On Length-Sensitive Fréchet Similarity

Kevin Buchin[1][0000−0002−3022−7877], Brittany Terese Fasy[2][0000−0003−1908−0154], Erfan Hosseini Sereshgi[3][0000−0003−2548−7428], and Carola Wenk[3][0000−0001−9275−5336]

[1] TU Dortmund, Germany
kevin.buchin@tu-dortmund.de
[2] Montana State University, United States
brittany.fasy@montana.edu
[3] Tulane University, United States
{shosseinisereshgi,cwenk}@tulane.edu

Abstract. Taking length into consideration while comparing 1D shapes is a challenging task. In particular, matching equal-length portions of such shapes regardless of their combinatorial features, and only based on proximity, is often required in biomedical and geospatial applications. In this work, we define the length-sensitive partial Fréchet similarity (LSFS) between curves (or graphs), which maximizes the length of matched portions that are close to each other and of equal length. We present an exact polynomial-time algorithm to compute LSFS between curves under L_1 and L_∞. For geometric graphs, we show that the decision problem is NP-hard even if one of the graphs consists of one edge.

Keywords: Fréchet distance, partial matching, curves, graphs in $\mathbb{R}^2$

1 Introduction

Measuring the similarity between geometric objects is a fundamental task with many applications. One way to measure similarity is to take the reciprocal of a distance measure, such as the Fréchet distance for curves [7]. While the Fréchet distance matches the entire curves to each other, many situations require only matching parts of the curves. For instance, if we want to evaluate whether one curve is a subset of the other, a perfect matching makes little sense. For such situations, partial Fréchet similarity has been proposed [11], which aims to match curves to each other so that the total length of the portions that are close to each other is maximized.

When maximizing lengths, the matching may be skewed towards noisier portions of a curve, as noise makes a curve longer. While this may be acceptable if the noise occurs on both curves, it should not be possible to increase the similarity score by adding noise on one of the curves, such as in the example of Fig. 1, where the partial Fréchet similarity matches the spiral of g to the middle of f. To

[2] Supported by the National Science Foundation grant CCF 2046730
[3] Supported by the National Science Foundation grant CCF 2107434

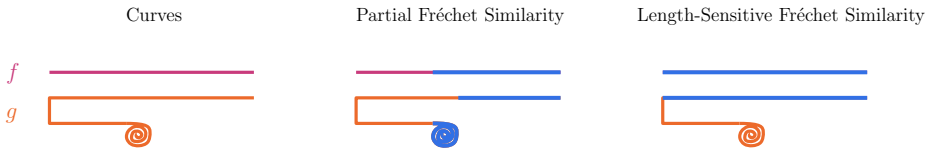


Fig. 1: Two curves f, g with different matchings, illustrated in blue.

mitigate this issue, we introduce the *length-sensitive Fréchet similarity (LSFS)*, where only equal-length portions of curves that are close to each other may be matched. as in Fig. 1 (right).

Non-partial length-sensitive Fréchet matchings have been considered previously. Buchin, Buchin, and Gudmundsson consider speed constraints on Fréchet matchings [10], and Maheshwari et al. study the Fréchet distance with *speed limits* [19]. Similar speeds on the curves result in matching similar lengths to each other. However, all of these approaches restrict to matching the whole curves within a given distance, which is very different from the setting of partial matchings, and makes the techniques non-applicable in our setting.

Our motivation for studying length-sensitive partial Fréchet matchings stems from the problem of comparing paths [18] and geometric networks [2, 3, 5, 6, 13], more specifically from the *map reconstruction problem*. To evaluate different reconstruction algorithms that construct road networks from movement trajectories (e.g., [4, 9, 12, 15, 16]), one needs to evaluate how similar the reconstructions are to the underlying network. Usually trajectories only cover portions of the network, and therefore a partial similarity is desired. However, to avoid favoring algorithms that build noisy reconstructions, length-sensitive matchings are desirable. In this context, length-sensitive Fréchet matchings can be seen as a continuous version of the commonly used graph-sampling technique [1, 8].

Our Contributions In this paper, we define length-sensitive partial Fréchet similarity (LSFS) for curves and geometric graphs. Specifically, the LSFS maximizes the length of matched portions that are close to each other and of equal length. In Section 3, we define LSFS for curves and graphs, providing a clean mathematical definition for this intuitive concept. For geometric graphs we show in Section 5 that LSFS is NP-hard even if one of the graphs consists of only one edge. In Section 4, we present a polynomial-time algorithm to compute LSFS for curves under L_1; the same approach generalizes to L_∞ as well.[4]

2 Preliminaries

A **polygonal curve** f in $\mathbb{R}^d$ is a finite sequence of line segments (or edges) in $\mathbb{R}^d$. Its **length**, $\text{len}(f) = L_f$, is the sum of the lengths of the edges. Another way to represent f is as an arc-length parameterized map $f\colon [0, L_f] \to \mathbb{R}^d$,

[4] We do not consider L_2, since we expect to encounter the same algebraic obstacles that make partial Fréchet similarity unsolvable over the rational numbers [14].

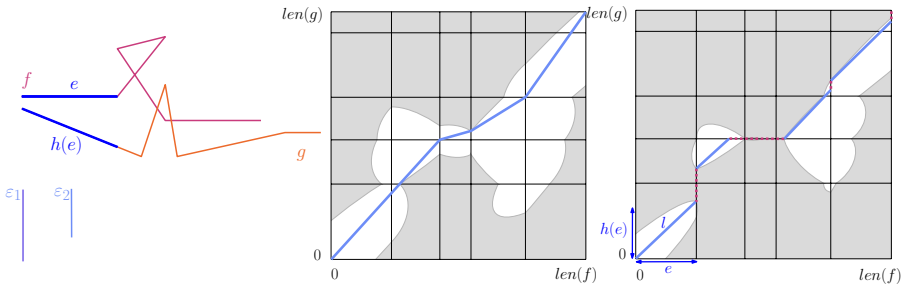
where $\text{len}(f([0,t])) = t$ for all $t \in [0, L_f]$. A **geometric graph (G, ϕ)** in $\mathbb{R}^d$ is a finite abstract graph $G = (V_G, E_G)$ along with a continuous map $\phi\colon G \to \mathbb{R}^d$ such that for each edge $e \in E$, the restriction $\phi|_e$ is a polygonal curve. The **length** of G is $\text{len}(G) = \sum_{e \in E} \text{len}(\phi|_e)$. $\mathcal{G}$ denotes the set of all geometric graphs in $\mathbb{R}^d$, up to reparameterization, i.e., (G, ϕ_G) and (H, ϕ_H) are equivalent in $\mathcal{G}$ if there exists a homeomorphism $h\colon G \to H$ such that $\phi_G = \phi_H \circ h$. Let $(G, \phi_G), (H, \phi_H) \in \mathcal{G}$ and let $h\colon H \to G$ be a function that is homeomorphic onto its image. If, for each path π in (H, ϕ_H), the path $h(\pi)$ in (G, ϕ_G) has the same (intrinsic) length, we say that h is **length-preserving**.

The **Fréchet distance** between two curves f and g in $\mathbb{R}^d$ is

$$d_F(f, g) = \inf_{h:[0,L_f]\to[0,L_g]} \max_{t\in[0,L_f]} ||f(t) - g(h(t))||_p, \tag{1}$$

where h is an homeomorphism such that $h(0) = 0$, and $||.||_p$ is the p-norm. In this paper, we focus on the case when $p = 1$ or $p = \infty$.

For polygonal curves f and g and threshold $\varepsilon > 0$, Alt and Godau [7] provided a polynomial time dynamic programming algorithm based on the free-space diagram. The **free-space diagram $\mathcal{D}_\varepsilon(f, g)$** is a binary function defined over $[0, L_f] \times [0, L_g]$. For a point $(x, y) \in [0, L_f] \times [0, L_g]$, $\mathcal{D}_\varepsilon(f, g)$ is **free** (colored white) if $||f(x) - g(y)||_p \leq \varepsilon$; otherwise, it is **infeasible** (colored gray). The set of all free points is the **free-space**. See Fig. 2. If f, g are polygonal curves with m and n edges, respectively, then $[0, L_f] \times [0, L_g]$ can be decomposed into **cells** of an $m \times n$ grid, where the i-th column corresponds to the i-th edge of f and the j-th row corresponds to the j-th edge of g. By convexity of the L_p-norm, free space within a cell is convex; if $p = 2$ it is the intersection of an ellipse with the cell, if $p = 1$ or $p = \infty$ it is the intersection of a parallelogram with the cell. There exists a bi-monotone path in the free space from $(0, 0)$ to (L_f, L_g) if and only if $d_F(f, g) \leq \varepsilon$. The dynamic program marches through $\mathcal{D}_\varepsilon(f, g)$ cell-by-cell propagating reachability information from $(0, 0)$.



(a) Two Curves. (b) FSD with ε_1. (c) LSFS with ε_2.

Fig. 2: The free-space diagram $\mathcal{D}_\varepsilon(f, g)$, for $p = 2$, for two threshold values, ε_1 and ε_2. In (b), there is a path from $(0, 0)$ to (L_f, L_g), so $d_F(f, g) \leq \varepsilon_1$. In (c), the slope-one line segments (shown in blue) correspond to length-preserving matchings. e is a line segment on f that was matched with $h(e)$ on g.

3 Defining Length-Sensitive Fréchet Similarity

Here, we introduce the length-sensitive Fréchet similarity (LSFS), a notion of similarity between curves (or between graphs).

