Map-Matching Using Shortest Paths

Erin Chambers
Department of Computer Science
Saint Louis University
Saint Louis, Missouri, USA
erin.chambers@slu.edu

Brittany Terese Fasy
School of Computing and Dept. Mathematical Sciences
Montana State University
Montana, USA
brittany@cs.montana.edu

Yusu Wang
Department of Computer Science and Engineering
The Ohio State University
Columbus, Ohio, USA
yusu@cse.ohio-state.edu

Carola Wenk
Department of Computer Science
Tulane University
New Orleans, Louisianna, USA
cwenk@tulane.edu

ABSTRACT

We consider several variants of the *map-matching* problem, which seeks to find a path Q in graph G that has the smallest distance to a given trajectory P (which is likely not to be exactly on the graph). In a typical application setting, P models a noisy GPS trajectory from a person traveling on a road network, and the desired path Q should ideally correspond to the actual path in G that the person has traveled. Existing map-matching algorithms in the literature consider *all* possible paths in G as potential candidates for Q. We find solutions to the map-matching problem under different settings. In particular, we restrict the set of paths to shortest paths, or concatenations of shortest paths, in G. As a distance measure, we use the Fréchet distance, which is a suitable distance measure for curves since it takes the continuity of the curves into account.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**;

KEYWORDS

shortest paths, Fréchet distance, map matching

ACM Reference Format:
Erin Chambers, Brittany Terese Fasy, Yusu Wang, and Carola Wenk. 2018. Map-Matching Using Shortest Paths. In *IWISC 2018: 3rd International Workshop on Interactive and Spatial Computing, April 12–13, 2018, Richardson, TX, USA.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3191801.3191812

1 INTRODUCTION

The *map-matching* problem seeks to find a path Q in a planar graph $G = (V, E)$ that has the smallest distance to P. In a typical application setting, P models a noisy GPS trajectory from a person traveling on a road network, modeled as the planar graph G, and the desired path Q should correspond to the actual path in G that the person has traveled. Map-matching algorithms in the literature [1, 2, 6] consider *all* possible paths in G as potential candidates for Q, and apply similarity measures such as Hausdorff or Fréchet distance to compare input curves.

We propose to restrict the set of potential paths in G to a natural subset: those paths that correspond to shortest paths, or concatenations of shortest paths, in G. Restricting the set of paths to which a path can be matched makes sense in many settings. In particular, vehicles often follow routes computed by a navigation system, which often prefers certain types of routes over others; and drivers often have preferences for different roads [7]. The current literature also does not consider the case where the vehicle makes multiple stops. For example, consider a person running several errands in one trip, where we are given the approximate path that the person followed, along with the underlying map. In this setting, knowledge of the number of stops as well as the type of path preferred (shortest travel time, shortest distance, or perhaps avoiding certain types of roads) can improve the final quality of the path that our algorithm matches to in the graph.

Related work. Map-matching is widely used in practice, e.g., to establish fast routes or points of interest from a large set of trajectories [17, 18]. Common approaches include the use of Fréchet distance variants [2, 6], incremental methods [6, Sec. 3], matching low-sampling-rate trajectories using spatial-temporal constraints [12, 15], and hidden Markov models [13, 16]. Despite this, only few map-matching algorithms provide quality guarantees.

Only a small proportion of prior work considers restricting the set of paths in G. Instead, common practice reduces the space of paths by cropping G inside an ε-neighborhood around P before applying a general map-matching algorithm. Gheibi et al. [9] gave a map-matching algorithm that minimizes the sum of the lengths of walks on P and Q within some Fréchet distance. Their algorithm runs in $O(Nm(N+m)\log(N+m))$ time and $O(Nm(N+m))$ space, where $n = |V|, m = |E|, N = |P|$, and computes a shortest path in a discretized free space. Finally, Zhu et al. [20] consider a similar notion as the one we study here, and break a longer path in order to map each subsegment to a short path; however, the curve matching

IWISC 2018, April 12–13, 2018, Richardson, TX, USA

ACM ISBN 978-1-4503-5439-4/18/04…$15.00
https://doi.org/10.1145/3191801.3191812

is done via a "Longest Common Subsequence" score, incorporating Hausdorff like similarity measures, and does not guarantee absolutely shortest paths for each sub trajectory matched.

Our contribution. We provide algorithms for variants of the map-matching problem, in which the set of paths are restricted to shortest paths, or concatenations of shortest paths, in the graph [1]. As a distance measure between paths, we use the Fréchet distance, which is a standard distance measure for curves in this setting that produces better matchings than some others, such as the Hausdorff distance, since it takes the continuity of the curves into account.

In Section 3, we provide an algorithm to match P to the shortest possible path within a given Fréchet distance in G. We prove properties of a distance function on the free space and we compute it incrementally, which allows us to use less space than [9]. In Section 4, we give algorithms to match P to concatenations of shortest paths in G: In the **min-k** variant, we find a path Q in G consisting of the smallest number k of shortest path pieces that does not exceed a given Fréchet distance. In the **min-ε** variant, we find a path Q in G consisting of at most k shortest paths, for given k, such that the Fréchet distance to P is minimized. We assume that all break-points between shortest paths lie on vertices of G and map to vertices of P. In Section 5, we relax this constraint on the break-points, and provide approximation algorithms that approximate the number of shortest path pieces as well as the Fréchet distance ε when break-pointscan lie in the interior of edges in G and can be mapped to the interior of edges of P.

