Annu. ACM Sympos. Comput. Geom.*, 2008. To appear.

[14] M. Clausen and A. Mosig. Approximately matching polygonal curves with respect to the Fréchet distance. *Comput. Geom. Theory Appl.*, 30:113–127, 2005.

[15] A. F. Cook and C. Wenk. Geodesic Fréchet distance inside a simple polygon. In *Proc. 25th Internat. Sympos. Theoret. Asp. Comp. Sci.*, pages 193–204, 2008.

[16] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields, new algorithms for computing similarity between curves. *J. Mathematic Imaging and Vision, To appear*, 2007.

[17] A. Efrat, S. Har-Peled, L. J. Guibas, J. S. Mitchell, and T. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete Comput. Geom.*, 28:535–569, 2002.

[18] P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 102–106, 2002.

[19] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 457–465, 1999.

[20] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive dataset. In *Proc. of the Third Euro. Conf. Princip. Data Mining and Know. Disc.*, pages 1–11, 1999.

[21] M. Kim, S. Kim, and M. Shin. Optimization of subsequence matching under time warping in time-series databases. In *Proc. ACM symp. Applied comput.*, pages 581–586, New York, NY, USA, 2005.

[22] R. Kolodny, P. Koehl, and M. Levitt. Comprehensive evaluation of protein structure alignment: Scoring by geometric measures. *J. Mol. Biol.*, 346:1173–1188, 2005.

[23] S. Kwong, Q. H. He, K. F. Man, K. S. Tang, and C. W. Chau. Parallel genetic-based hybrid pattern matching algorithm for isolated word recognition. *Int. J. Pattern Recognition & Artificial Intelligence*, 12(5):573–594, August 1998.

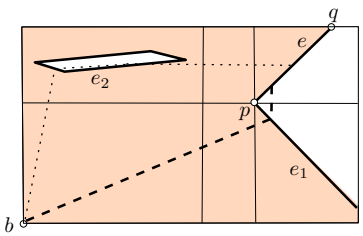
[24] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 15, pages 633–701. Elsevier, 2000.

[25] M. Parizeau and R. Plamondon. A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):710–717, 1990.

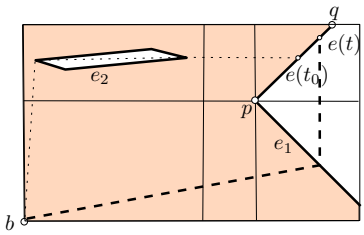
[26] G. Rote. Computing the Fréchet distance between piecewise smooth curves. Technical Report ECG-TR-241108-01, Freie Universität, Berlin, May 2005. To appear in Comput. Geom. Theory Appl.

[27] C. Wenk. *Shape Matching in Higher Dimensions.* PhD thesis, Dept. of Comput. Sci., Freie Universität, Berlin, 2002.

A An Example



(a)



(b)

Given a slope edge e, one can construct an example where there is an edge-sequence S such that the set of points in e it is optimal at does not form a single subsegment. See the right figure for an illustration. Note that we show only part of the free-space diagram in the figure, not the refined diagram. Consider the edge $e = pq$. Assume that the slope of e is 1 and that of e_1 is -0.5. Parametrize e by $e(t) = (1-t)p + q$. When t is small, the optimal path from $\mathbf{b}$ to $e(t)$ will follow the configuration as shown by the think dashed chain in (a). We call such path *right-paths*. As t increases, it is possible to reach $e(t)$ via a path from left, as shown by the thin dotted path in (a). We call such path *left-paths*. At this point, the rate of increasing for the score of the left-path, with respect to t, is at least $1 + 1/(slope(e) \cdot slope(e_2)) = 1 + 1/slope(e_2)$. Adjust the slope of e_2 to make it small enough so that the score coming from such left-path soon exceeds that of the right path. However, for $t > t_0$ as shown in (b), the optimal path from the left will first reach $e(t_0)$, and then follow the edge of e to $e(t)$. Hence the rate of increasing for the score of the left-path now becomes $1 + slope(e) = 2$, while the rate of increasing for the score of the right-path is $1 + slope(e) + slope(e_1) = 2.5$. Thus for e long enough, at some point the score of the right-path exceeds that of the left-path. Hence the edge-sequence optimal at p and q are the same (by following the right-path), while in the interior, there is a subsegment of e that can only be optimally reached by a different edge-sequence (by following the left-path). Note that this example exploits the fact that we are allowing only monotone paths.