3.1 LSFS For Curves

Let f, g be two curves and let $\varepsilon > 0$. To define length-sensitive Fréchet similarity, we consider the homeomorphisms h in Equation (1), which correspond to paths γ_h in the free-space diagram $\mathcal{D}_\varepsilon(f, g)$ from $(0, 0)$ to (m, n). We maximize the length of the portions of γ_h that are free and length-preserving. As before, free portions of γ_h are exactly the points such that $||f(x) - g(h(x))||_p \leq \varepsilon$. The length-preserving matchings between sub-curves of f and g correspond to slope-one segments of γ_h. Putting this together, we get $\boldsymbol{I_h^\varepsilon} =$

$$\{x \in [0, L_f] \mid (x, h(x)) \text{ is free and } \exists \delta > 0 \colon h|_{(x-\delta, x+\delta)} \text{ is length-preserving.}\}$$

Here, I_h^ε is the portion of (the parameter space of) f that h maps to g in a length-preserving way while staying within distance ε. We quantify this by defining the **length-sensitive Fréchet similarity (LSFS)** as:

$$\boldsymbol{\mathbb{F}_\varepsilon(f, g)} = \sup_{h:[0,L_f]\to[0,L_g]} \text{len}(I_h^\varepsilon) \ ,$$

where h ranges over all homeomorphisms such that $h(0) = 0$. We interpret I_h^ε as a set of (maximal) intervals, which means that $f(I_h^\varepsilon)$ is a set of subcurves in $\mathbb{R}^d$. Then $\text{len}(I_h^\varepsilon)$ measures the total length of these subcurves of f, since f is arc-length parameterized.[5] Note that the definition above only requires h to be *locally* length-preserving (i.e., only within δ of x). However, this is actually a global property, as proven in Corollary 1 in Appendix A.

We note that the condition of equal length is less harsh than it might seem at first: sub-curves that are close and have similar lengths can still be matched; the score is the length of the shorter portion. As an example of LSFS, consider Fig. 2. We are looking for a bi-monotone path from bottom-left to top-right that maximizes the total length of slope-one segments in the free (white) space. The line segment l, which has slope one, indicates a length-preserving matching because the corresponding matched line segments, e and $h(e)$ on the two curves.

3.2 LSFS For Graphs

Let $(G, \phi_G), (H, \phi_H) \in \mathcal{G}$ and $\varepsilon > 0$. We extend the definition of LSFS to a similarity measure between graphs. Let G be a graph, let C be a connected subgraph of G, and let $h \colon C \to H$ be a continuous map that is homeomorphic onto its image. We define $\boldsymbol{C_h^\varepsilon} =$

$$\{x \in C \mid ||\phi_G(x) - \phi_H(h(x))||_p \leq \varepsilon \text{ and } \exists \delta > 0 \colon h|_{\mathbb{B}_p(x,\delta)} \text{ is length-preserving}\}.$$

[5] For LSFS it suffices to measure lengths of subcurves on f. Partial Fréchet similarity [11] measures lengths of subcurves on f and g, but is not length-preserving.

Here, $\mathbb{B}_p(x, \delta)$ is the open ball centered at x with radius $\delta \in \mathbb{R}_{\geq 0}$. Fig. 3 shows an example. The set C_h^ε is a subgraph of (G, ϕ_G), and so $(C_h^\varepsilon, \phi_G|_{C_h^\varepsilon})$ is in $\mathcal{G}$. While C is a connected graph, we note that C_h^ε need not be connected.

The restriction to connected components of G is important. For example, consider Fig. 4, which demonstrates what can happen if we do not enforce this requirement.

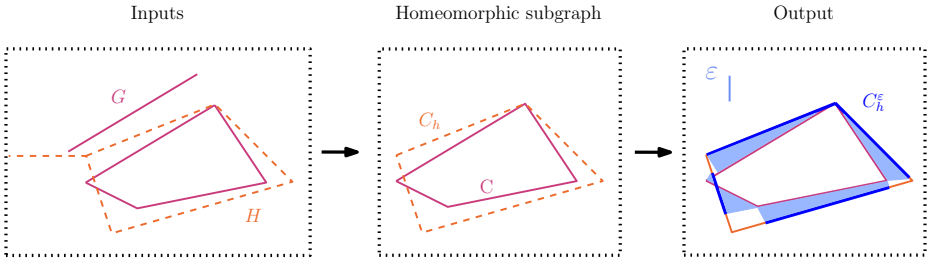


Fig. 3: Computing C_h^ε on two given graphs, G (in magenta) and H (orange). First, we take C, a connected subgraph of G. Then, we find $h\colon C \to H$, a map such that C is homeomorphic onto its image. For a given ε, dark blue segments are C_h^ε. The total length of those segments is the LSFS, $\mathbb{F}_\varepsilon(G, H)$.

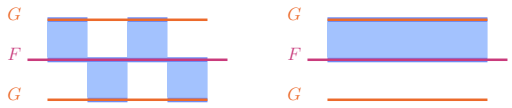


Fig. 4: Two graphs, F (in magenta) and G (orange). Blue represents the matched portions of the graphs. Although the total length matched is the same for both examples, the right is more preferable, as it matches one connected component of G to an entire interval instead of breaking it up into four connected matchings.

Given this setup, the **length-sensitive Fréchet similarity** is the maximum matched length that can be obtained through such homeomorphisms:

$$\mathbb{F}_\varepsilon(G, H) = \sup_{C \subset G} \sup_{h: C \to H} \operatorname{len}(C_h^\varepsilon).$$

4 Computing LSFS for Curves

In this section, we present a polynomial-time dynamic programming algorithm for computing the LSFS for two curves in $\mathbb{R}^2$ under the L_1 and L_∞ norms. To enable efficient propagation of the score function (defined in Section 4.2), we first refine each cell of the free-space diagram (Section 4.1). We then concentrate on a single refined cell and show in Section 4.4 how to propagate the score function based on the lemmas and observations from Section 4.3. Moreover, we show the complexity of the score function in each refined cell. Finally, in Section 4.5, we describe the overall dynamic programming algorithm and its total complexity.

4.1 Refining the Free-Space Diagram

To compute the score function using dynamic programming, we refine the free-space diagram as follows: Consider $\mathcal{D}_\varepsilon(f, g)$, using $p = 1$ or $p = \infty$. Then the free space of a diagram grid cell $\mathcal{C}' = \mathcal{D}_\varepsilon[i][j]$ is the intersection of a parallelogram with the cell. Thus, the free space in $\mathcal{C}'$ is defined by a polygon with up to eight vertices, and the infeasible space can have up to four connected components which may not be convex. For every vertex v of the free-space polygon in $\mathcal{C}'$, we extend a horizontal and a vertical line from v to the boundaries of the cell. This results in a subdivision of $\mathcal{C}'$ that splits all polygons in the cell into (simpler) triangles, quadrilaterals, and pentagons; see Fig. 5. In addition, each sub-cell now has at most two connected components of the infeasible space, each of which is a convex polygon defined by at most five vertices.

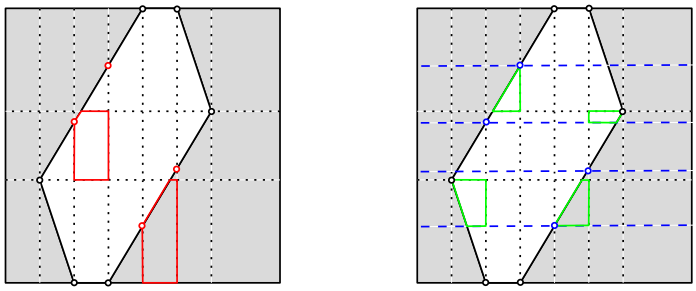


Fig. 5: Refining a cell $\mathcal{C}'$ in $\mathcal{D}_\varepsilon(f, g)$. The left shows $\mathcal{C}'$ after extending the horizontal and vertical lines through each vertex of the white polygon. The red pentagons (one in free space and one in infeasible space) result from these lines intersecting edges of the white polygon. We split these pentagons into two propagatable polygons by drawing the horizontal blue lines from those intersections (blue points). The resulting green polygons on the right are all propagatable.

All of the new polygons (in both the free and infeasible regions), except the pentagons, share a property that we call *propagatability*.[6] A propagatable polygon is a polygon with at most four edges, at least two of which are horizontal or vertical. For each (non-propagatable) pentagon we add an additional split by finding a vertex v that does not lie on a horizontal line, and then extend a horizontal line through v to the boundaries of $\mathcal{C}'$. After this split, even the pentagons are now split into propagatable polygons. The arrangement of all horizontal and vertical lines subdivides $\mathcal{C}'$ into a set of *refined cells*, and every free-space polygon inside a refined cell is propagatable, see Fig. 6:

Observation 1 (Refined Cells Contain Propagatable Polygons) *The free-space polygons inside refined cells are propagatable. The set of possible configurations of the refined cells are:*

[6] We use the term *propagatable*, as these regions allow for easier propagation of a score function in the dynamic program, which we elaborate on in the next sub-sections.

a) All free. b) Free space on right; dividing line has positive slope. c) Free space on left; dividing line has positive slope. d) Free space on right; dividing line has negative slope. e) Free space on left; dividing line has negative slope. f) Two components of infeasible space; dividing lines have positive slopes. g) Two components of infeasible space; dividing lines have negative slopes. h) All infeasible.

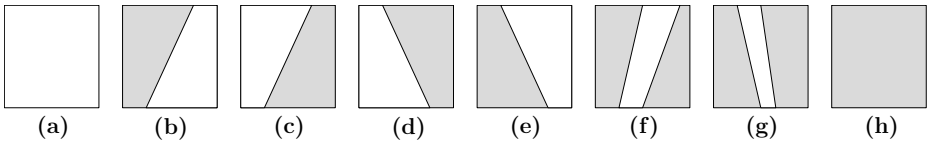


Fig. 6: All possible configurations of a cell in the refined free-space diagram.