To the best of our knowledge, we present the first study of Fréchet-based map-matching algorithms that consider a subset of paths with pre-defined properties in G to be matched to P. Our paper expands this new perspective on map-matching and provides further theoretical foundations for the practically relevant problem, where we consider a restricted set of path classes.

2 PRELIMINARIES

Let $G = (V, E)$ be a geometric graph where each node in V is embedded in the Euclidean space, and each graph edge is mapped to the segment connecting its two nodes. Let P be a polygonal path. We parameterize each (undirected) edge $e = (u, v) \in E$ linearly by $e(s) := (1 - s)u + sv$ for $s \in [0, 1]$, where the direction of the parameterization is fixed, but arbitrary. Let $p_0, p_1, \ldots, p_N$ be the sequence of $N+1$ vertices defining the polygonal path P. We identify each of these vertices with a point in the plane, and we parameterize each line segment edge $e_i = (p_i, p_{i+1})$ linearly by $p_i(t) := (1-t)p_i + tp_{i+1}$ for $t \in [0, 1]$. We use $P[p_i, p]$ to denote the polygonal subpath from p_i to some other point $p \in P$.

The length of a path or subpath, either in G or P, is simply the sum of all edge-lengths in the path; in the case of partial edges, we use the fact that we have an arc length parameterization of all edges, and take the arc length of the partial edge.

We are interested in finding a path in G that is *close* to an input path P. To measure this *closeness*, we use the Fréchet distance [8]. Consider any two curves $\alpha, \beta : [0, 1] \to R^2$. Let ϕ and ψ be orientation-preserving homeomorphisms that serve as reparameterizations of $[0, 1]$. We can measure the distance between $(\alpha \circ \phi)$ and $(\beta \circ \psi)$ pointwise, and take the supremum. Then, the Fréchet distance $\delta_F(\alpha, \beta)$ is defined to be the infimum of this measurement over all reparameterizations ϕ and ψ. Formally: $\delta_F(\alpha, \beta) = \inf_{\phi,\psi} \sup_{s,t\in[0,1]} ||(\alpha \circ \phi)(t) - (\beta \circ \psi)(t)||$. Intuitively, one can imagine a man walking along one curve and a dog along the other, continuously from beginning to end without backtracking. Then, the Fréchet distance is the shortest leash needed to connect the man and dog on their walk.

[1]We note that while it has been recognized that factors other than purely shortest distances may affect how people choose routes (see e.g [21]), many advanced models still rely heavily on the shortest path assumption e.g., [4, 5, 11, 14, 19]

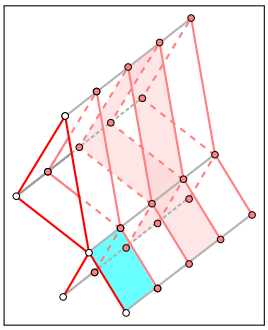

Figure 1: We illustrate the parameter space $G \times P$, a graph G (shown in red with white vertices) times a path P of length four. For convenience, the path is drawn as a straight path. A *slice* is the graph cross an edge e of the path: $G \times e$; see the shaded pink region. A *level* is the graph cross a vertex v in the path: $G \times v$. Each level can be thought of as a copy of G.

In order to match the path to the graph, we consider the cell complex $G \times P$; see Figure 1. By convention, we say that the graph $G = (V, E)$ is *horizontal* and the path P is *vertical*. For an edge $(u, v) \in E$ and consecutive path vertices p_i and p_{i+1}, we consider the cell $(u, v) \times (p_i, p_{i+1}) \subseteq G \times P$ to be drawn with (u, v) as a horizontal edge and (p_i, p_{i+1}) as a vertical edge, as shown in Figure 2. A *slice* is the graph G cross an edge (p_i, p_{i+1}) of the path, $G \times (p_i, p_{i+1})$, and a *level* is the graph cross a vertex p_i of the path, $G \times p_i$.

For $\varepsilon > 0$, the corresponding *free space diagram* D_ε is the subset of $G \times P$ such that for all pairs $(g, p) \in D_\varepsilon$, the following inequality is satisfied: $||g - p|| \leq \varepsilon$. The free space of a cell is equal to an ellipse intersected with the cell [3]. As a consequence, equality $||g - p|| = \varepsilon$ holds for at most two points on each vertical or horizontal edge in the complex. On a vertical edge $u \times (p_i, p_{i+1})$, we denote these two points by a^i_u and b^i_u. Where appropriate, we slightly abuse notation and use a^i_u to also identify the parameter t for which $p_i(t) = a^i_u$. In this way, we say $a^i_u \leq b^i_u$. Likewise, on a horizontal edge $(u, v) \times p_i$, we denote $c^i_u \leq d^i_u$ as the points for which $||c^i_u - p_i|| = ||d^i_u - p_i|| = \varepsilon$; see Figure 2.

