INSTITUTE OF
COMPUTER SCIENCE
FREIE UNIVERSITÄT BERLIN

Bachelor Thesis

Lexicographic Fréchet Matchings with Degenerate Inputs

Anton Irmfried Norbert Begehr
Matriculation-Nr: 5013449
a.begehr@fu-berlin.de

Supervisor: Prof. Dr. Günter Rote

Berlin, July 11, 2018

Abstract

Ok.

Statutory Declaration

I declare that I have developed and written the enclosed Bachelors Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Bachelors Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Anton Begehr
Berlin, July 11, 2018

Contents

1 Introduction

1.1 Motivation

!!!TODO!!! compare two paths

1.2 Applications

!!!TODO!!! This is a short overview of real world applications of the Fréchet Distance.

1.2.1 Molecule Chains (Proteins)

1.2.2 Handwriting Recognition

1.2.3 Comparing Routes

1.2.4 Computer Vision

1.2.5 Wifi access point placing

1.3 Problem Metaphor (Dog Walking)

The Fréchet distance is often visually explained using the "dog walking"-metaphor, which is described in the following paragraphs.

Imagine two curves:

- Curve C_P tracing the walking curve of a man
- Curve C_Q tracing the walking curve of the man's dog

The Fréchet Distance then is the shortest possible length of a leash, the man would need to keep ahold of his dog. Both the man and his dog have to traverse their entire paths and neither are allowed to walk backwards.

Both the man and his dog can independently vary their respective speeds. With the conditions described above, at every point in time, either both the man and his dog are walking forwards along their respective paths or one is idle while the other is walking along his path.

To make calculating the Fréchet distance more feasible by simplification, the arbitrary curves C_P, on which the man is walking, and and C_Q, on which his dog is walking, are approximated each by a connected chain of line segments, a polygonal chain. The arbitrary curve C_P is approximated by the polygonal chain P. The arbitrary curve C_Q is approximated by the polygonal chain Q.

2 Preliminaries

2.1 Polygonal Chains

Given are two polygonal chains P and Q. A polygonal chain is defined by a list of points, sorted by the order of the chain, ascending from beginning to end. The first point of the list is the start of the polygonal curve and the last point of the list is the end of the polygonal curve. This order implies the direction of the directional polygonal curve.

P and Q are polygonal chains consisting of n and m points respectively.

$$P_{points} = [P_1, P_2, \dots, P_n]$$
$$Q_{points} = [Q_1, Q_2, \dots, Q_m]$$

Points of P and Q are expressed as vectors:

$$P_i = \begin{pmatrix} x_{Pi} \\ y_{Pi} \end{pmatrix}, Q_j = \begin{pmatrix} x_{Qj} \\ y_{Qj} \end{pmatrix}$$

For simplification we assume all points P_i of P and Q_j of Q lie in the xy-plane, but the results generalize easily to arbitrary dimensions[2].

2.2 Parametrisation

We will use an arc-length parametrisation to define the polygonal curves P and Q going forward. We are using the arc-length parametrisation, instead of an unit-length parameter, to achieve visualisations with proportional scaled cells in the further course of this examination.

First we calculate the euclidian distances between the points of P and Q in the order they appear:

$$d_{Pi} = \|P_{i+1} - P_i\|, 1 \leq i \leq n-1$$
$$d_{Qj} = \|Q_{j+1} - Q_j\|, 1 \leq j \leq m-1$$

We also calculate the distances L_{Pi} and L_{Qi}, that define the distance from beginning of P and Q to the point P_i and Q_j on the P and Q respectively:

$$L_{Pi} = \sum_{k=1}^{i} d_{Pk}, 1 \leq i \leq n-1$$

$$L_{Qi} = \sum_{k=1}^{j} d_{Qk}, 1 \leq j \leq m-1$$

The total lengths of P and Q are then defined as $L_P = L_{Pn-1}$ and $L_Q = L_{Qm-1}$.

We apply an arc-length parametrisation on P and Q using parameters $x \in [0, L_P]$ and $y \in [0, L_Q]$ respectively. These parametrisation can be expressed as follows:

$$P(x) = \begin{cases} P_1 + \frac{x}{d_{P1}}(P_2 - P_1) & 0 \le x \le L_{P1} \\ P_2 + \frac{x - L_{P1}}{d_{P2}}(P_3 - P_2) & L_{P1} < x \le L_{P2} \\ \dots \\ P_{n-2} + \frac{x - L_{Pn-2}}{d_{Pn-1}}(P_{n-1} - P_{n-2}) & L_{Pn-2} < x \le L_{Pn-1} \end{cases}$$

$$Q(y) = \begin{cases} Q_1 + \frac{y}{d_{Q1}}(Q_2 - Q_1) & 0 \le y \le L_{Q1} \\ Q_2 + \frac{y - L_{Q1}}{d_{Q2}}(Q_3 - Q_2) & L_{Q1} < x \le L_{Q2} \\ \dots \\ Q_{m-2} + \frac{x - L_{Qm-2}}{d_{Qm-1}}(Q_{m-1} - Q_{m-2}) & L_{Qm-2} < x \le L_{Qm-1} \end{cases}$$

2.3 Height Function

To calculate all possible distances between points on P and Q, we will define the height function δ that maps each point pair R consisting of one point on P and one point on Q to the euclidian distance between the points.

$$R = [0, L_P] \times [0, L_Q]$$
$$(x, y) \in R$$
$$\delta(x, y) := \|P(x) - Q(y)\|$$

3 Classical Fréchet Distance

This section introduces the Fréchet distance and gives an overview of the main findings in the 1995 paper "Computing the Fréchet distance between two polygonal curves" by Alt and Godau[1], which later will be essential for computing the lexicographic Fréchet distance.

3.1 Traversal of P and Q

Let us revisit the "dog walking"-metaphor (Section 1.3) of the man and his dog walking along, or in other words traversing, their paths. The Fréchet distance is a dissimilarity measure that considers the traversal of both polygonal chains P and Q with following conditions:

(A) Both P and Q are traversed completely.

(B) Both P and Q are traversed monotonically ascending in the direction of P and Q respectively. Meaning a traverser on either P or Q can stand idle and go forwards, but may never go backwards.

We traverse both P and Q together. This means that at every point in time one of the following cases holds true:

- Both the traverser on P and the traverser on Q are moving.
- The traverser on P is moving and the traverser on Q is standing idle.
- The traverser on Q is moving and the traverser on P is standing idle.
- Neither the traverser on P, nor the traverser on Q is moving.

The last case is identical to the end of both traversals. Both traverses have reached the end of their polygonal chains.

We define a traversal of both P and Q together as a joint parametrisation $(\alpha(t), \beta(t))$. This joint parametrisation represents a continuous curve in the parameter area R. The curve represented by $(\alpha(t), \beta(t))$ is a monotonic curve, because of above condition (B) which states that both P and Q are traversed only forwards and never backwards.