Since there are a constant number of dividing lines per cell, each diagram cell contains a constant number of refined cells, which yields:

Lemma 1 (Cell Complexity). *Given two polygonal curves f and g with m and n segments respectively, the total number of refined cells in $\mathcal{D}_\varepsilon(f,g)$ is $\Theta(nm)$.*

4.2 Score Function in a Cell

To compute LSFS using dynamic programming in $\mathcal{D}_\varepsilon(f,g)$, we define a score function that maps $(x,y) \in \mathcal{D}_\varepsilon(f,g)$ to the LSFS defined for the corresponding prefixes of f and g. The score function $\mathcal{S} : [0, L_f] \times [0, L_g] \to \mathbb{R}$ is defined as

$$\mathcal{S}(x,y) = \mathbb{F}_\varepsilon(f|_{[0,x]}, g|_{[0,y]}) = \sup_{h:[0,x]\to[0,y]} \operatorname{len}(I_h^\varepsilon)\ . \tag{2}$$

The dynamic program in Section 4.5 computes the score function on the cell boundaries, by propagating from the left and bottom of a refined cell to the top and right of the cell. Let $\mathcal{C} = [x_L, x_L + x_R] \times [y_B, y_B + y_T] \subseteq [0, L_f] \times [0, L_g]$ be a refined cell, and denote with L, R, B, T the left, right, bottom, and top boundaries of $\mathcal{C}$, respectively. For ease of exposition we represent $\mathcal{C}$ using the local coordinate system $[0, x_R] \times [0, y_T]$ and use the following notation (see Fig. 7). The score functions restricted to L, R, B, T are:

$$\begin{aligned} \mathcal{S}_L(l) &= \mathcal{S}(x_L, y_B + l) & \mathcal{S}_R(r) &= \mathcal{S}(x_R, y_B + r) \\ \mathcal{S}_B(b) &= \mathcal{S}(x_L + b, y_B) & \mathcal{S}_T(t) &= \mathcal{S}(x_L + t, y_T) \end{aligned}$$

Any bi-monotone path from $(0,0)$ in $\mathcal{D}_\varepsilon(f,g)$ to a point in $\mathcal{C}$ has to go through L or B. We can therefore express $\mathcal{S}_T$ and $\mathcal{S}_R$ as

$$\mathcal{S}_T(t) = \max(\mathcal{S}_{L\to T}(t), \mathcal{S}_{B\to T}(t)) \text{ and } \mathcal{S}_R(r) = \max(\mathcal{S}_{L\to R}(r), \mathcal{S}_{B\to R}(r))\ . \tag{3}$$

Here, $\mathcal{S}_{L\to T}$ models the propagation from L to T. It is a restriction of LSFS on T that considers only those homeomorphisms h that pass through L. Such a homeomorphism corresponds to a bi-monotone path in $\mathcal{D}_\varepsilon(f,g)$ that is comprised

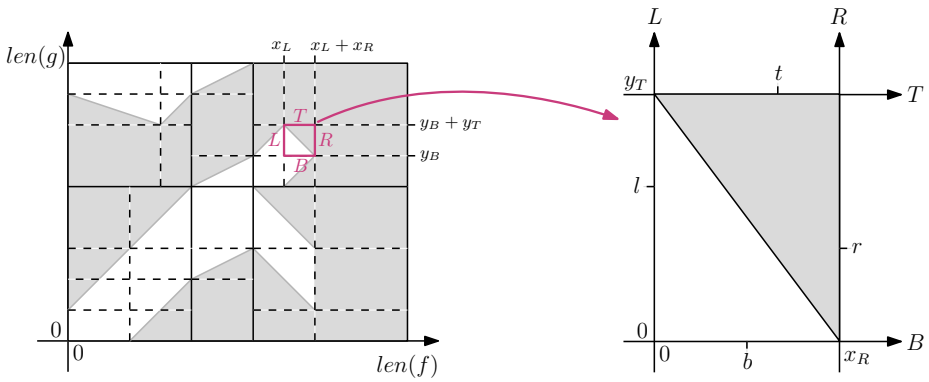


Fig. 7: A refined cell $\mathcal{C}$ shown in $\mathcal{D}_\varepsilon(f,g)$ (left) and in local coordinates (right).

of a path from $(0,0)$ to a point on L concatenated with a path from this point to a point on T. $\mathcal{S}_{B\to T}, \mathcal{S}_{L\to R}, \mathcal{S}_{B\to R}$ model the remaining types of propagations from L or B to T or R and are defined as:

$$\mathcal{S}_{L\to T}(t) = \sup_{l\in[0,y_T]} \mathcal{S}_L(l) + \mathcal{L}((0,l),(t,y_T)) \quad \mathcal{S}_{B\to T}(t) = \sup_{b\in[0,x_R]} \mathcal{S}_B(b) + \mathcal{L}((b,0),(t,y_T))$$
$$\mathcal{S}_{L\to R}(r) = \sup_{l\in[0,y_T]} \mathcal{S}_L(l) + \mathcal{L}((0,l),(x_R,r)) \quad \mathcal{S}_{B\to R}(r) = \sup_{b\in[0,x_R]} \mathcal{S}_B(b) + \mathcal{L}((b,0),(x_R,r)) \ ,$$

where $\mathcal{L}((x_1,y_1),(x_2,y_2)) = \mathbb{F}_\varepsilon(f|_{[x_L+x_1,x_L+x_2]}, g|_{[y_B+y_1,y_B+y_2]})$ measures the LSFS between points $(x_1,y_1),(x_2,y_2)$ in the local coordinate system of $\mathcal{C}$. Note that a bi-monotone path from (x_1,y_1) to (x_2,y_2) is not necessarily unique, however $\mathcal{L}((x_1,y_1),(x_2,y_2))$ is, see Fig. 8.

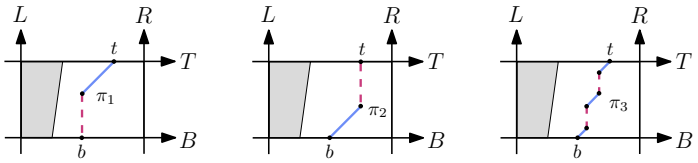


Fig. 8: π_1,π_2 and π_3 are different monotone paths from $(b,0)$ to $(t, y_B + y_T)$ that have the same slope-one length, $\mathcal{L}((b,0),(t,y_B+y_T))$ (blue). Other edges in these paths are horizontal or vertical.

In the remainder, we extensively use the following observation:

Observation 2 *Let (x,y) and $(x+\Delta x, y+\Delta y)$ be points in $\mathcal{D}_\varepsilon(f,g)$. Then the maximum length-preserving portion that can be achieved between those two points is* $\min(\Delta x, \Delta y)$. *Therefore* $\mathcal{L}((x,y),(x+\Delta x, y+\Delta y)) \le \min(\Delta x, \Delta y)$.

4.3 Properties of the Score Function

To compute the score function restricted to the top and right boundaries $\mathcal{S}_T$ and $\mathcal{S}_R$ of a refined cell $\mathcal{C}$, one has to take the maximum of $\mathcal{S}_{B\to T}$ and $\mathcal{S}_{L\to T}$ for

a point (t, y_T) on T or $\mathcal{S}_{B\to R}$ and $\mathcal{S}_{L\to R}$ for (x_R, r) on R. We show finding such maximum is a simple operation using the following observation and lemmas:

Observation 3 (Optimal Substructure) *Let $(x, y), (x', y') \in \mathcal{D}_\varepsilon(f, g)$. If (x', y') is on an optimal path from bottom left of $\mathcal{D}_\varepsilon(f, g)$ to (x, y), then $\mathcal{S}(x, y) \leq \mathcal{S}(x', y') + \min(y - y', x - x')$*

Lemma 2 (Single Breakpoint). *For all refined cells there is a point (x_0, y_0) on the top or right boundary such that one of the following holds:*

1. *if $(x_0, y_0) \in T$:*

$$\mathcal{S}_T(x) = \begin{cases} \mathcal{S}_{L\to T}(x) \text{ for } x < x_0 \\ \mathcal{S}_{B\to T}(x) \text{ for } x \geq x_0 \end{cases} \text{, and } \mathcal{S}_R(y) = S_{B\to R}(y) \tag{4}$$

2. *if $(x_0, y_0) \in R$:*

$$\mathcal{S}_T(x) = S_{L\to T}(x) \text{ , and } \mathcal{S}_R(y) = \begin{cases} \mathcal{S}_{L\to R}(y) \text{ for } y > y_0 \\ \mathcal{S}_{B\to R}(y) \text{ for } y \leq y_0 \end{cases} \tag{5}$$

Lemma 3 (Slope Upper-Bound). *The score function on a refined cell boundary is piecewise lienar, with each piece of slope less than or equal to 1.*

We provide proofs for Lemmas 2 and 3 in Appendix B.

4.4 Score Function Propagation Within a Cell

In this section, we demonstrate how to compute the score function from a cell boundary to another. In particular, we seek the functions $\mathcal{S}_{L\to R}$, $\mathcal{S}_{B\to R}$, $\mathcal{S}_{L\to T}$, and $\mathcal{S}_{B\to T}$ for all cases of the refined cells. To compute such functions, finding the maximum slope-one length that can be gained in a cell is essential. Since the free polygons in all cases are propagatable, such a slope-one line segment has to intersect one of the vertices of the free space. In the majority of the cases we determine the optimal path using Observation 3.

Case (a): We explore the propagation from L to R and B to R. Since going from L and B to T are symmetrical to these, one can compute them in a similar manner. For every point (x_R, r) on R there is a path with maximum slope-one length that reaches it through L. We aim to find corresponding points $(0, l)$ on L for every point (x_R, r) on R such that $\mathcal{S}_{L\to R}(r) = \mathcal{S}_L(l) + \mathcal{L}((0, l), (x_R, r))$ is maximal. $\mathcal{S}_L(l)$ is a non-decreasing piecewise linear function and all the pieces have slopes less than or equal to one. Therefore, finding $(0, l)$ and (x_R, r) that result in a maximal $\mathcal{L}((0, l), (x_R, r))$ is crucial.