3 SHORTEST AMONG MATCHING PATHS

In this section, we consider only paths in G that have restricted Fréchet distance to an input polygonal curve, and among those paths, we wish to find a shortest path. In short, we are interested in finding the *shortest matching path*:

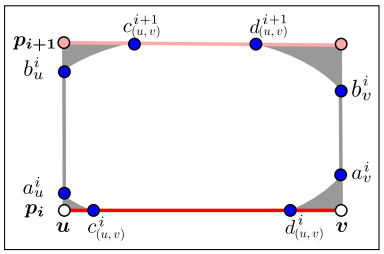

Figure 2: We illustrate one free space cell $(u, v) \times (p_i, p_{i+1})$, where (u, v) is an edge in the graph and p_i, p_{i+1} are consecutive points in P. The *free space* is equivalent to an ellipse intersecting this rectangle. Therefore, each edge of the rectangle has at most two points (g, p) for which $||g - p|| = \varepsilon$.

Problem 1 (Shortest Matching Path). *Given a parameter $\varepsilon > 0$ and a path P, find the shortest path in G that is within Fréchet distance ε to P.*

Note that the source and destination of such a "shortest path" may not be graph nodes in G. We provide an incremental algorithm for computing such a shortest matching path. Our algorithm computes a distance function on all edges of the free space. We also prove properties of this distance functions which may be of independent interest.

3.1 Algorithm

Any path Q in G with $\delta_F(P, Q) \leq \varepsilon$ corresponds to a P-monotone path π in free space $D_\varepsilon \subseteq G \times P$. A shortest such path Q then corresponds to a shortest P-monotone path π in D_ε, where the length of π is only measured along G, i.e., in the horizontal direction. Our algorithm follows a dynamic programming approach that combines the computation of paths in the free space diagram with shortest path computations.

We define a function $\varphi : G \times P \to \mathbb{R}$ such that $\varphi(g, p) = \min_Q |Q|$, where Q ranges over all paths in G ending at g such that $\delta_F(P[p_0, p], Q) \leq \varepsilon$, and $|Q|$ denotes the length of Q. If no such path to (g, p) exists, then $\varphi(g, p) = \infty$. In particular, $\varphi(g, p) = \infty$ for $(g, p) \notin D_\varepsilon$. We have that $\varphi(g, p) = \min_\pi |\pi|$, where π ranges over all P-monotone paths in D_ε that end at (g, p), and the length $|\pi|$ is measured along G only. We call π a *G-shortest path*, or *shortest path* for short. Thus, φ captures the length of G-shortest paths in free space. Our algorithm computes φ slice-by-slice over $G \times P$, with the goal to compute $\varphi(g, p_N)$ for some $g \in G$. Observe that a G-shortest path π has to be monotone in each cell of $G \times P$. Therefore, it suffices to compute φ on the vertical and horizontal edges of $G \times P$. In each slice of $G \times P$, we perform a Bellman-Ford inspired computation to propagate φ between the vertical edges by relaxing along the horizontal edges.

For a vertical edge defined by $v \in V$ and an edge (p_i, p_{i+1}) of the path, let $\varphi_{v,i}(t) : [0, 1] \to \mathbb{R}$ be defined by $\varphi_{v,i}(t) = \varphi(v, p_i(t))$. For a horizontal edge defined by $e \in E$ and a vertex p_i of the path, let $\varphi_{e,i} : [0, 1] \to \mathbb{R}$ be defined by $\varphi_{e,i}(s) := \varphi(e(s), p_i)$. Note that for each (undirected) edge $(u, v) \in E$, we only store one φ-function, say, $\varphi_{(u,v),i}(s)$, since $\varphi_{(v,u),i}(s) = \varphi_{(u,v),i}(1 - s)$.

Algorithm 1: Shortest Among Fréchet-Matching Paths

1 Initialize $\varphi_{v,i}(t) = \varphi_{e,i}(s) = \infty$ for all v, e, i, s, t.
2 **forall** $v \in V$ *with* $||v - p_0|| \leq \varepsilon$ **do**
3 $\varphi_{v,0}(0) = 0$
4 **forall** $e \in E$ *and* $s \in [0, 1]$ *with* $||e(s) - p_0|| \leq \varepsilon$ **do**
5 $\varphi_{e,0}(s) = 0$
6 **for** $i = 0, \ldots, N$ **do** *// Compute slices*
7 **forall** $v \in V$ *and* $t \in [0, 1]$ *with* $||v - p_i(t)|| \leq \varepsilon$ **do**
  // Initialize vertical edges
8 $\varphi_{v,i}(t) = \min\{\varphi_{v,i}(a^i_v), \min_{u \in Adj(v)} \min_{s \in [0,1]} \{\varphi_{(u,v),i}(s) + (1 - s)||u - v||\}\}$
9 **while** *there exist edges $e \in E$ to be relaxed* **do**
  // Compute vertical edges in slice i
10 **forall** $e = (u, v) \in E$ *and* $t \in [0, 1]$ **do**
  // Relax edge e (both directions)
11 $\varphi_{v,i}(t) = \min\{\varphi_{v,i}(t), \varphi_{u,i}(\max\{t, a^i_v\}) + ||u - v||\}$
12 $\varphi_{u,i}(t) = \min\{\varphi_{u,i}(t), \varphi_{v,i}(\max\{t, a^i_u\}) + ||u - v||\}$
13 **forall** $e \in E$ **do**
  // Compute horizontal edges in level $i + 1$
14 Compute $\varphi_{e,i+1}(s)$ according to Lemma 3.3.