Revisiting the "dog walking"-metaphor from Section 1.3, the Fréchet distance is the length of the shortest leash, with which it is possible for the man and his dog to traverse their paths completely by only travelling forward. In other words, the Fréchet distance is defined as the largest distance between the traverser on path P and the traverser on path Q of the traversal with the smallest maximum distance between the respective traversers.

Using the joint parametrisation $(\alpha(t), \beta(t))$ as representative for a traversal, next the distance function $f(t)$ for the joint parametrisation needs to be defined to determine the Fréchet distance. The height function defined in Section 2.3 will be used to define the distance function $f(t)$ for a joint parametrisation $(\alpha(t), \beta(t))$:

$$f(t) = \delta(\alpha(t), \beta(t)) := \|P(\alpha(t)) - Q(\beta(t))\|$$

Now, the Fréchet distance of P and Q is the largest value of $f(t)$ for the joint parameterisations $(\alpha(t), \beta(t))$ with the smallest maximum of all possible joint parameterisations.

3.2 Free-Space Diagram

The free-space diagram F_ϵ is the set of all points of R that lie below the given height ϵ:

$$(x, y) \in R$$

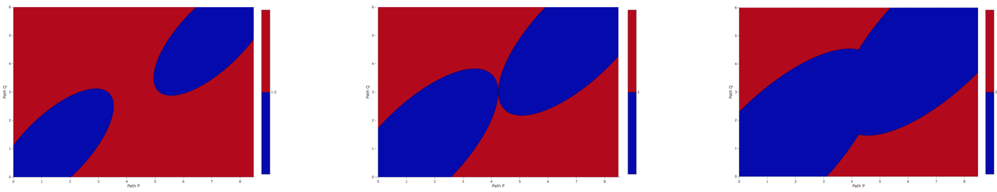
$$F_\epsilon = \{(x, y) \mid \|P(x) - Q(y)\| \leq \epsilon\}$$

The fundamental insight of Alt and Godau[1] is that the Fréchet distance of P and Q is the smallest ϵ for which a monotonic path exists from $(0, 0)$ to (L_P, L_Q) in F_ϵ.[2]

3.3 Decision Problem

The decision problem asks if a height function δ can be traversed monotonically while not exceeding the height of a given ϵ. In their paper, Alt and Godau[1] define the decision algorithm, which solves the decision problem by using the free-space diagram F_ϵ.

In the following example we will see three free-space diagrams F_ϵ for the same height function with different ϵ. ϵ_F denotes the Fréchet distance.



(a) $\epsilon < \epsilon_F$ (b) $\epsilon = \epsilon_F$ (c) $\epsilon > \epsilon_F$

Figure 1: Three free-space diagrams with same height function δ[1]

The following cases are depicted in Figure 1 above:

(a) ϵ is smaller than the Fréchet distance ϵ_F. The decision algorithm returns false.

[1]View the example here: `https://abegehr.github.io/frechet/?p=(1_2)(4_5)(7_2)&q=(1_3)(7_3)`

(b) ϵ is equal to the Fréchet distance ϵ_F. The decision algorithm returns true.

(c) ϵ is larger than the Fréchet distance ϵ_F. The decision algorithm returns true.

3.4 Classical Critical Events

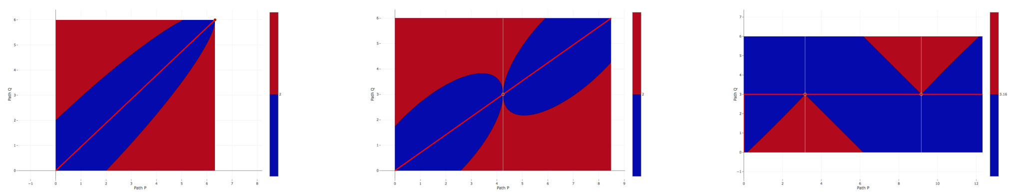
Alt and Godau also identified, that the critical height ϵ_F, where a critical pathway in the free-space diagram closes, occurs in one of three cases. These are the three types of classical critical events:

(a) ϵ_F at $(0, 0) \in F_\epsilon$ or $(L_P, L_Q) \in F_\epsilon$

(b) A new passage with height ϵ_F opens at border between two cells.

(c) A new horizontal or vertical passage with height ϵ_F opens on borders with cells in between.

In Figure 2 example free-space diagrams for each type of classical critical event are shown.



(a) type a[2] (b) type b[3] (c) type c[4]

Figure 2: Examples for the three types of classical critical events

Critical events will be denoted with C_k, where $C_k.A$ is the starting-point and $C_k.B$ is the ending-point. For classical critical events of types a and b the starting-point and ending-point are the same.

[2] Classical critical event of type a at height ϵ_F is at the upper-right corner (L_P, L_Q). View the example here: `https://abegehr.github.io/frechet/?p=(1_3)(7_5)&q=(1_3)(7_3)`

[3] Classical critical event of type b at height ϵ_F is at vertical border connecting the left to the right cell. View the example here: `https://abegehr.github.io/frechet/?p=(1_2)(4_5)(7_2)&q=(1_3)(7_3)`

[4] Classical critical event of type c at height ϵ_F is at vertical border connecting the left to the right cell over the middle cell. View the example here: `https://abegehr.github.io/frechet/?p=(4_3)(7_4)(1_4)(4_3)&q=(1_3)(7_3)`

3.5 Computing the Classical Fréchet Distance

We now have the tools to compute the classical Fréchet distance. To compute the classical Fréchet distance, we can apply *Algorithm 2* of Alt and Godau's paper.[1] The algorithm first determines all classical critical events of type a, b, and c, including their critical ϵ and then using a binary search and the decision algorithm, finds the smallest critical ϵ for which the height function δ is traversable.

4 Lexicographic Fréchet Matchings

In his 2014 paper "Lexicographic Fréchet Matchings", Rote extended upon the findings from Alt and Godau in their paper "Computing the Fréchet distance between two polygonal curves" to establish an algorithm that produces a lexicographic Fréchet matching.

Taking a lexicographic approach roughly means that we want to minimize the time $T(s)$ during which the height exceeds a threshold s.[2] We want to spend as little time as possible at the current height and want to descend as quickly as possible.

4.1 Steepest Descent

To achieve a lexicographically optimized traversal, Rote makes use of the steepest descent. Therefore he developed a steepest descent algorithm.

Because we chose our input to be two paths consisting of straight line segments, the level set in a cell consists of concentric ellipses. Rote defines two lines for every cell, l and l':

(l) The points where the ellipses have vertical tangents and δ has a linear horizontal gradient lie on line l through the common center.[2]

(l') The points where the ellipses have horizontal tangents and δ has a linear vertical gradient lie on line l through the common center.[2]

Depending on whether we are descending from a point on one of the lines or from in between one of the quadrants they span up, the direction of the steepest descent is affected. Consult Rote's paper "Lexicographic Fréchet Matchings"[2] (Figure 5) for more detail. It is important to note at this point, that the steepest descent for a point is unique.

4.2 New Lexicographic Type of Critical Event

Rote also identifies a new lexicographic type of critical event, which does not influence the global Fréchet distance ϵ_F, but is essential for achieving a lexicographically correct traversal.