Consider cell $\mathcal{C}$ in Fig. 9, since $\mathcal{C}$ is free, $\mathcal{L}((0, l), (x_R, r)) = \min(y_T, x_R)$ and this value can be achieved by drawing slope-one line segment from $(0, 0)$. Any point (x_R, r_1) on R below this line segment can be reached from $(0, 0)$ on L while $\mathcal{L}((0, 0), (x_R, r_1)) = r_1$. Any point (x_R, r_2) above this line segment on R can be reached from $(0, r_2 - x_R)$ while achieving the maximum slope-one length.

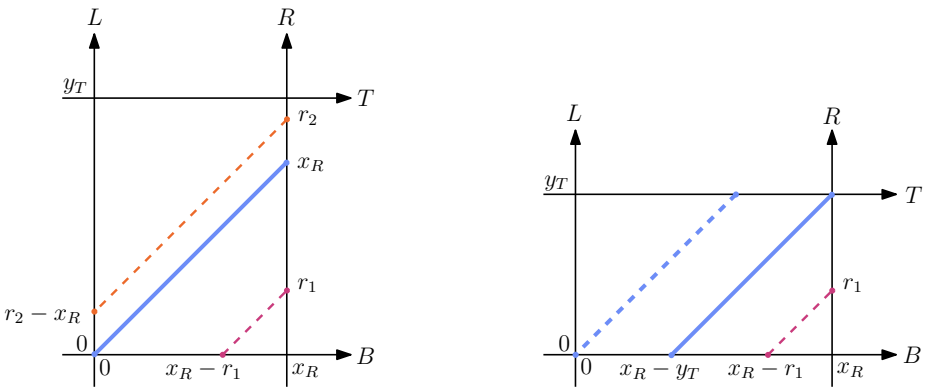


Fig. 9: Case (a), when $x_R < y_T$ on the left, and when $x_R > y_T$ on the right. The blue line shows the maximum slope-one length possible. Maximal slope-one segments to r_1 and r_2 are shown in magenta and orange, respectively.

Note that if $x_R > y_T$, a slope-one line through $(0, 0)$ does not intersect with R within $\mathcal{C}$ hence there is technically no point on R above such a line.

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le x_R \\ \mathcal{S}_L(r - x_R) + x_R & \text{for } r \ge x_R \end{cases}$$

For constructing $\mathcal{S}_{B\to R}(r)$, note that $\mathcal{L}((b, 0), (x_R, r))$ is maximal when a slope-one line segment going through (x_R, y_T) intersects B. If such line does not exist the maximal slope-one length is achieved by drawing a slope-one line from $(0, 0)$. For any point (x_R, r_1) below this line we draw a slope-one line segment from (x_R, r_1) that cuts B on $(x_R - r_1, 0)$. For any point (x_R, r_2) above such a line (if only $x_R < y_T$), the optimal path from B starts at $(0, 0)$.

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B(x_R - r) + r & \text{for } r \le x_R \\ \mathcal{S}_B(0) + x_R & \text{for } r \ge x_R \end{cases}$$

Theorem 1. *The propagation within Case (a) adds $O(1)$ breakpoints/complexity.*

Proof. Assuming $\mathcal{S}_L$ has n breakpoints, we want to determine the number of breakpoints on $\mathcal{S}_{L\to R}$, which consists of two pieces. In the first piece, $\mathcal{S}_L(l)$ is added to an increasing linear function r so the resulting function has at most n breakpoints. The second piece of $\mathcal{S}_{L\to R}(r)$ is a constant function without breakpoints. Thus, $\mathcal{S}_{L\to R}(r)$ has at most $n+1$ breakpoints. Similarly, $\mathcal{S}_{B\to R}(r)$, $\mathcal{S}_{L\to T}(t)$, and $\mathcal{S}_{B\to T}(t)$ each also have at most $n + 1$ breakpoints.

Case (c)–Case (h): We illustrate the rest of the cases in details in Appendix C.

4.5 Dynamic Programming Algorithm

The length-sensitive Fréchet similarity between two polygonal curves f and g for a given threshold ε can be computed using a dynamic programming (DP)

algorithm on the corresponding free-space diagram. Let m and n be the number of edges in f and g, respectively. As a pre-processing step, we refine the cells of the free-space diagram using the method in Section 4.1. Then, we process diagram cells row by row starting from the bottom left. The refined cells also have a local order within their diagram cell that is row by row, starting from the bottom left refined cell. The following operations are performed on every diagram cell $\mathcal{C}'$ in the DP algorithm:

1. **Initialization** (of the input functions): The score functions $S_L^{\mathcal{C}'}, S_B^{\mathcal{C}'}$ on the left and bottom boundaries of $\mathcal{C}'$ are divided and assigned to the corresponding score functions $S_L^{\mathcal{C}}, S_B^{\mathcal{C}}$ on those refined cells $\mathcal{C}$ that lie on the left and bottom boundary of $\mathcal{C}'$.
2. **Propagation:** For each refined cell $\mathcal{C}$, in bottom up order within $\mathcal{C}'$, we:
   (a) Compute $\mathcal{S}_{L\to R}^{\mathcal{C}}$, $\mathcal{S}_{B\to R}^{\mathcal{C}}$, $\mathcal{S}_{B\to T}^{\mathcal{C}}$, and $\mathcal{S}_{L\to T}^{\mathcal{C}}$ using the methods in Section 4.4 and Appendix C
   (b) Compute $\mathcal{S}_R^{\mathcal{C}}$ and $\mathcal{S}_T^{\mathcal{C}}$ based on Equation (3).
3. **Concatenation:** After all refined cells in the current diagram cell $\mathcal{C}'$ have been processed, we find the score function on the right boundary $S_R^{\mathcal{C}'}$ by concatenating the corresponding $\mathcal{S}_R$ functions on the refined cells. Similarly, we can compute the score function on the top boundary of the diagram cell.

As the DP algorithm finishes processing the last diagram cell, the $LSFS$ between f and g can be found in the top right corner of the free-space diagram.

Theorem 2. *The Length-Sensitive Fréchet similarity between two curves f and g with m and n pieces, respectively, can be computed in $O(m^2n^2)$.*

Proof. The DP algorithm is nested and processes all refined cells within all diagram cells. The total number of refined cells $O(mn)$; see Lemma 1. The initialization and concatenation steps take time linear in the complexity of the involved score functions. As demonstrated in Theorems 1-10, computing $\mathcal{S}_{L\to R}^{\mathcal{C}}$, $\mathcal{S}_{B\to R}^{\mathcal{C}}$, $\mathcal{S}_{B\to T}^{\mathcal{C}}$, and $\mathcal{S}_{L\to T}^{\mathcal{C}}$ in each cell adds 1 breakpoint to $S_L^{\mathcal{C}}$ and $S_B^{\mathcal{C}}$. Constructing $S_R^{\mathcal{C}}$ and $S_T^{\mathcal{C}}$ adds at most 1 breakpoint to one of these functions according to Lemma 2. Thus, the number of breakpoints in cell $\mathcal{D}[i][j]$ is in $O(ij)$, i.e., linear in the number of previous cells. Hence, the complexity of the score functions on the top right cell is $O(mn)$ and the total runtime of the DP is $O(m^2n^2)$.

5 Hardness of LSFS for Graphs

Unfortunately, deciding whether an optimal Length-Sensitive Fréchet similarity measure is above a given threshold is NP-hard.

Theorem 3 (Maximum LSFS is NP-hard). *Deciding if $\mathbb{F}_\varepsilon(G, H) > L$ is NP-hard, even if G consists of only one edge and H is a plane graph.*

Proof. We reduce from the Hamiltonian path problem in grid graphs, which is known to be NP-hard [17], even for induced grid graphs of degree at most

three [20]. Let $H' = (V', E')$ be a grid graph; that is, the vertex set is a finite subset of $\mathbb{Z}^2$ and there is an edge between two vertices u, v if and only if $\|u - v\| = 1$. We construct the graph H as follows: for every vertex, we add an edge to a new degree-one vertex at distance > 1; see Fig. 10. Formally, let $V'' = V' + (3/4, 3/4)$ be the set V' translated by $(3/4, 3/4)$ and $E'' = \{(v', v'') \in V' \times V'' \mid v'' = v' + (3/4, 3/4)\}$. The edges in E'' have length $\sqrt{2} \cdot 3/4 \approx 1.06$. We choose $H = (V' \cup V'', E' \cup E'')$. Without loss of generality, we assume that the coordinates of the vertices of H are between 0 and $n = |V'| > 1$, since we assume that H' is connected. Let G consist of only one edge $(0, n)$. We choose $\varepsilon = n$. We claim that if H' has a Hamiltonian path then $\mathbb{F}_\varepsilon(G, H) = n + 1$, and otherwise $\mathbb{F}_\varepsilon(G, H) < n + 1/5$, as desired.

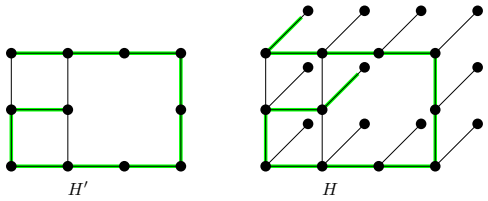


Fig. 10: A grid graph H' with Hamiltonian path in green. The graph H and the image of G corresponding to the Hamiltonian path.

We have chosen ε sufficiently large such that h (defined in Section 3.2) maps G onto any simple path in H (not necessarily ending at vertices). If H' has a Hamiltonian path, then we map G length-preserving onto the corresponding path in H extended by parts (of length 1) of edges in E'' at the beginning and end. Thus, $\mathbb{F}_\varepsilon(G, H)$ is the length of the edge $(0, n)$, i.e, $n + 1$. If H' does not have a Hamiltonian path, then the longest simple path in H' that starts and ends at vertices has length at most $n - 2$. Thus, the longest simple path in H has length at most $n - 2 + 2\sqrt{2} \cdot 3/4 \approx n + 0.12 < n + 1/5$.