Lemma 3.1 (Vertical Monotonicity). *The function $\varphi_{v,i}(t)$ is monotone non-increasing for $t \in [a^i_v, b^i_v]$.*

Proof. Observe that $[a^i_v, b^i_v]$ corresponds to the intersection of the freespace D_ε with the vertical edge. Since φ measures the length of paths in D_ε in the G-direction only, paths can move in the vertical direction without increasing in length. □

In particular, we note that a direct consequence of the above lemma is the fact that the minimum of this edge is attained at a^i_v: $\varphi_{v,i}(a^i_v) \leq \varphi_{v,i}(t)$ for all $t \in [a^i_v, b^i_v]$.

Our dynamic programming algorithm, Algorithm 1, is based on the reachability propagation introduced by Alt and Godau to compute the Fréchet distance [3]. Instead of propagating binary reachability information from cell to cell, we propagate function values for φ along vertical and horizontal edges of $G \times P$. We will see in Lemma 3.4 and Lemma 3.5 that φ is piecewise linear on a vertical or horizontal edge. We therefore store each $\varphi_{v,i}(t)$ and $\varphi_{e,i}(s)$ as a list of linear pieces. Updates such as the ones in lines 8, 11, 12 then take linear time in the length of the lists. The condition in line 9 is true if there exists an edge $e = (u, v)$ such that $\varphi_{v,i}(t)$ or $\varphi_{u,i}(t)$ are updated in lines 11 and 12.

3.2 Properties

Algorithm 1 is based on the recursive formulas given in Lemma 3.2 and Lemma 3.3.

Lemma 3.2 (Compute Vertical Edges). *Consider a vertical edge $v \times (p_i, p_{i+1})$ for any $v \in V$ and $i \in \{0, \ldots, N\}$. Then, for any $t \in [0, 1]$, we have:*

- *If* $||v - p_i(t)|| > \varepsilon$ *then* $\varphi_{v,i}(t) = \infty$.
- *If* $||v - p_i(t)|| \leq \varepsilon$ *then* $\varphi_{v,i}(0) = 0$, *and for* $t \in (0, 1]$:

$$\varphi_{v,i}(t) = \min \left\{ \begin{array}{l} \varphi_{v,i}(a_v^i), \\ \min\limits_{u \in Adj(v)} \varphi_{u,i}(\max\{t, a_u^i\}) + ||u - v||, \\ \min\limits_{u \in Adj(v)} \min\limits_{s \in [0,1]} \{\varphi_{(u,v),i}(s) + (1-s)||u - v||\} \end{array} \right\}$$

Proof. The first two equalities follow directly from the definition of φ. To prove the third equality, consider a shortest monotone path π in D_ε ending at $(v, p_i(t))$ for some $t \in [0, 1]$. The last segment of π connects to one of the following:

(1) The bottom-most feasible point, a_v^i, on the same vertical edge $v \times (p_i, p_{i+1})$,
(2) a point on a vertical edge $u \times (p_i, p_{i+1})$ for a vertex $u \in V$ adjacent to v, or
(3) a point on a horizontal edge $(u, v) \times p_i$ for a vertex $u \in V$ adjacent to v.

A shortest monotone path always exists for which this last segment is a straight-line segment. The three cases correspond to the three values minimized over in the theorem. Measuring lengths in G, we observe that vertical paths in D_ε have length zero. Hence, the length of the corresponding path in G in the first case is $\varphi_{v,i}(a_v^i)$, the lengths in the other cases minimize over all vertices u adjacent to v, and the value $\varphi_{v,i}(t)$ is the minimum of these three lengths. In the second case, the projection of π onto G traverses the entire edge (u, v), which contributes length $||u - v||$. The third case minimizes over all possible connections to the horizontal edge $e \times p_i$ where $e = (u, v)$. A segment connecting $(v, p_i(t))$ to a point $(e(s), p_i)$ has length $(1 - s)||u - v||$, assuming e is parameterized by $e(s) = (1 - s)u + sv$. □

Lemma 3.3 (Compute Horizontal Edges). *Consider a horizontal edge* $e \times p_{i+1}$ *for any* $e = (u, v) \in E$ *and* $i \in \{0, \ldots, N\}$. *Then, for any* $s \in [0, 1]$ *we have:*