The new lexicographic type of critical event occurs while descending on a steepest descent path. It occurs when we would horizontally or vertically surpass a critical opening of the height at which we currently are.

Figure 3 shows a minimal example of these new lexicographic types of critical events.

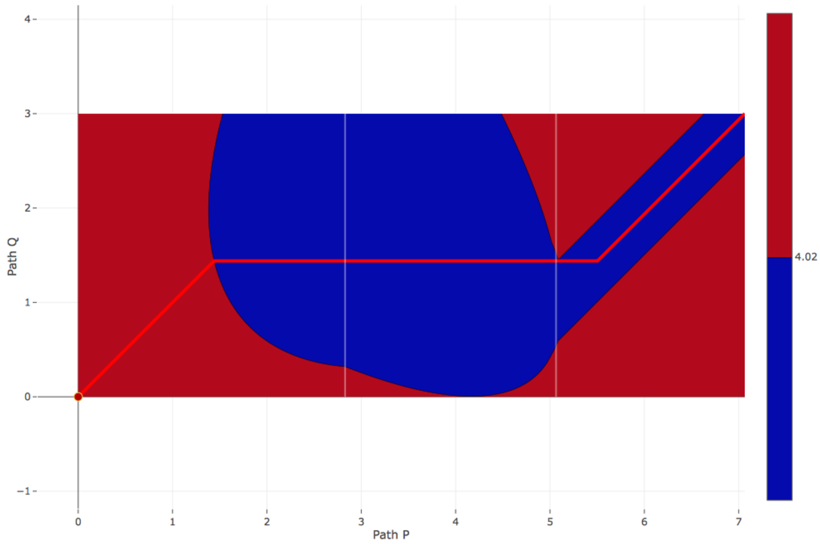


Figure 3: Example of new lexicographic type of critical event[5]

4.3 Lexicographic Fréchet Matching Algorithm

We now have the tools needed to understand Rote's algorithm for lexicographic Fréchet matchings[2] under *general inputs* (not yet regarding degenerate inputs, as described in Section 5). In contrary to the algorithm for computing the classical Fréchet distance, the algorithm for lexicographic Fréchet matching utilizes recursion to allow for a locally correct lexicographic traversal. The algorithm as described here is aligned to the implementation of the online visualization.

[5]New lexicographic critical event of type a below height ϵ_F connecting the middle of the left cell to the vertical border between the middle and the right cell. View the example here: `https://abegehr.github.io/frechet/?p=(3_2)(5_4)(3_3)(5_3)&q=(2_7)(5_7)`

Algorithm 1 Requirements for Lexicographic Fréchet Matching Algorithm

function DecisionAlgorithm(δ, A, B, ϵ)
    ▷ Implementation of the decision algorithm, that decides if δ is traversable from A to B with ϵ.
    return $true$ or $false$

function SteepestDescent($\delta, A, B, \epsilon_A, \epsilon_B$)
    ▷ Attempts a steepest descent as far as possible for the higher of A and B, or both if they reside at equal height. Attempt the steepest descent to A' and B' until A' or B' reach a cell border or a line l or l', they would pass each other vertically or horizontally, or they meet each other. Descend to an equal height if possible.
    return $A', B', \epsilon_{A'}, \epsilon_{B'}$

function ReverseDescent($\delta, A, B, \epsilon_A, \epsilon_B$)
    ▷ Reverses steepest descent until a classical critical event of type b or c, or a new lexicographic critical event is found. Also returns the critical event c.
    return $A', B', \epsilon_{A'}, \epsilon_{B'}, c$

function ClassicalFréchet($\delta, A, B, \epsilon_A, \epsilon_B$)
    ▷ Using a binary search and the decision algorithm, this function finds the lowest traversable classical critical event that is needed to traverse δ from A to B. This is the same as the algorithm for the classical Fréchet distance.
    return ϵ_F, c

Algorithm 2 Lexicographic Fréchet Matching Algorithm

function TraverseRecursive($\delta, A, B, \epsilon_A, \epsilon_B$)
  if $A = B$ **or** $B.x \leq A.x$ **or** $B.y \leq A.y$ **then**
    ▷ Traversal done. Connect A and B.
    return Connect(A, B)
  $\epsilon \leftarrow \max(\epsilon_A, \epsilon_B)$
  if DecisionAlgorithm(δ, A, B, ϵ) **then**
    $A', B', \epsilon_{A'}, \epsilon_{B'} \leftarrow$ SteepestDescent($\delta, A, B, \epsilon_A, \epsilon_B$)
    $\epsilon' \leftarrow \max(\epsilon_{A'}, \epsilon_{B'})$
    if DecisionAlgorithm($\delta, A', B', \epsilon'$) **then**
      ▷ Traversable after steepest descent.
      return TraverseRecursive($\delta, A', B', \epsilon_{A'}, \epsilon_{B'}$)
    else ▷ Not traversable. The descent passes a critical event.
      while $A \neq A'$ **or** $B \neq B'$ **do**
        $A', B', \epsilon_{A'}, \epsilon_{B'}, c \leftarrow$ ReverseDescent($\delta, A', B', \epsilon_{A'}, \epsilon_{B'}$)
        $\epsilon' \leftarrow \max(\epsilon_{A'}, \epsilon_{B'})$
        if DecisionAlgorithm($\delta, A', B', \epsilon'$) **then**
          ▷ Traversable with the critical event c.
          return TraverseRecursive($\delta, A', c.B, \epsilon_{A'}, \epsilon_c$) + c + TraverseRecursive($\delta, c.A, B', \epsilon_c, \epsilon_{B'}$)
  else ▷ A critical event with $\epsilon_c > \epsilon$ is needed to traverse.
    $\epsilon_F, c \leftarrow$ ClassicalFréchet($\delta, A, B, \epsilon_A, \epsilon_B$)
    return TraverseRecursive($\delta, A', c.B, \epsilon_{A'}, \epsilon_c$) + c + TraverseRecursive($\delta, c.A, B', \epsilon_c, \epsilon_{B'}$)

5 Lexicographic Fréchet Distance with Degenerate Inputs

5.1 Problem

As Rote describes in Section 7 of his paper "Lexicographic Fréchet Matchings" and as we hinted to in Section 4.3 of this paper, multiple critical events can occur for the same ϵ.

Multiple critical events for the same ϵ turn out to be problematic, because as can be seen in Algorithm 2, we assume the functions defined in Algorithm 1 return only one critical event, which will be traversed.

This section aims to answer the question of how to handle multiple critical events with the same critical ϵ by visualizing the problem and postulating an algorithm that decides which combination of critical events result in a lexicographic traversal. The basis of this section is Section 7 of Rote's paper "Lexicographic Fréchet Matchings" [2].

5.2 Examples

In this subsection, four examples will be discussed, for which multiple critical events for the same critical ϵ require a decision on which of them should be traversed to result in a lexicographic traversal. Instead of providing in-depth detail on solutions, this subsection aims to illustrate the problem through examples.