6 Conclusions and Discussion

We defined length-sensitive Fréchet similarity as a natural partial similarity measure for geometric graphs and curves in $\mathbb{R}^d$. We presented an efficient algorithm for computing it under the L_1 norm for curves, and showed that the corresponding decision problem for geometric graphs is NP-hard. However, there are several directions that can be explored in this area. Can our similarity measure be transformed into a distance measure that is a metric?

For curves in $\mathbb{R}^d$, we conjecture that the running time in Theorem 2 might not be optimal. In [11], a faster running time is achieved by making use of the fact that the score functions are piecewise concave, which is not the case for our functions. For geometric graphs, finding an approximation algorithm to compute LSFS could lead to a practical similarity measure between road networks. A first step might be to consider restricted graph classes. A natural next question is whether there is a polynomial time algorithm for trees.

References

1. Aguilar, J., Buchin, K., Buchin, M., Hosseini Sereshgi, E., I. Silveira, R., Wenk, C.: Graph sampling for map comparison. In: 3rd ACM SIGSPATIAL International Workshop on Spatial Gems (2021)
2. Ahmed, M., Fasy, B.T., Hickmann, K.S., Wenk, C.: Path-based distance for street map comparison. ACM Transactions on Spatial Algorithms and Systems (28 pages, 2015)
3. Ahmed, M., Fasy, B.T., Wenk, C.: Local persistent homology based distance between maps. In: Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 43–52. ACM (2014)
4. Ahmed, M., Karagiorgou, S., Pfoser, D., Wenk, C.: Map Construction Algorithms. Springer (2015)
5. Akitaya, H.A., Buchin, M., Kilgus, B., Sijben, S., Wenk, C.: Distance measures for embedded graphs. Computational Geometry: Theory and Applications **95**(101743) (2021)
6. Alt, H., Efrat, A., Rote, G., Wenk, C.: Matching planar maps. Journal of Algorithms **49**(2), 262–283 (nov 2003). https://doi.org/10.1016/s0196-6774(03)00085-3
7. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. IJCGA **5**(1–2), 75–91 (1995)
8. Biagioni, J., Eriksson, J.: Inferring road maps from global positioning system traces. Transportation Research Record: Journal of the Transportation Research Board **2291**(1), 61–71 (2012)
9. Buchin, K., Buchin, M., Gudmundsson, J., Hendriks, J., Sereshgi, E.H., Sacristan, V., Silveira, R., Staals, F., Wenk, C.: Improved map construction using subtrajectory clustering. In: Proc. 4th ACM SIGSPATIAL LocalRec Workshop. pp. 5:1–5:4 (2020), https://doi.org/10.1145/3423334.3431451
10. Buchin, K., Buchin, M., Gudmundsson, J.: Constrained free space diagrams: a tool for trajectory analysis. International Journal of Geographical Information Science **24**(7), 1101–1125 (2010). https://doi.org/10.1080/13658810903569598
11. Buchin, K., Buchin, M., Wang, Y.: Exact algorithms for partial curve matching via the Fréchet distance. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms. p. 645–654. SODA '09, Society for Industrial and Applied Mathematics, USA (2009)
12. Chen, C., Lu, C., Huang, Q., Yang, Q., Gunopulos, D., Guibas, L.: City-scale map creation and updating using gps collections. In: Proc./ 22nd ACM SIGKDD. p. 1465–1474 (2016). https://doi.org/10.1145/2939672.2939833
13. Cheong, O., Gudmundsson, J., Kim, H.S., Schymura, D., Stehn, F.: Measuring the similarity of geometric graphs. In: International Symposium on Experimental Algorithms. pp. 101–112. Springer (2009)
14. De Carufel, J.L., Gheibi, A., Maheshwari, A., Sack, J.R., Scheffer, C.: Similarity of polygonal curves in the presence of outliers. Computational Geometry **47**(5), 625–641 (2014). https://doi.org/10.1016/j.comgeo.2014.01.002
15. Duran, D., Sacristán, V., Silveira, R.: Map construction algorithms: a local evaluation through hiking data. GeoInformatica **24**, 633–681 (2020)
16. He, S., Bastani, F., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S.: Roadrunner: improving the precision of road network inference from GPS trajectories. In: Proc. 26th ACM SIGSPATIAL GIS. pp. 3–12 (2018)
17. Itai, A., Papadimitriou, C.H., Szwarcfiter, J.L.: Hamilton paths in grid graphs. SIAM Journal on Computing **11**(4), 676–686 (1982)

18. Koide, S., Xiao, C., Ishikawa, Y.: Fast subtrajectory similarity search in road networks under weighted edit distance constraints. Proc. VLDB Endow. **13**(12), 2188–2201 (sep 2020). https://doi.org/10.14778/3407790.3407818
19. Maheshwari, A., Sack, J.R., Shahbaz, K., Zarrabi-Zadeh, H.: Fréchet distance with speed limits. Computational Geometry **44**(2), 110–120 (2011). https://doi.org/10.1016/j.comgeo.2010.09.008, special issue of selected papers from the 21st Annual Canadian Conference on Computational Geometry
20. Papadimitriou, C.H., Vazirani, U.V.: On two geometric problems related to the travelling salesman problem. Journal of Algorithms **5**(2), 231–246 (1984)

A LSFS is Length-Preserving on Connected Components

By definition, C_h^ε only requires the restriction of h to small balls to be length-preserving, we show that indeed all connected components are length-preserving.

Lemma 4 (Path-Connected Components Are Length-Preserving). *Each restriction of h to the preimage of a path-connected component of C_h^ε is a length-preserving map.*

Proof. Let $\widetilde{C}$ be a path-connected component of C_h^ε. Let $x, y \in \widetilde{C}$. Since $\widetilde{C}$ is path-connected, let $\gamma\colon [0,1] \to \widetilde{C}$ be a path that starts at x and ends at y. Then, by the definition of C_h^ϵ, for each $t \in [0,1]$, there exists a $\delta_t > 0$ such that h restricted to $B_t := \mathbb{B}_p(h^{-1}(\gamma(t)), \delta_t)$ is length-preserving. Let $\mathcal{U} := \{\gamma(B_t)\}_{t\in[0,1]}$. Since $\mathrm{Im}(\gamma)$ is a compact subspace of $\mathbb{R}^2$ and since $\mathcal{U}$ covers $\mathrm{Im}(\gamma)$, there exists a finite sub-cover $\widehat{\mathcal{U}}$ of $\mathcal{U}$. Then, there exists a decomposition of γ into sub-paths such that each sub-path lies entirely in at least one open set in $\widehat{\mathcal{U}}$. Let $\{\gamma_i\}_{i=1}^n$ be one such decomposition. Then, we know that $\mathrm{len}(\gamma) = \sum_i \mathrm{len}(p_i)$.

Since each $\mathrm{Im}\, p_i$ is contained in some $U \in \widehat{\mathcal{U}}$ and since h restricted to U is length-preserving, we know that $\mathrm{len}(\gamma_i) = \mathrm{len}(h(\gamma_i))$. Taking the sum over all sub-paths, we find $\mathrm{len}(\gamma) = \mathrm{len}(h(\gamma))$. Thus, we have shown that h restricted to $\widetilde{C}$ is length-preserving.

Noting that a curve $f\colon [0, L_f] \to \mathbb{R}^d$ is simply a directed graph in $\mathbb{R}^d$, we obtain:

Corollary 1. *Each restriction of h to the preimage of a path-connected component of H^ε is a length-preserving map.*

B Proof of Lemmas 2 and 3

In this section we discuss the proofs for Lemmas 2 and 3.

B.1 Proof of Lemma 2

Proof. First we show that for any two points (x_1, y_1) and (x_2, y_2) on the top or right boundary, there exist optimal paths to (x_1, y_1) and (x_2, y_2) that might

overlap but do not cross each other. Assuming that the optimal paths to (x_1, y_1) and (x_2, y_2) cross at a point a_0, there is another optimal path to (x_1, y_1) that goes through a_0 and overlaps with the optimal path to (x_2, y_2) before reaching a_0 (as shown in Fig. 11b).

(a) Optimal paths intersect transversally. (b) Optimal paths have initial overlap.

Fig. 11: A refined cell $\mathcal{C}$ in the free-space diagram $\mathcal{D}_\varepsilon(f, g)$. In each sub-figure, we see two optimal paths to $(x_1, y_1) \in T$ and to $(x_2, y_2) \in R$. (a) The two paths intersect transversally, at a_0. (b) The two paths first overlap, then separate at a_0.

Therefore, the optimal path to any point on the top or right boundary of a cell cuts the cell into two parts such that for all points on the top or right boundary in one part there is an optimal path going through them within that part. Considering the example in Fig. 11b with the purple path being an optimal path to (x_2, y_2), without loss of generality, we can say any point (x, y) such that $y > y_2$ has an optimal path that comes from the left boundary (more specifically, above the purple path) which means all points on the top boundary have an optimal path coming from the left boundary, hence (4) applies. Similarly, consider (x_1, y_1) and the orange path in Fig. 11a to be optimal. In this case, all the points (x, y) such that $x > x_1$ have an optimal path that comes from the bottom boundary (the right side of the orange path) which results in (5).