- $\varphi_{e,0}(s) = \left\{ \begin{array}{ll} 0, & \text{if } ||e(s) - p_0|| \leq \varepsilon \\ \infty, & \text{else} \end{array} \right.$
- *If* $||e(s) - p_{i+1}|| > \varepsilon$, *then* $\varphi_{e,i+1}(s) = \infty$.
- *If* $||e(s) - p_{i+1}|| \leq \varepsilon$, *then*

$$\varphi_{e,i+1}(s) = \min \left\{ \begin{array}{l} \varphi_{u,i}(b_u^i) + s||u - v||, \\ \varphi_{v,i}(b_v^i) + (1-s)||u - v||, \\ \min\limits_{s' \in [0,1]} \{\varphi_{e,i}(s') + |s - s'| \cdot ||u - v||\} \end{array} \right. \quad (1)$$

Proof. The first two equalities follow directly from the definition of φ. It remains to prove the last equality given in Equation (1). Consider a shortest monotone path π in D_ε ending at $(e(s), p_{i+1})$. The last segment of π connects to one of the following:

(1) a point on the vertical edge $u \times (p_i, p_{i+1})$,
(2) a point on the vertical edge $v \times (p_i, p_{i+1})$, or
(3) a point on the horizontal edge $e \times p_i$.

These cases correspond to the three values minimized over in Equation (1). By definition, $\varphi_{e,i+1}(s)$ is the minimum of these three values. In the first case, the last segment of π connects to b_u^i (or to a point below it on $u \times (p_i, p_{i+1})$ with the same value of φ), since $\varphi_{v,i}(t)$ is monotone decreasing; the length of this segment is $s||u - v||$. The second case is analogous to the first case, for the other vertical edge in the free space cell. The third case minimizes over all possible connections to the horizontal edge $e \times p_i$. A segment connecting $(e(s), p_{i+1})$ to a point $(e(s'), p_i)$ has length $|s - s'| \cdot ||u - v||$. □

The following two lemmas will be used to prove correctness of Algorithm 1 in Theorem 3.6.

Lemma 3.4 (Vertical Function Complexity). *Let* $v \times (p_i, p_{i+1})$ *be a vertical edge. Then, for* $t \in [a_v^i, b_v^i]$, *the function* $\varphi_{v,i}(t)$ *is piecewise constant and monotone non-increasing with complexity* $O(n)$.

Proof. If $t \in [a_v^1, b_v^i]$, then $\phi_{v,i}(t) \leq \phi_{v,i}(a_v^i)$ since the path from $p(a_v^1)$ to $p(t)$ has length zero in G. The endpoints of each constant piece in $\varphi_{v,i}(t)$ can only be lower endpoints a_u^i of the free space on vertical edges $u \times (p_i, p_{i+1})$, for any $u \in V$. Hence, the complexity is $O(n)$. □

Lemma 3.5 (Horizontal Function Complexity). *Let* $e \times p_i$ *be a horizontal edge. Then, for* $s \in [c_e^i, d_e^i]$, *the function* $\varphi_{e,i}$ *is a piecewise-linear function, where each piece is of slope* $||e||$, $-||e||$ *or zero. Note that such a function is necessarily* $||e||$*-Lipschitz. Furthermore, the complexity of* $\varphi_{e,i}$ *is* $O(i)$.

Proof. We prove this claim by induction on i. By definition, we know that $\varphi_{e,0}(s) = 0$ for all $s \in [c_e^0, d_e^0]$. As a consequence of Lemma 3.3, we have that $\varphi_{e,i+1}$ is the lower envelope of a linear function with slope $||e||$, a linear function with slope $-||e||$, and $\min_{s' \in [0,1]}\{\varphi_{e,i}(s') + |s - s'| \cdot ||e||\}$. Since, by inductive hypothesis, $\varphi_{e,i}$ is piecewise linear, where each piece is of slope $||e||$, $-||e||$ or zero, the minimum of the last term is attained as follows: If $s < c_e^i$ then $s' = c_e^i$ and $c_e^i \leq s \leq d_e^i$, then $s' = s$, and if $d_e^i \leq s$, then $s' = d_e^i$. Hence, the function $\varphi_{e,i+1}$ consists of a translated copy of $\varphi_{e,i}$ with at most two additional linear pieces at each end. Therefore, we know that $\varphi_{e,i+1}$ has the desired structure, and its complexity is $O(i)$. □

We prove the correctness and analyze the runtime of Algorithm 1 in the following theorem:

Theorem 3.6 (Correctness and Time Complexity). *Algorithm 1 computes the length of a shortest matching path in* $O(N(kmn + mN))$ *time and* $O(n^2 + mN)$ *space, where* k *is the number of edges in the shortest matching path in* G.

Proof. For each vertical edge $v \times (p_i, p_{i+1})$ (and each horizontal edge $e \times p_i$), we compute $\varphi_{v,i}$ (and $\varphi_{e,i}$, respectively). The time for initialization (lines 1-5) is $O(n + m)$. From Lemma 3.4, we know that each $\varphi_{v,i}$ has complexity $O(n)$, and from Lemma 3.5, that each $\varphi_{e,i}$ has complexity $O(i)$. We use these discrete representations of $\varphi_{v,i}$ and $\varphi_{e,i}$ throughout the algorithm. Since the algorithm computes one slice at a time, we only need to store $\varphi_{v,i}$ and $\varphi_{e,i}$ for only one slice. Hence, the total storage complexity is $O(n^2 + mN)$.