5.2.1 Minimal Example: Traverse Both

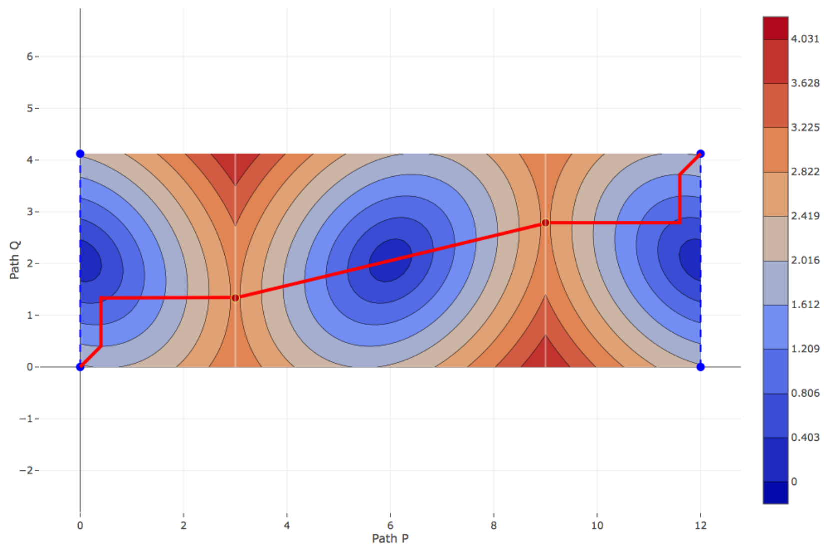


Figure 4: One Path Over Two Critical Events[6]

Figure 4 shows a minimal example with two critical events with the same ϵ. There are two classical critical events of type b at $\epsilon \approx 2.91$ at the two vertical cell-borders. Both critical events need to be traversed.

[6] View the example here: `https://abegehr.github.io/frechet/?p=(5_5)(8_5)(2_5)(5_5)&q=(5.5_3)(4.5_7)`

5.2.2 Minimal Example: Traverse Either

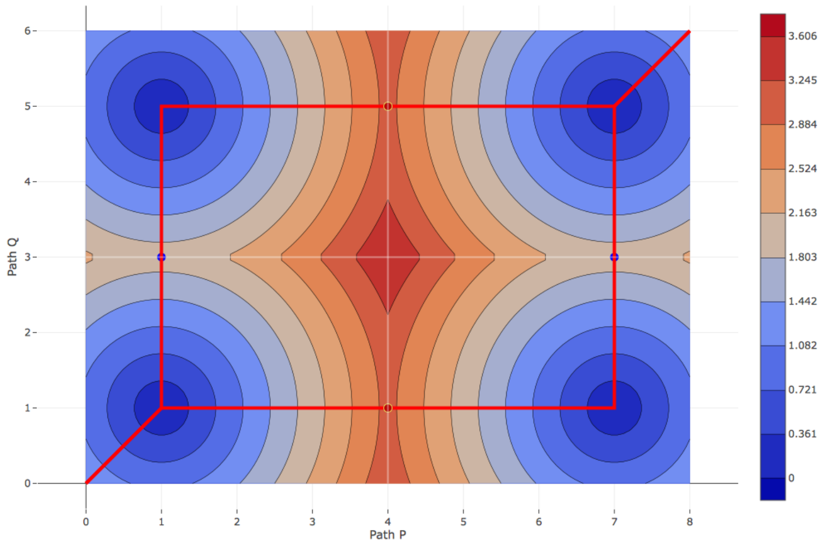


Figure 5: Two Equivalent Paths[7]

Figure 5 shows a minimal example of multiple critical events for the same ϵ. There are four critical events to consider. Two classical critical events of type b with equal $\epsilon = 3$ at the two vertical cell-borders. And two classical critical events of type b with equal $\epsilon = 2$ at the two horizontal cell-borders. The decision problem shows that $\epsilon = 3$ is needed to traverse this height function δ. Either one of the critical events at $\epsilon = 3$ need to be traversed.

[7]View the example here: `https://abegehr.github.io/frechet/?p=(4_5)(8_5)(4_5)&q=(5_4)(5_7)(5_4)`

5.2.3 Minimal Example: Traverse One

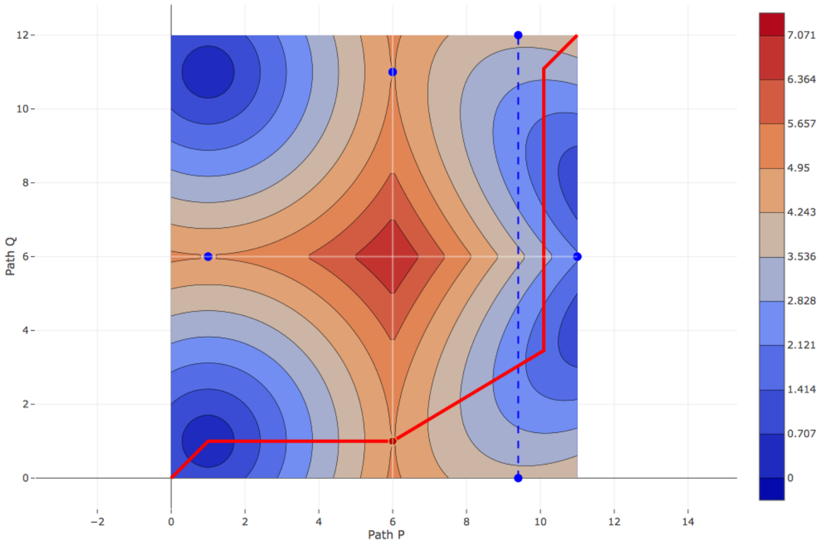


Figure 6: Two Inequivalent Paths[8]

Figure 6 shows three classical critical events of type b at $\epsilon = 5$ to consider: $C_1(6|1)$, $C_2(1|6)$, and $C_3(6|11)$. There are two possible paths through these three critical events: traversing just C_1 or traversing C_2 and C_3. Since the ascends to and descends from C_1, C_2, and C_3 are similar, we choose the path over only C_1.

5.2.4 Complex Example

Figure **??** shows a more complex example of multiple critical events for the same ϵ.

5.3 Possible Paths

We have seen in the examples that in some cases we need to traverse a path through multiple of the critical events at equal height ϵ_0, while in other cases a path through one critical events is sufficient. To decide which path to traverse, first, we need to find all possible paths through the critical events.

The possible paths through multiple critical events can be represented as a graph. The start node represents the lower left corner A and the end

[8]View the example here: `https://abegehr.github.io/frechet/?p=(3_2)(3_8)(6_4)&q=(2_3)(8_3)(2_3)`

node represents the upper right corner B of the current δ subsection. Each critical event $C_k \in C$ of height ϵ_0 is a node. Edges are directional and denote traversability without another critical event of height ϵ_0 in between.