B.2 Proof of Lemma 3

Proof. Let $\mathcal{C}$ be a refined cell in $\mathcal{D}_\varepsilon(f, g)$. Let $(x_1, y_B + y_T), (x_2, y_B + y_T)$ be two points on T of $\mathcal{C}$. We want to show: $\mathcal{S}_T(x_2) - \mathcal{S}_T(x_1) = \mathcal{S}(x_2, y_B + y_T) - \mathcal{S}(x_1, y_B + y_T) \leq x_2 - x_1$. Consider the vertical line $\ell = \{(x, y) \in D \mid x = x_1\}$. The optimal monotone path from $(0, 0)$ to $(x_2, y_B + y_T)$ has to cross ℓ at a point (x_1, y_0). Note that (x_1, y_0) does not have to be inside $\mathcal{C}$, see Fig. 12. We know $\mathcal{S}(x_1, y_B + y_T) \geq \mathcal{S}(x_1, y_0)$ and from Observation 3 follows that $\mathcal{S}(x_2, y_B + y_T) - \mathcal{S}(x_1, y_0) \leq \min(y_B + y_T - y_0, x_2 - x_1)$. Thus we can conclude: $\mathcal{S}_T(x_2) - \mathcal{S}_T(x_1) = \mathcal{S}(x_2, y_B + y_T) - \mathcal{S}(x_1, y_B + y_T) \leq \min(y_B + y_T - y_0, x_2 - x_1) \leq x_2 - x_1$. Similarly we can show the equation above for other boundaries of $\mathcal{C}$ as well.

C Score Function Propagation Within a Cell, Continued

We discuss the remainder of the cases and score functions. Due to symmetric nature of Case (b),Case (c), Case (d), and Case (e) with regard to $\mathcal{S}_{L \to T}$ and

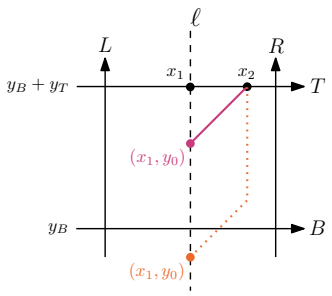


Fig. 12: A cell $\mathcal{C}$ in $\mathcal{D}_\varepsilon(f, g)$. (x_1, y_0) is on the optimal path to (x_2, y_T).

$\mathcal{S}_{B\to R}$ (likewise, for Case (f) and Case (g)), we do not explain $\mathcal{S}_{L\to T}$ for these cases to avoid repetition.

Case (b): Consider Fig. 13 and the cell $\mathcal{C}$ that is divided into two propagatable polygons by a line with a positive slope. The maximum value for $\mathcal{L}((0, l), (x_R, r))$ in this case is $x_R - j$ if $x_R < y_T$, and y_T if $x_R > y_T$. Such value can be achieved by drawing a slope-one line from $(j, 0)$ and if the divider's slope is less than one (see Fig. 13, right), by drawing a slope-one line from (x_R, y_T). For $\mathcal{S}_{L\to R}$, for any point (x_R, r_1) on R, below the maximum line, the optimal path goes from $(0, 0)$ to $(x_R - r_1, 0)$ horizontally and then to (x_R, r_1) via a slope-one segment (red dashed line).

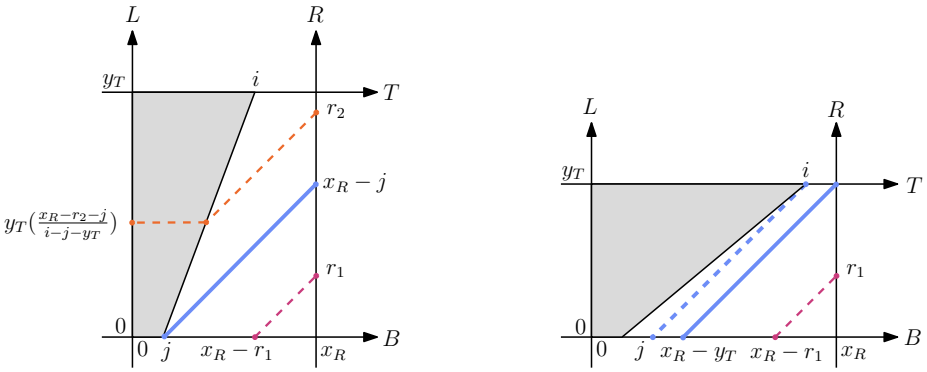


Fig. 13: Case (b) when $\frac{y_T}{i-j} > 1$ on the left, and when $\frac{y_T}{i-j} < 1$ on the right. The blue line shows the maximum slope-one length possible. Maximum slope-one segments to r_1 and r_2 are shown in magenta and orange, respectively.

For all points (x_R, r_2) above the maximum, the start point of the optimal path depends on $\mathcal{S}_L(l)$. In other words, taking the longest slope-one path to (x_R, r_2) in $\mathcal{C}$ does not necessary yield the maximum score for (x_R, r_2) in this case. This is due to the fact that $\mathcal{L}((0, l), (x_R, r))$ over l (and r) has negative

slope. In Theorem 4, we demonstrate how this decision is made and why the number of start-point candidates is at most the number of breakpoints on $\mathcal{S}_L(l)$.

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le x_R - j \\ \max\limits_{l\in[0,y_T(\frac{x_R-r-j}{i-j-y_T})]} \mathcal{S}_L(l) + \mathcal{L}((\frac{i-j}{y_T}(l+j), l), (x_R, r)) & \text{for } r \ge x_R - j \end{cases}$$

Propagation from B to R can be done similar to Case (a), yielding:

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B(x_R - r) + r & \text{for } r \le x_R - j \\ \mathcal{S}_B(j) + x_R - j & \text{for } r \ge x_R - j \end{cases}$$

$$\mathcal{S}_{B\to T}(t) = \begin{cases} \mathcal{S}_B(t) & \text{for } t \le j \\ \mathcal{S}_B(j) + (t - j) & \text{for } t \ge j \end{cases}$$

Theorem 4. *The propagation within Case (b) adds $O(1)$ complexity.*

Proof. Assuming $\mathcal{S}_L$ has n breakpoints, we want to determine the number of breakpoints on $\mathcal{S}_{L\to R}$, which consists of two pieces. The first piece, $\mathcal{S}_L(0) + r$ is a linear function. The second piece, is the upper-envelope of the summation of a non-decreasing piecewise linear and a decreasing linear function. See Fig. 14 for an example. To compute $\mathcal{S}_{L\to R}$, one can find the summation on the breakpoints of $\mathcal{S}_L$ and connect them, which results in at most n breakpoints. Taking the upper-envelope makes the resulting function non-decreasing and can only reduce the number of breakpoints. Therefore, there are at most $n+1$ breakpoints on $\mathcal{S}_{L\to R}$, and the complexity of $\mathcal{S}_{L\to R}$ is equal to the complexity of $\mathcal{S}_L$. The complexity for $\mathcal{S}_{B\to R}$, $\mathcal{S}_{B\to T}$, and $\mathcal{S}_{L\to T}$ can be shown similar to Theorem 1.

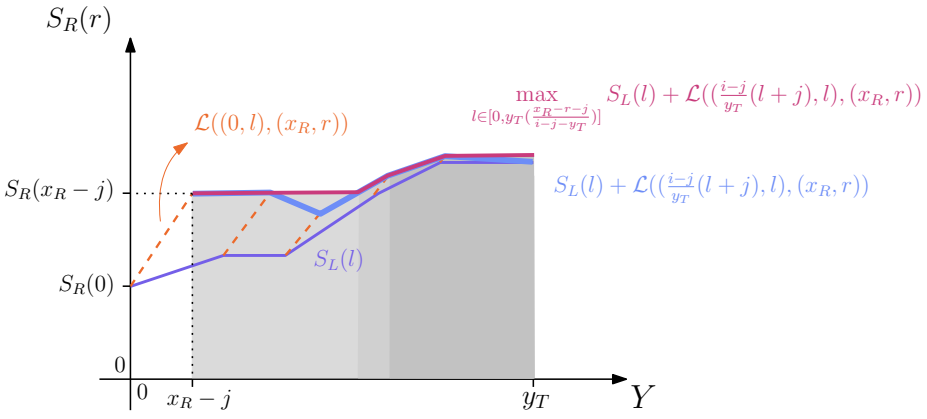


Fig. 14: The score function for $S_{L\to R}(r)$ when $r \ge x_R - j$.

Case (c): Case (c) represents a cell divided by a positive-slope line with free space on the left, therefore, the maximum length slope-one segment goes through (i, y_T), if the divider's slope is greater than one ($\frac{y_T}{i-j} > 1$). Otherwise the maximum segment intersects B at $(j, 0)$.

The divider's slope is greater than one: Consider Fig. 15, we first explore the paths from L to R. We draw a slope-one segment starting at $(0, 0)$ to find the score function's breakpoint. For any point below $(x_R, \frac{j}{\frac{y_T}{i-j}-1})$ the optimal path starts from $(0, 0)$. For all points (x_R, r) above the breakpoint, the optimal path starts from $(0, (1 - \frac{i-j}{y_T})r - j)$.