The correctness of the algorithm follows from Lemma 3.2 and Lemma 3.3. In particular, lines 7-12 are based on the recursive formula given in Lemma 3.2. All $\varphi_{v,i}$ on vertical edges $v \times (p_i, p_{i+1})$ are initialized in Lines 7-8 with values from the bottom horizontal edge. Then lines 9-12 perform a Bellman-Ford shortest path propagation across all vertical edges in slice i. We continue the while loop in line 9 as long as at least one $\varphi_{v,i}$ (or $\varphi_{u,i}$) was updated in

lines 11-12. Hence, the number of iterations of the while loop is $k+1$ (once the shortest paths are found, no improvements will be made). After all, $\varphi_{v,i}$ have been computed in slice i, all $\varphi_{e,i+1}$ are computed from the vertical edges and the horizontal edges in level i, according to Lemma 3.3. Lines 7-8 take $O(n^2)$ time, lines 11-12 take $O(n)$ time, and line 14 takes $O(i)$ time. Hence, lines 6-14 of the algorithm take time $O(N(n^2 + kmn + mN))$, and thus the total runtime is $O(N(kmn + mN)$. □

Remark 1. *As stated, Algorithm 1 enforces monotonicity on P but not on edges of $G = (V, E)$. If desired, the algorithm can be modified to enforce monotonicity on the edges in E as follows: The cell complex would need to be defined using* directed *edges E', where undirected edges in E are represented using two directed edges. The propagations according to Lemma 3.2 need to use adjacency lists $Adj(v) = \{(u, v) \mid (u, v) \in E'\}$. The horizontal propagation in Lemma 3.3 needs to be adjusted, by replacing equation (1) with $\varphi_{e,i+1}(s) = \min\{\varphi_{u,i}(b_u^i) + s||u - v||, f_{e,i}(s)\}$. Here, $f_{e,i}(s) = 0$ if $c_e^i \le s \le d_e^i$, and $f_{e,i}(s) = s - d_e^i$ if $d_e^i < s$. This formula models monotone propagation in the same way as in Alt and Godau [3], just that in addition to reachability we propagate the length of a G-shortest path.*

4 MATCH TO CONCATENATION OF SHORTEST PATHS

In this section, we are interested in matching the path P to a concatenation of shortest paths in G. We consider two variants of the problem, one that minimizes the number of shortest paths that are concatenated, the other that minimizes the Fréchet distance δ_F.

Problem 2 (Min-k). *Given a parameter $\varepsilon \ge 0$, find a path Q in G that is a concatenation of the smallest number of shortest paths in G, such that $\delta_F(P, Q) \le \varepsilon$.*

Problem 3 (Min-ε). *Given a parameter $k \ge 1$, find a path Q in G that is a concatenation of at most k shortest paths in G, such that the Fréchet distance between P and Q is minimized.*

In this section, we assume that the paths in G must begin and end at a vertex. We begin by exploring the case where $k = 1$ in Section 4.1, then consider the more general case in Section 4.2 and Section 4.3. Allowing paths to start or end anywhere on an edge makes the problem considerably harder. We sketch approximation algorithms for this case in Section 5.

4.1 Matching to Shortest Paths

As a warm-up, we consider the min-ε problem for the case where $k = 1$, i.e., we wish to find a shortest path Q in G that minimizes the Fréchet distance to P, among all shortest paths in G that start and end at vertices in V.

First, we compute an implicit representation of all shortest paths between all pairs of vertices in V, by running Dijkstra's shortest path algorithm for each $s \in V$ as a source vertex. Shortest paths with a common start vertex are stored in a shortest path directed acyclic graph (DAG); note that while algorithms usually assume uniqueness of shortest paths and store only a tree, we wish to keep all possible shortest paths since we must store all of them in order to consider their Fréchet distance to P. The shortest path DAGs are computed and stored for each $s \in V$ as a source vertex, in total in $O(n(m + n \log n))$ time and $O(n^2)$ space.

Then, we need to compute the Fréchet distance between P and each shortest path, in order to identify the minimum distance. We batch these computations by computing the Fréchet distance between a path and the entire shortest path DAGs. The following lemma and the resulting corollaries show that distances between shortest path prefixes and prefixes of P can be computed efficiently in a batched manner. We state these results for a general DAG with a single root – We note that a more general version of Lemma 4.1 (which is between two DAGs) has already been observed in [10]. We include the proof here for our version for completeness.