To generate this graph of critical traversals, we make use of the free-space diagram and the technique used in the decision algorithm. Fist we use the free-space diagram represented by L^F_{ij} and B^F_{ij} and the decision algorithm (see Alt and Godau [1]) to generate the reachable border bounds L^R_{ij} and B^R_{ij} of all cells. Then we use the reachable border bounds and the critical events C at height ϵ_0 to generate for each vertical and horizontal cell-border, which subset of critical events C can reach the border. We call the subset of C that reaches the left and bottom border of cell (i, j), L^C_{ij} and B^C_{ij} respectively. After computing each L^C_{ij} and B^C_{ij}, we then connect critical events in L^C_{ij} and B^C_{ij} to reachable critical events starting on the top and right border.

See the algorithm 3 for more detail on how we generate the traversability graph of multiple critical traversals C_k that are at same height ϵ_0. p and q is the number of cells between the starting-point A and the ending-point b.

Algorithm 3 Generate Traversability Graph of multiple critical events at ϵ_0

function GenerateTraversalGraph(A, B, C, ϵ_0)
  $\triangleright$ A is start-point of traversal. B is ending-point of traversal. C holds all critical events C_k.
  Add A and B as critical events to C.
  $\forall i, j : L^C_{ij} := \{\} \wedge B^C_{ij} := \{\}$
  $L^R, B^R \leftarrow$ DecisionAlgorithm'(A, B, ϵ_0)
  for $C_k \in C$ **do**
    Add $C_k.B$ (ending-points) to L^C_{ij} or B^C_{ij} where $C_k.B$ lies.
  for $i := 0$ to p **do** determine $L^C_{i,0}$
  for $j := 0$ to q **do** determine $B^C_{0,j}$
  for $i := 0$ to p **do**
    for $j := 0$ to q **do**
      for C_{k2} that starts on right or top border **do**
        for $C_{k1} \in L^C_{i,j} \cup B^C_{i,j}$ **do**
          if C_{k1} can reach C_{k2} **then**
            Add edge (C_k1, C_k2) to graph.
      construct $L^C_{i+1,j}$ and $B^C_{i,j+1}$ from L^C_{ij}, B^C_{ij}, $L^R_{i+1,j}$, $B^R_{i,j+1}$
  return graph

5.3.1 Can C_{k1} reach C_{k2}

In the above algorithm 3 on line 14, we seek to determine if the critical event C_{k1} can reach the critical event C_{k2}. It is known that C_{k1} can reach the left or bottom border of the current cell (i, j), because $C_{k1} \in L^C_{i,j} \cup B^C_{i,j}$. It also is know that C_{k2} starts at the top or right border of the current cell.

We discard the potential edge (C_{k1}, C_{k2}) in the trivial case where C_{k2} is not monotonically reachable from C_{k1}: $\neg(C_{k1}.B.x \leq C_{k2}.A.x \wedge C_{k1}.B.y \leq C_{k2}.A.y)$.

To determine if C_{k2} is reachable from C_{k1}, we first look at if C_{k1} comes the left or bottom border and if C_{k2} is on the right or top border. There are four cases to consider:

1. $C_{k1} \in L^C_{i,j}$ enters on left border and C_{k2} starts on right border.
2. $C_{k1} \in L^C_{i,j}$ enters on left border and C_{k2} starts on top border.
3. $C_{k1} \in B^C_{i,j}$ enters on bottom border and C_{k2} starts on right border.
4. $C_{k1} \in B^C_{i,j}$ enters on bottom border and C_{k2} starts on top border.

Reachability intervals I_{ij} of right and top borders can be computed as follows:

1. From left to right: $I^{l \to r}_{ij} = L^R_{i,j} \cap L^R_{i+1,j}$
2. From left to top: If $L^R_{i,j}$ is open: $I^{l \to t}_{ij} = B^R_{i,j+1}$. Otherwise closed.
3. From bottom to right: If $B^R_{i,j}$ is open: $I^{b \to r}_{ij} = L^R_{i+1,j}$. Otherwise closed.
4. From bottom to top: $I^{b \to t}_{ij} = B^R_{i,j} \cap B^R_{i,j+1}$

The superscript denotes the sides of the cell. $I^{b \to t}_{ij}$ for example represents the reachability interval of the top border for traversals coming from the bottom border.

It can be considered to use solely these reachability intervals I_{ij} to decide if C_{k2} can be reached from C_{k1}. One would simply check if C_{k2} lies in the reachability intervals from the side on which C_{k1} lies. This is not sufficient, as the example visualized in figure 7 shows.

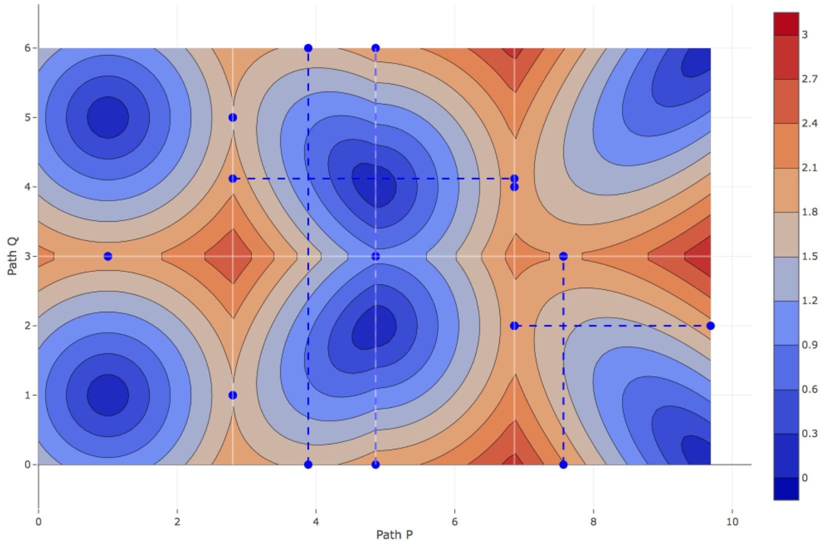


Figure 7: Considering solely reachability intervals is not sufficient[9]

In figure 7 the decision algorithm tells us that the classical Fréchet distance is $\epsilon_F = 2$. There are two critical events of type b with $\epsilon = 2$ that we consider: $C_1(3, 1)$ and $C_2(\sim 6.86, 4)$. If we were to use solely the reachability intervals to check if C_{k2} is reachable from C_{k1} as described above, the generated graph would find an edge between C_1 and C_2, even though there exists no such path, because it is closed by the right border of cell $(0, 1)$: $L^R_{1,1}$. Using solely the reachability intervals, C_2 would still appear reachable from C_1, because $L^R_{2,1}$ starts below $L^R_{1,1}$, since $L^R_{2,1}$ can also be reached from $B^R_{1,1}$, and not just from $L^R_{1,1}$. We can now learn from this counter example and apply better measures.

To solve the problem of reachability intervals for generating a traversability graph of critical events, we need to memorize for each critical event not just that it can reach border L_{ij} or B_{ij}, but also for vertical borders the minimum y-coordinate and for horizontal borders the minimum x-coordinate that it needs to reach the border, as these can differ for different critical events that both reach a border. Then when we connect a critical event C_{k1} to C_{k2}, we test the reachability intervals and we determine if the C_{k2} lies on or above the minimum x- or y-coordinate that C_{k1} needs to reach the border in question.