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le \frac{j}{\frac{y_T}{i-j}-1} \\ \mathcal{S}_L((1-\frac{i-j}{y_T})r - j) + \frac{i-j}{y_T}r + j & \text{for } r \ge \frac{j}{\frac{y_T}{i-j}-1} \end{cases}$$

Similarly for paths from B to R we have:

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B((\frac{i-j}{y_T} - 1)r + j) + r & \text{for } r \le \frac{j}{\frac{y_T}{i-j}-1} \\ \mathcal{S}_B(0) + \frac{i-j}{y_T}r + j & \text{for } r \ge \frac{j}{\frac{y_T}{i-j}-1} \end{cases}$$

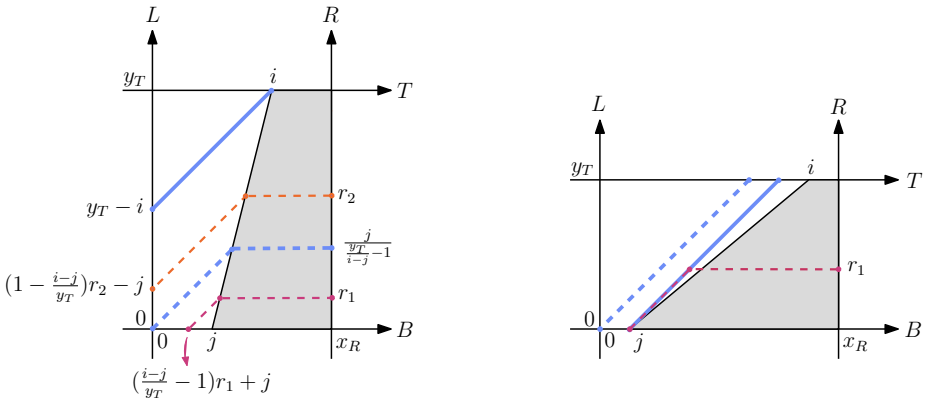


Fig. 15: Case (c), when $\frac{y_T}{i-j} > 1$ on the left, and when $\frac{y_T}{i-j} < 1$ on the right. The blue line shows the maximum slope-one length possible. Maximal slope-one segments to r_1 and r_2 are shown in magenta and orange, respectively.

$$\mathcal{S}_{B\to T}(t) = \begin{cases} \mathcal{S}_B(0) + t & \text{for } t \le i \\ \max(\mathcal{S}_B(0) + t, \mathcal{S}_B(t)) & \text{for } t \ge i \end{cases}$$

The divider's slope is less than one: Consider the right picture in Fig. 15. Since the maximum of $\mathcal{L}$ can be achieved from $(j, 0)$, we do not have a breakpoint in the score functions:

$$\mathcal{S}_{L\to R}(r) = \mathcal{S}_L(0) + r \quad \text{and} \quad \mathcal{S}_{B\to R}(r) = \mathcal{S}_B(j) + r$$

However for $\mathcal{S}_{B\to T}(t)$ we have:

$$\mathcal{S}_{B\to T}(t) = \begin{cases} \mathcal{S}_B(0) + t & \text{for } t \le y_T \\ \mathcal{S}_B(t - y_T) + y_T & \text{for } y_T \le t \le j + y_T \\ \max(\mathcal{S}_B(t - y_T) + y_T, \mathcal{S}_B(t)) & \text{for } t \ge j + y_T \end{cases}$$

Theorem 5. *The propagation within Case (c) adds $O(1)$ complexity.*

Proof. The $O(1)$ complexity follows in a similar way to Theorem 1. Note that for $\mathcal{S}_{B\to T}$ the max function adds at most one breakpoint.

Case (d):

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le \frac{j}{\frac{y_T}{i-j}-1} \\ \max\limits_{l\in[0,(1-\frac{i-j}{y_T})r-j]} \mathcal{S}_L(l) + \mathcal{L}((0,l),(\frac{i-j}{y_T}r + j, r)) & \text{for } r \ge \frac{j}{\frac{y_T}{i-j}-1} \end{cases}$$

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B((\frac{i-j}{y_T} - 1)r + j) + r & \text{for } r \le \frac{j}{\frac{y_T}{i-j}-1} \\ \mathcal{S}_B(0) + \frac{j}{\frac{y_T}{i-j}-1} & \text{for } r \ge \frac{j}{\frac{y_T}{i-j}-1} \end{cases}$$

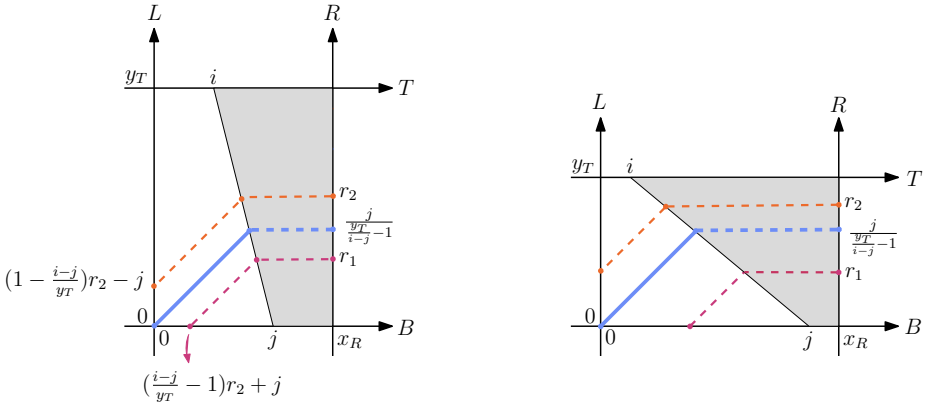


Fig. 16: Case (d), when $\frac{y_T}{i-j} < -1$ on the left, and when $\frac{y_T}{i-j} > -1$ on the right. The blue line shows the maximum slope-one length possible. Maximal slope-one segments to r_1 and r_2 are shown in magenta and orange, respectively.

Theorem 6. *The propagation within Case (d) adds $O(1)$ complexity.*

Proof. The $O(1)$ complexity can be discussed in a similar way to Theorem 4

Case (e):

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le x_R \\ \mathcal{S}_L(y_T(\frac{x_R - r - j}{i - j - y_T})) + r - y_T(\frac{x_R - r - j}{i - j - y_T}) & \text{for } r \ge x_R \end{cases}$$

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B(x_R - r) + r & \text{for } r \le x_R - j \\ \mathcal{S}_B(\frac{x_R - r - \frac{jy_T}{i-j}}{1 - \frac{y_T}{i-j}}) + x_R - \frac{x_R - r - \frac{jy_T}{i-j}}{1 - \frac{y_T}{i-j}} & \text{for } r \ge x_R - j \end{cases}$$

$$\mathcal{S}_{B\to T}(t) = \begin{cases} \mathcal{S}_B(t) & \text{for } r \le t \le i \\ \mathcal{S}_B(\frac{y_T}{i-j} + t - j) + t - \frac{y_T}{i-j} + t - j & \text{for } t \ge i \end{cases}$$

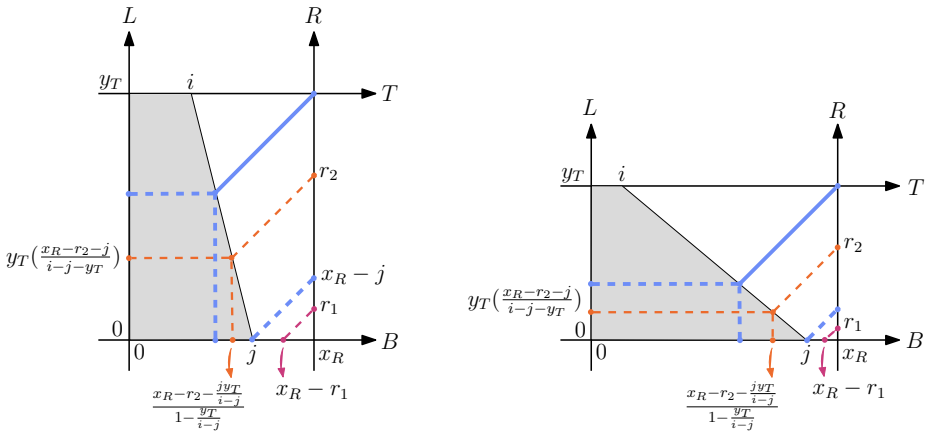


Fig. 17: Case (e), when $\frac{y_T}{i-j} < -1$ on the left, and when $\frac{y_T}{i-j} > -1$ on the right. The blue line shows the maximum slope-one length possible. Maximal slope-one segments to r_1 and r_2 are shown in magenta and orange, respectively.

Theorem 7. *The propagation within Case (e) adds $O(1)$ complexity.*

Proof. The $O(1)$ complexity can be discussed in a similar way to Theorem 1

Case (f): As mentioned in Case (c), when a divider has a positive slope and is located on the right side of a free space, extra care is required. Suppose a divider's slope is less than one, a slope-one segment can be drawn above it, starting at $(k, 0)$. This segment can be a part of the optimal path from B to R. Therefore, we have four possibilities for this case:

Both dividers' slopes are greater than one: When both slopes are greater than one, the optimal path between two points does not necessarily have at most three segments. Consider Fig. 18 (right) and point (x_R, r_2'). The longest slope-one path to (x_R, r_2') is shown in orange, and has multiple slope-one segments. By

Lemma 3, this path is always maximal when going through B. The same cannot be said for paths going through L in this case, since there are vertical segments in such a path. Therefore, for a point (x_R, r) above $(x_R, \frac{j+k}{1-\frac{w-k}{y_T}})$, $\mathcal{S}_{L\to R}(r)$ is the maximum over l of $\mathcal{S}_L(l)$ and the length of slope-one segments in the feasible space between l and r.

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le \frac{j+k}{1-\frac{w-k}{y_T}} \\ \max\limits_{l\in[0,\frac{(1-\frac{w-k}{y_T})r_2-k+j}{1-\frac{i-j}{y_T}}]} \mathcal{S}_L(l) + \mathcal{L}((\frac{(i-j)}{y_T}l + (i-j)j, l), (\frac{w-k}{y_T}r + k, r)) & \text{for } r \ge \frac{j+k}{1-\frac{w-k}{y_T}} \end{cases}$$

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B((\frac{i-j}{y_T} - 1)r + j) + r & \text{for } r \le \frac{j+k}{1-\frac{w-k}{y_T}} \\ \mathcal{S}_B(j) + \frac{w-k}{y_T}r + k - j & \text{for } r \ge \frac{j+k}{1-\frac{w-k}{y_T}} \end{cases}$$

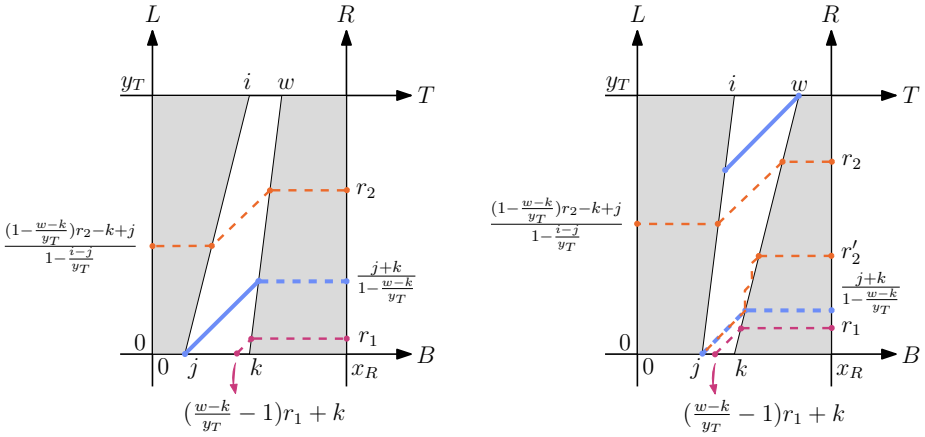


Fig. 18: Case (f) when both dividers have slope greater than one. The blue line shows the maximum slope-one length possible. Maximal slope-one segments to r_1, r_2 and r'_2 are shown in magenta and orange.