Lemma 4.1. *Let $T = (V_T, E_T)$ be a DAG with a root r and $|E_T| = m_T$. Let P be a polygonal path with vertices $p_0, p_1, \ldots, p_N$. A path in T from the root to a leaf, that has the smallest Fréchet distance to P, can be computed in $O(m_T N \log(m_T + N))$ time.*

Proof. This is a simple modification of Alt and Godau's computation of the Fréchet distance for two polygonal paths [3], and a special case of the map-matching setting considered in [2]. For fixed $\varepsilon > 0$, we compute the free space in $T \times P$. We then propagate reachability information from (r, p_0) in dynamic programming fashion in this free space. Starting with filling reachability information in $r \times P$, we then propagate the reachability monotonically across both T and P, traversing T in an order determined by a topological sort of T, and P from p_0 to p_N. For each edge $(u, v) \in E_T$, the reachable points in $(u, v) \times P$ are computed by straight-forward propagation from the reachable points in $u \times P$. But since v may have multiple incoming edges, the reachability information for $v \times P$ is then computed as the union of all the propagated reachability information for all $(u, v) \in E_T$. It takes time and space $O(m_T N)$ to solve the decision problem. With parametric search [2, 3], the path in T from the root to a leaf, that has the smallest Fréchet distance to P, can be found in $O(m_T N \log(m_T + N))$ time. □

For fixed $\varepsilon > 0$, the algorithm described in the proof of Lemma 4.1 does in fact compute reachability information for *all* paths starting in the root of T and *all* prefixes of P:

Corollary 4.2. *Let $T = (V_T, E_T)$ be a DAG with a root r and $|E_T| = m_T$, and let $\varepsilon > 0$. In $O(m_T N)$ time, one can compute for all points $g \in T$ and $p \in P$ whether there exists a path $Q_{r,g}$ in G from r to g such that $\delta_F(Q_{r,g}, P[p_0, p]) \le \varepsilon$.*

And in fact, reachability can be computed efficiently if *either* the start point of the path P *or* the start point of a corresponding path in T is allowed to vary along an edge:

Corollary 4.3. *Let $T = (V_T, E_T)$ be a DAG with root r, and let $|E_T| = m_T$ and $\varepsilon > 0$. The following can be computed in $O(m_T N)$ time:*

(i) *The existence of a path $Q_{r,g}$ in the graph G from r to g such that $\delta_F(Q_{r,g}, P[x, p]) \le \varepsilon$, for each triple $g \in T$, $p \in P$, and $x \in (p_0, p_1)$.*
(ii) *The existence of a path $Q_{x,g}$ in G from x to g such that $\delta_F(Q_{x,g}, P[p_0, p]) \le \varepsilon$, for each triple $g \in T$, $p \in P$, and $x \in (r, v)$, where (r, v) is the only edge incident on the root.*

Proof. For (i), a simple modification of the reachability initialization step in the proof of Lemma 4.1 results in computing reachability

from (r, x) for any $x \in (p_0, p_1)$. For (ii), if $g \notin (r, v)$, then a simple modification of the reachability initialization step in the proof of Lemma 4.1 results in computing reachability from any $x \in (r, v)$. If both x and g are on the same edge (r, v), then we compute the reachability in $(r, v) \times P$ directly. □

We apply Lemma 4.1 to the shortest path DAG T_s for each start vertex $s \in V$. We compute a shortest path in T_s that has the smallest Fréchet distance to P in $O(mN \log(m + N))$ time. Repeating this for each source vertex, and accounting for running Dijkstra's algorithm in the beginning, results in a total runtime of $O(nmN \log(m + N))$ and $O(n(n + N))$ space. We summarize our result:

Theorem 4.4 (Matching to Shortest Path). *A path Q that minimizes the Fréchet distance to P, among all shortest paths in G that start and end at vertices in V, can be computed in $O(nmN \log(m+N))$ and $O(n(n + N))$ space.*

4.2 The Min-k Problem

In this section, we solve the min-k problem: For fixed $\varepsilon \geq 0$, we wish to find a path Q that is a concatenation of the smallest number of shortest paths in G such that $\delta_F(P, Q) \leq \varepsilon$. We require that all shortest paths start and end at vertices in V.

Auxiliary Graph. We build an auxiliary graph $G' = (V', E')$ as follows. The set of vertices are ordered pairs of a vertex in V and a vertex in P, formally $V' = \{\langle v, p_i\rangle \mid v \in V, i \in \{0, \ldots, N\}\}$. There is an edge between $\langle u, p_i\rangle$ and $\langle v, p_j\rangle$, if there is a shortest path Q in G from u to v such that the Fréchet distance between $P[i, j]$ and Q is at most ε. Formally, $E' = \{(\langle u, p_i\rangle, \langle v, p_j\rangle) \mid 0 \leq i \leq j \leq N$, and there is a shortest path Q from u to v in G such that $\delta_F(Q, P[i, j]) \leq \varepsilon\}$. We have $|V'| = nN$ and $|E'| \in O(n^2N^2)$.

This auxiliary graph can be constructed as follows: We compute all shortest path DAGs T_u by running Dijkstra's shortest path algorithm for every $u \in V$. For fixed $u \in V$ and $i \in \{0 \leq i \leq N\}$, we use Corollary 4.2 to compute the reachability information. For each $v \in V$ and $i \leq j \leq N$, we can then read off whether there exists a shortest path in G from u to v such that $\delta_F(Q_{u,v}, P[i, j]) \leq \varepsilon$. This determines whether $(\langle u, p_i\rangle, \langle v, p_j\rangle) \in E'$. The runtime is $O(n(m + n \log n))$ to compute all shortest path DAGs, $O(mN)$ to compute the edges for fixed u and i, and hence $O(n(mN^2 + n \log n))$ time total to compute E'.