We can now define two algorithm in more detail. First, the algorithm for determining if C_{k1} can reach C_{k2}. And second, the algorithm that constructs

[9] View the example here: `https://abegehr.github.io/frechet/?p=(5_5)(5_7.8)(4_6)(4_8)(6_6)&q=(6_6)(3_6)(6_6)`

$L^C_{i+1,j}$ and $B^C_{i,j+1}$ from L^C_{ij}, B^C_{ij}, $L^R_{i+1,j}$, $B^R_{i,j+1}$.

Algorithm 4 Generate Traversability Graph Helper Functions

function CANREACH($C_{k1}, min_x, min_y, C_{k2}, I^{l\to r}_{ij}, I^{l\to t}_{ij}, I^{b\to r}_{ij}, I^{b\to t}_{ij}$)
 ▷ Determines if C_{k1} can reach C_{k2}.
  if C_{k1} is on left border and C_{k2} is on right border **then**
    return $C_{k2}.A.y \in I^{l\to r}_{ij} \wedge C_{k2}.A.y \geq min_y$
  if C_{k1} is on left border and C_{k2} is on top border **then**
    return $C_{k2}.A.x \in I^{l\to t}_{ij} \wedge C_{k2}.A.x \geq min_x$
  if C_{k1} is on bottom border and C_{k2} is on right border **then**
    return $C_{k2}.A.y \in I^{b\to r}_{ij} \wedge C_{k2}.A.y \geq min_y$
  if C_{k1} is on bottom border and C_{k2} is on top border **then**
    return $C_{k2}.A.x \in I^{b\to t}_{ij} \wedge C_{k2}.A.x \geq min_x$

function CONSTRUCTCRITICALREACH(L^C_{ij}, B^C_{ij}, $L^R_{i+1,j}$, $B^R_{i,j+1}$)
 ▷ Constructs $L^C_{i+1,j}$ and $B^C_{i,j+1}$ from L^C_{ij}, B^C_{ij}, $L^R_{i+1,j}$, $B^R_{i,j+1}$
  $L^C_{i+1,j} := \{\}$
  $B^C_{i,j+1} := \{\}$
  if $L^R_{i+1,j}$ is not closed **then**
    for $(C_k, min_x) \in B^C_{ij}$ **do**
      Add $(C_k, L^R_{i+1,j}.min)$ to $L^C_{i+1,j}$
    for $(C_k, min_y) \in L^C_{ij}$ **do**
      if $min_y \leq L^R_{i+1,j}.max$ **then**
        $min_y \leftarrow max(min_y, L^R_{i+1,j}.min)$
        Add (C_k, min_y) to $L^C_{i+1,j}$
  if $B^R_{i,j+1}$ is not closed **then**
    for $(C_k, min_y) \in L^C_{ij}$ **do**
      Add $(C_k, B^R_{i,j+1}.min)$ to $B^C_{i,j+1}$
    for $(C_k, min_x) \in B^C_{ij}$ **do**
      if $min_x \leq B^R_{i,j+1}.max$ **then**
        $min_x \leftarrow max(min_x, B^R_{i,j+1}.min)$
        Add (C_k, min_x) to $B^C_{i,j+1}$
  return $L^C_{i+1,j}, B^C_{i,j+1}$

The two functions defined in algorithm 4 are necessary for algorithm 3 to generate the correct traversability graph for a set of critical events at the same height ϵ_0.

Checking if $C_{k2}.A$ lies in the respective reachability interval might be superfluous, because we are already checking for the minimum x- or y-coordinate. Nevertheless, also checking the reachability interval I_{ij} does not hurt the valid functioning.

5.4 Traversal cross-section $f(t)$ and profile $\hat{f}(s)$

Due to our input being two paths consisting of connected straight line-segments, every straight cross-section through our height functionδ is a continuous composition of hyperbolas. We use the following function to denote these hyperbolas h:

$$h : \epsilon(t) = \sqrt{u^2 + a(t-v)^2}$$

Each hyperbola has three parameters u, a, and v. A hyperbola is always a cross-section though a segment of the height function δ. An equation system is used to determine the three parameters u, a, and v. There are two cases to consider:

1. For a horizontal of vertical cross-section, $a = 1$. Therefore, only two points (t, ϵ) are needed to determine u and v. The two points (t, ϵ) are calculated by sampling two pairs of points from the input line-segments and determining there t and distance ϵ.

2. For a diagonal cross-section three points (t, ϵ) are needed to determine u, a and v. The two points (t, ϵ) are calculated by sampling three pairs of points from the input line-segments and determining there t and distance ϵ.

For simplicity, we define the hyperbolas all as starting at $t = 0$. This can be done by adjusting the parameter v.

To attain a cross-section $f(t)$ of an entire traversal, we simply create a composite function of the N hyperbolas and move them along the t-axis to the correct position. This yields $f(t)$. Δt_k is the time interval in which the k^{th} hyperbola is traversed.

$$f(t) = \begin{cases} \sqrt{u_1^2 + a_1(t-v_1)^2} & 0 \le t \le \Delta t_1 \\ \sqrt{u_2^2 + a_2(t-v_2-\Delta t_1)^2} & \Delta t_1 \le t \le \Delta t_2 \\ \dots & \\ \sqrt{u_N^2 + a_N(t-v_N-\Delta t_{N-1})^2} & \Delta t_{N-1} \le t \le \Delta t_N \end{cases}$$

For the example in figure 4, the cross-section $f(t)$ is shown in figure 8. The x-axis shows the time and the y-axis shows the current ϵ. Notice that when the traversal moves on the l-lines, the cross-section is linear. This is because the addend u^2 inside the square root equals zero for this case. Therefore, the square and the square root cancel and we are left with the absolute value of a linear function: $|\sqrt{a}(t-v)|$.

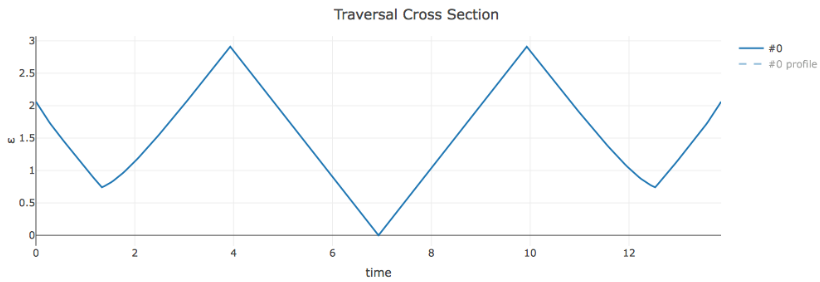


Figure 8: Traversal cross section $f(t)$ for example in figure 4[10]

In addition to the cross-section $f(t)$, we can also look at the profile $\hat{f}(s)$. Rote defines the profile function $\hat{f}(s)$ as the time that ϵ exceeds a threshold s:

$$\hat{f}(s) = \mu(\{t \mid f(t) \geq s\})$$

where μ denotes the Lebesgue measure.[2]

Figure 9 shows the profile function $\hat{f}(s)$ in addition to the function of the cross-section shown in figure 8 for the example from figure 4.