Theorem 8. *The propagation within Case (f) adds $O(1)$ complexity.*

The proof is omitted, as it is similar to Theorem 4 and Theorem 5.

Left divider's slope is greater than one: Since a divider can only have slope less than one if $x_R > y_T$, the maximum length slope-one segment that can obtained in this case is y_T. This amount can be achieved by simply drawing a slope-one line from $(K, 0)$ (see Fig. 19) because the right divider's slope is less than one. We have:

$$\mathcal{S}_{L\to R}(r) = \mathcal{S}_L(0) + r$$

$$\mathcal{S}_{B\to R}(r) = \mathcal{S}_B(k) + r$$

Right divider's slope is greater than one: Similar to the previous sub-case, the maximum length slope-one segment is achievable. However, since the right divider's slope is greater than one, the maximum can be gained by drawing a slope-one line from (w, y_T) (Fig. 19).

$$\mathcal{S}_{L\to R}(r) = \mathcal{S}_L(0) + r$$

$$\mathcal{S}_{B\to R}(r) = \mathcal{S}_B((\frac{w-k}{y_T} - 1)r + k) + r$$

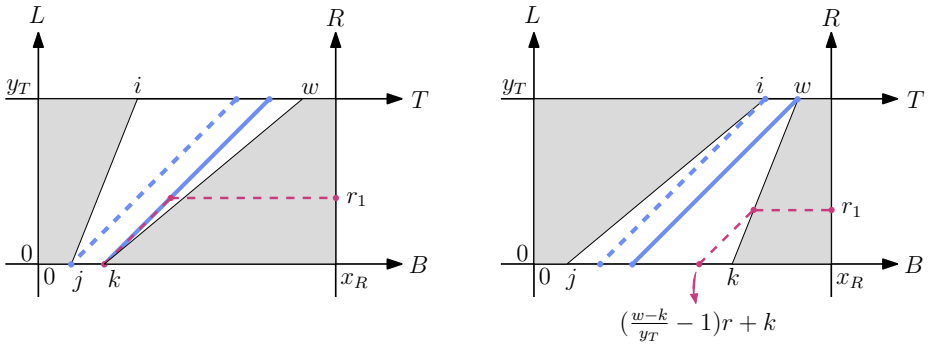


Fig. 19: Case (f) when one divider's slope is less than one and the other's is greater than one.

Both dividers' slopes are less than one:

$$\mathcal{S}_{L\to R}(x_R, r) = \mathcal{S}_L(0) + r$$

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B(k) + r & \text{for } r \le \frac{(i-j)(j-k)}{(i-j)-y_T} \\ \max\limits_{b\in[k,\frac{(w-k)((1-\frac{i-j}{y_T})r+ky_T-j(i-j))}{y_T-w+k}]} \mathcal{S}_B(b) + \mathcal{L}((b, \frac{y_T}{w-k}b - ky_T), (\frac{i-j}{y_T}r + j(i-j), r)) & \text{for } r \ge \frac{(i-j)(j-k)}{(i-j)-y_T} \end{cases}$$

Case (g): Case (g) consists of two dividers with negative slope. Naturally, there are three possibilities for this case:

Two parallel dividers (II):

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le \frac{j+k}{1-\frac{w-k}{y_T}} \\ \mathcal{S}_L(\frac{(1-\frac{w-k}{y_T})r-k+j}{1-\frac{i-j}{y_T}}) + \frac{(\frac{w-k-i+j}{y_T})r+k-j}{1-\frac{i-j}{y_T}} & \text{for } r \ge \frac{j+k}{1-\frac{w-k}{y_T}} \end{cases}$$

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B((\frac{w-k}{y_T})r + k) + r & \text{for } r \le \frac{j+k}{1-\frac{w-k}{y_T}} \\ \mathcal{S}_B(j) + \frac{j+k}{1-\frac{w-k}{y_T}} & \text{for } r \ge \frac{j+k}{1-\frac{w-k}{y_T}} \end{cases}$$

Theorem 9. *The propagation within Case (g) adds $O(1)$ complexity.*

Proof. The $O(1)$ complexity for sub-case II follows in a way similar to Theorem 1. Likewise, for V and Λ we can use the same approach as in Theorem 4.

V-shaped and Λ-shaped cases share some functions and properties with the parallel case II. We discuss their corresponding functions below:

The left divider's slope is greater than the right divider's (V): The score function on R coming from L is identical to the parallel case (II), since $\mathcal{L}((0,l),(x_R,r))$ is non-decreasing over l and r (see Fig. 20). However, $\mathcal{L}((0,b),(x_R,r))$ is decreasing over b, so for $\mathcal{S}_{B\to R}(r)$ we have:

$$\mathcal{S}_{B\to R}(r) = \begin{cases} \mathcal{S}_B(j) + r & \text{for } r \le \frac{j+k}{1-\frac{w-k}{y_T}} \\ \max\limits_{B\in[0,(\frac{w-k}{y_T}-1)r+k]} \mathcal{S}_B(b) + \mathcal{L}((b, \frac{y_T}{i-j}b - jy_T), (\frac{w-k}{y_T}r + k, r)) & \text{for } r \ge \frac{j+k}{1-\frac{w-k}{y_T}} \end{cases}$$

The right divider's slope is greater than the left divider's (Λ): The score function on R coming from B is identical to the parallel case (II) because $\mathcal{L}((0,b),(x_R,r))$ is non-decreasing over b.[7] However, that is not the case with $\mathcal{L}((0,l),(x_R,r))$:

$$\mathcal{S}_{L\to R}(r) = \begin{cases} \mathcal{S}_L(0) + r & \text{for } r \le \frac{j+k}{1-\frac{w-k}{y_T}} \\ \max\limits_{l\in[0,\frac{(1-\frac{w-k}{y_T})r-k+j}{1-\frac{i-j}{y_T}}]} \mathcal{S}_L(l) + \mathcal{L}((\frac{(i-j)}{y_T}l + (i-j)j, l), (\frac{w-k}{y_T}r + k, r)) & \text{for } r \ge \frac{j+k}{1-\frac{w-k}{y_T}} \end{cases}$$

Case (h) Here, the cell is filled with infeasible space; therefore, for any pair of points p_1 and p_2 on the boundaries, $\mathcal{L}(p_1,p_2) = 0$. Hence, going from L to R (or B to T) is simply: $\mathcal{S}_{L\to R}(r) = \mathcal{S}_L(r)$ (and $\mathcal{S}_{B\to T}(t) = \mathcal{S}_B(t)$). Since the score functions on the boundaries are non-decreasing (by Observation 3), we have: $\mathcal{S}_{B\to R}(r) = \mathcal{S}_B(x_R)$ (and $\mathcal{S}_{L\to T}(t) = \mathcal{S}_L(y_T)$).

[7] Note that in both sub-cases, if $\mathcal{S}_B(x_R) > \mathcal{S}_B(k) + \frac{j+k}{1-\frac{t-k}{y_T}}$, then $\mathcal{S}_R(r) = \mathcal{S}_B(x_R)$

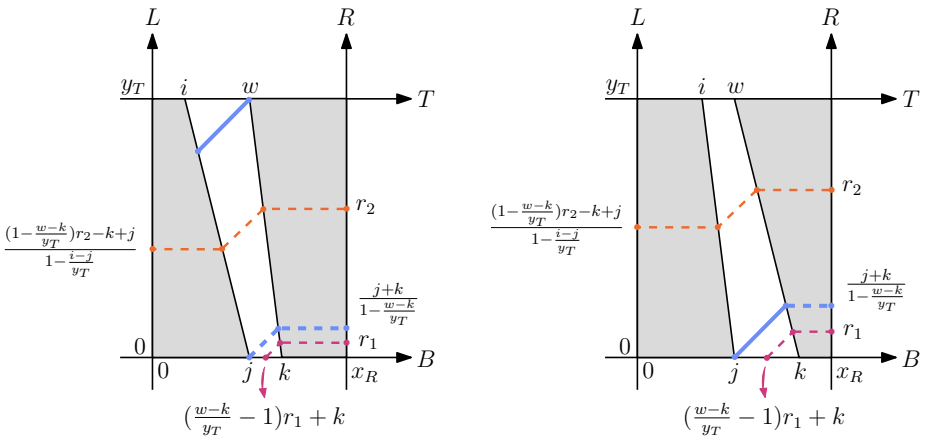


Fig. 20: Case (g) with V on the left and Λ on the right side. The blue line shows the maximum slope-one length possible. Maximal slope-one segments to r_1 and r_2 are shown in magenta and orange.

Theorem 10. *The propagation within Case (h) adds no complexity.*

Proof. Since $\mathcal{S}_{L\to R}(r) = \mathcal{S}_L(l)$ directly, there is no additional breaking point on $\mathcal{S}_{L\to R}$. Furthermore, $\mathcal{S}_{B\to R}$ is a constant function, which means it does not transfer any complexity from $\mathcal{S}_B$. We have similar explanations for $\mathcal{S}_{B\to T}$ and $\mathcal{S}_{L\to T}$.