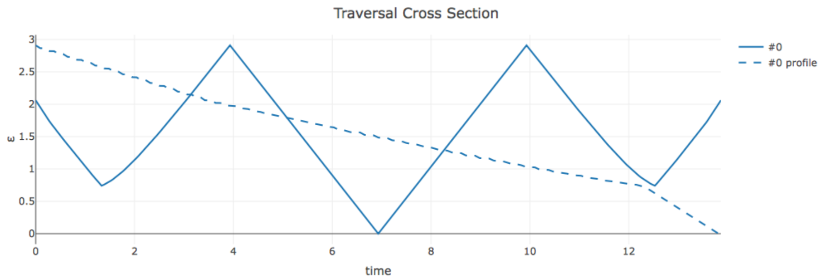


Figure 9: Traversal cross section $f(t)$ and profile $\hat{f}(s)$ for example in figure 4[10]

Next the profile function $\hat{f}(s)$ is defined using the cross-section hyperbolas. It is important to note that each hyperbola function $H_k(t)$ is defined so that $0 \leq t \leq \Delta t_k - \Delta t_{k-1}$; therefore, we can simply sum up the inverses of

[10] View the example here: `https://abegehr.github.io/frechet/?p=(4_5)(8_5)(4_5)&q=(5_4)(5_7)(5_4)`

the hyperbolas in these bounds to calculate the profile in the following two steps:

1. Compute the inverse function of a hyperbolas function $h^{-1}(\epsilon)$. First solve $\epsilon(t)$ for t:

$$
\begin{aligned}
\epsilon &= \sqrt{u^2 + a(t-v)^2} \\
\epsilon^2 &= u^2 + a(t-v)^2 \\
\epsilon^2 - u^2 &= a(t-v)^2 \\
\frac{\epsilon^2 - u^2}{a} &= (t-v)^2 \\
\pm\sqrt{\frac{\epsilon^2 - u^2}{a}} &= t - v \\
v \pm \sqrt{\frac{\epsilon^2 - u^2}{a}} &= t
\end{aligned}
$$

   This results in the following definition for $h^{-1}(\epsilon)$:

$$
h_k^{-1} : t_k(\epsilon) = \begin{cases} v_k \pm \sqrt{\frac{\epsilon^2 - u_k^2}{a_k}} & \text{if } h_k(0) \leq \epsilon \leq h_k(\Delta t_k - \Delta t_{k-1}) \\ 0 & \text{otherwise} \end{cases}
$$

   Note that h_k is valid in the interval $0 \leq t \leq \Delta t_k - \Delta t_{k-1}$; therefore, h_k^{-1} is defined for $h_k(0) \leq \epsilon \leq h_k(\Delta t_k - \Delta t_{k-1})$. This is important to yield the correct profile function $\hat{f}(s)$. Also notice that each hyperbola h_k has its own three parameters: u_k, a_k, and v_k.

2. Sum up the inverses to yield the profile function $\hat{f}(s)$.

$$
\hat{f}(s) = \sum_{k=0}^{N} h_k^{-1}(s)
$$

Now we have defined the profile function $\hat{f}(s)$ for a traversal with cross-section $f(t)$ that allows us to calculate the amount of time a traverser traversing this traversal spends above the threshold s.

5.5 Ascents and Descents

To fulfill our goal of a lexicographic traversal, we want to minimize the time during which the height ϵ exceeds a threshold s.[2] Concerning our examples, we want to minimize the time needed to ascend to and descend from our critical height ϵ_0.

If we have multiple critical events at the critical height ϵ_0, we need to decide which to choose while fulfilling our goal of a lexicographic traversal. The profile function $\hat{f}(s)$, introduced in section 5.4, defines the time that a traversals exceeds a threshold s. Our goal of a lexicographic traversal is to minimize this time. This means that for two profile functions $\hat{f}(s)$ and $\hat{g}(s)$ that both are critical at ϵ_0, we need to compare the derivatives. Note that we only derive the hyperbola inverses that are defined for s.

$$\hat{f}'(s) = \left[\sum_{k=0}^{N} h_k^{-1}(s)\right]'$$

Because of the sum rule, we can apply the derivative to each H_k^{-1}.

$$\hat{f}'(s) = \sum_{k=0}^{N} [h_k^{-1}]'(s)$$

Lets look at h_k^{-1} again:

$$h_k^{-1} : t_k(\epsilon) = \begin{cases} v_k \pm \sqrt{\frac{\epsilon^2 - u_k^2}{a_k}} & \text{if } H_k(0) \leq \epsilon \leq H_k(\Delta t_k - \Delta t_{k-1}) \\ 0 & \text{otherwise} \end{cases}$$

Due to it's form, the derivative of h_k^{-1} is not trivial. h_k^{-1} is a composite function and has a $\pm$. We circumvent this problem by only looking at hyperbolas that ascend to and descend from the critical height ϵ_0.

For a critical event C_k at ϵ_0, we will denote the ascent hyperbola with $h_{k\uparrow}$ and the descent hyperbola with $h_{k\downarrow}$. For simplicity we define the hyperbolas so that $t = 0$ for the critical height ϵ: $h_k(0) = \epsilon_0$. This means that $h_{k\uparrow}(t)$ is defined for the interval $-(\Delta t_k - \Delta t_{k-1}) \leq t \leq 0)$ and $h_{k\downarrow}(t)$ are defined for the interval $0 \leq t \leq \Delta t_k - \Delta t_{k-1}$.

The hyperbolas are vertically symmetric at $t_v = v$, $h(t_v + t) = h(t_v - t)$; therefore, the inverse hyperbolas are horizontally symmetric at $t_v = v$. This means that to get rid of the $\pm$ in the inverse and get only positive slopes, we can simply use a $+$:

$$h_\uparrow^{-1}(\epsilon) = v_\uparrow + \sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}} \qquad h_\downarrow^{-1}(\epsilon) = v_\downarrow - \sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}$$

We now determine the derivatives of $h_\uparrow^{-1}(\epsilon)$ and $h_\downarrow^{-1}(\epsilon)$:

$$[h_\uparrow^{-1}]'(\epsilon) = \left[v_\uparrow + \sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}}\right]' \qquad [h_\downarrow^{-1}]'(\epsilon) = \left[v_\downarrow - \sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}\right]'$$

$$[h_\uparrow^{-1}]'(\epsilon) = \left[\sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}}\right]' \qquad [h_\downarrow^{-1}]'(\epsilon) = -\left[\sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}\right]'$$

$$[h_\uparrow^{-1}]'(\epsilon) = \frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}}} \qquad [h_\downarrow^{-1}]'(\epsilon) = -\frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}}$$

We have defined the hyperbolas so that $h(0) = \epsilon_0$; therefore, $h^{-1}(\epsilon_0) = 0$.

$$h_\uparrow^{-1}(\epsilon_0) = v_\uparrow + \frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}}} \qquad h_\downarrow^{-1}(\epsilon_0) = v_\downarrow - \frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}}$$

$$0 = v_\uparrow + \frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}}} \qquad 0 = v_\downarrow - \frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}}$$

$$v_\uparrow = -\frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\uparrow^2}{a_\uparrow}}} \qquad v_\downarrow = \frac{\epsilon}{\sqrt{\frac{\epsilon^2 - u_\downarrow^2}{a_\downarrow}}}$$

There are three cases to consider:

1. $\hat{f}'(s) > \hat{g}'(s)$: Choose traversal f.
2. $\hat{f}'(s) < \hat{g}'(s)$: Choose traversal g.
3. $\hat{f}'(s) = \hat{g}'(s)$: Cannot decide.

5.5.1 Example: First Derivative

Let us consider an example where we can decide from looking at the first derivative which traversal path to choose. Figure 10 shows such an example.

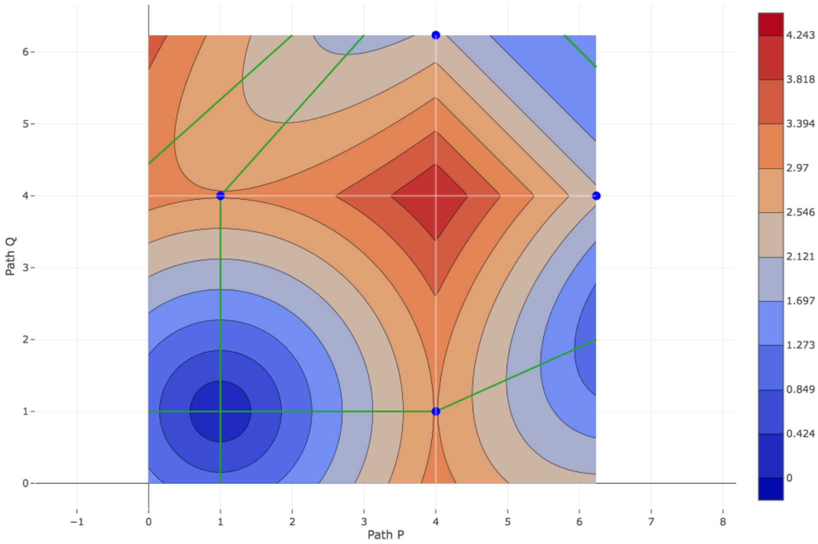


Figure 10: Two critical events with unequal first derivative of profiles[11]

In the example depicted in figure 10, there are two classical critical events of type b at critical height $\epsilon_0 = 3$ to consider: $C_1(1,4)$ and $C_2(4,1)$.

The algorithm 3 that generates a traversability graph for multiple critical events at the same height as defined in 5.3, returns that there are two possible paths through the critical events C_1 and C_2. Either the graph is traversed from the starting-point to the ending-point over C_1 or over C_2.

We will now look at the ascent to and descent from C_1 and C_2.

$$\begin{aligned}
H_{1\uparrow}(t) &= \sqrt{0^2 + 1 \cdot (t - (-3))^2} = |t+3| \\
H_{1\downarrow}(t) &= \sqrt{0^2 + 0.8 \cdot (t - 3.354)^2} = \left|\sqrt{0.8}(t - 3.354)\right| \\
H_{2\uparrow}(t) &= \sqrt{0^2 + 1 \cdot (t - (-3))^2} = |t+3| \\
H_{2\downarrow}(t) &= \sqrt{0^2 + 0.2 \cdot (t - 6.708)^2} = \left|\sqrt{0.2}(t - 6.708)\right|
\end{aligned}$$

We now determine the specific derivatives of the inverse hyperbolas for the example above in figure 10.

[11] View the example here: `https://abegehr.github.io/frechet/?p=(0_1)(4_1)(2_2)&q=(1_0)(1_4)(3_3)`

$$
\begin{aligned}
H'_{1\uparrow}(\epsilon_0) &= -3 + \frac{\epsilon_0}{\sqrt{\frac{\epsilon_0^2 - 0^2}{1}}} = -3 + \frac{3}{\sqrt{3^2}} \\
H'_{1\downarrow}(\epsilon_0) &= 3.354 + \frac{\epsilon_0}{\sqrt{\frac{\epsilon_0^2 - 0^2}{0.8}}} \\
H'_{2\uparrow}(\epsilon_0) &= -3 - \frac{\epsilon_0}{\sqrt{\frac{\epsilon_0^2 - 0^2}{1}}} \\
H'_{2\downarrow}(\epsilon_0) &= 6.708 + \frac{\epsilon_0}{\sqrt{\frac{\epsilon_0^2 - 0^2}{0.2}}}
\end{aligned}
$$

5.6 Postulate Algorithm

Postulate algorithm for choosing a critical event from several with equal epsilon: (visual examples!)

Algorithm:

1. For every critical event compute reciprocal of derivates descents and store as critical event's steepness.
2. From set of critical events, generate all possible sequences (ascending monotone).
3. For all sequences sum steepness of their critical events and store as sequence rank.
4. Starting with the smallest-rank sequence, decide if the sequence of critical events is traversable without passing critical events excluded in the sequence.
5. If decision is negative, repeat 4.
6. If decision is positive, repeat 4 for all sequences of equal rank.
7. If multiple sequences of equal rank are found to be valid, traverse all, and decide by comparing.
8. The steepness of the steepest decent.
9. The steepness of critical event that is reached.
10. And choose the path with the steepest summed recents of all paths.

11. If multiple paths with same hight profile are found, return all. Otherwise return the lexicographic optimum.

12.

5.7 Traversing critical events

Assume we are traversing a heat map generated by two curves. We arrive at a point where we have to decide which critical events need to be traversed for a lexicographic solution. The decision algorithm[1] can be applied for the ϵ of each possible critical event. We pick the smallest ϵ for which the decision algorithm returns a positive decision. We call it ϵ_0. Now two options are feasible:

(A) There is one critical event at height ϵ_0. $|C_{\epsilon_0}| = 1$

(B) There are multiple critical events at height ϵ_0. $|C_{\epsilon_0}| > 1$

In case option A holds true, this critical event will need to be traversed. The traversal-problem is divided into two sub problems, which are both examined recursively and independently.

In case option B holds true, we need to decide which of the critical events C_{ϵ_0} need to be traversed and which are not needed for an optimal solution. We will consider all subsets C'_{ϵ_0} of C_{ϵ_0} with size $n > 0$.

5.8 Example of Algorithm visualised (visual example!)

5.9 Analysing postulated algorithm for handling degenerate inputs

5.9.1 Runtime analysis

5.10 Real world application

Can you think of one?

6 Concluding Perspective

Concluding, all is great! :)

References

[1] Helmut Alt and Michael Godau. "Computing the Fréchet distance between two polygonal curves". In: *International Journal of Computational Geometry & Applications* 5.01n02 (1995), pp. 75–91.

[2] Günter Rote. "Lexicographic Fréchet Matchings". In: *30th European Workshop on Computational Geometry*. EuroCG. 2014.