Algorithm. We can now solve our problem by finding a shortest path in G', starting at any vertex $\langle u, p_0\rangle$ for any $u \in V$ and ending at any vertex $\langle v, p_N\rangle$. We connect a super-source $\hat{s}$ to all $\langle u, p_0\rangle$ for any $u \in V$. Since the length of the path is determined by the number of edges, we can compute such shortest paths by running breadth-first search from $\hat{s}$ in time $O(|V'| + |E'|) = O(n^2N^2)$. The total runtime is dominated by the time $O(n(mN^2 + n \log n))$ to compute the auxiliary graph. We summarize our result in the following theorem:

Theorem 4.5 (Min-k). *For fixed $\varepsilon \geq 0$, a path Q that is a concatenation of the smallest number of shortest paths in G such that $\delta_F(P, Q) \leq \varepsilon$ can be computed in $O(n(mN^2 + n \log n))$ time and $O(n^2N^2)$ space.*

4.3 Min-ε

Next, we show how we can use our solution for the min-k problem described in Section 4.2, in order to develop a solution for the min-ε problem. For fixed $k \geq 2$, we wish to find a path Q that is a concatenation of at most k shortest paths in G such that $\delta_F(P, Q)$ is minimized. Again, we require that all shortest paths start and end at vertices in V.

Let $k \geq 2$ be fixed. We modify the algorithm described in Section 4.2 to serve as a decision procedure for a given $\varepsilon \geq 0$: Return true if a shortest path exists of length $\leq k$, and false otherwise. We optimize ε by performing a binary search on a superset of the *critical values* for ε, which are values for which solutions to the decision procedure changes combinatorially. These combinatorial changes are caused by combinatorial changes in the free space diagram for a shortest path Q and P; see Alt and Godau [3]. We consider all possible critical values within each free space cell and across pairs of free space cells. Possible critical values are those ε for which $a^i_u = b^j_v$ or $c^i_e = d^j_e$ for $u, v \in V$, $e \in E$, and $j = i$ or $j = i + 1$. There are $O(n^2N + N^2n)$ such values that constitute a superset of the combinatorial changes that affect our decision procedure. We sort these critical values in $O((n^2N + N^2n) \log(n + N))$ time and perform a binary search using the decision procedure, which results in a total runtime of $O(n(mN^2 + n \log n) \log(n + N))$. We summarize our result as follows.

Theorem 4.6 (Min-ε). *For fixed $k \geq 0$, a path Q that is a concatenation of at most k shortest paths in G such that $\delta_F(P, Q)$ is minimized, can be computed in $O(n(mN^2 + n \log n) \log(n + N))$ time and $O(n^2N^2)$ space.*

5 APPROXIMATION ALGORITHM FOR k-SP WITHOUT VERTEX-CONSTRAINT

In this section, we consider the more general version of the k-shortest path problem where there is no vertex-constraint. Let $|G|$ denote the underlying space of G, consisting of all points, including those in the interior of edges, in G. We say that a path $Q \subset |G|$ in G is a *k-SP* if it can be partitioned into k consecutive pieces $Q = Q_1 \circ Q_2 \cdots \circ Q_k$ such that each Q_i is a shortest path between its two endpoints in $|G|$. Let $P = \{P_1, \ldots, P_k\}$ be a *k-partitioning* of the underlying space $|P|$ of the polygonal curve P; that is, $|P| = P_1 \circ P_2 \cdots \circ P_k$ with P_i and P_j disjoint in their interior for all $i \neq j$. We say that the graph G has a *(k, ε)-matching for P* if there exists a k-SP $Q = Q_1 \circ Q_2 \cdots \circ Q_k$ and a k-partitioning $P = \{P_1, P_2, \ldots P_k\}$ of P such that for any $i \in [1, k]$, the Fréchet distance is bounded: $\delta_F(Q_i, P_i) \leq \varepsilon$. (Note that this also implies that $\delta_F(Q, P) \leq \varepsilon$.) We refer to endpoints of each path in Q and in P as *break-points*. Note that the break-points could lie in the interior of edges in G or in P.

Problem 4. *Given k and $\varepsilon > 0$, the goal is to decide whether there exists a k-SP $Q = Q_1 \circ Q_2 \cdots \circ Q_k$ and a k-partitioning $P = \{P_1, P_2, \ldots P_k\}$ of P such that for any $i \in [1, k]$, the Fréchet distance is bounded: $\delta_F(Q_i, P_i) \leq \varepsilon$.*

This general version of the problem seems to be much more challenging. For example, consider Figure 3. Suppose we already know that point p_0 should be matched to some point on edge $e_1 = (u_1, u_2)$, and the last point p_N should be matched to some point on edge $e_2 = (w_1, w_2)$. Let π_1 be a shortest path from u_1 to w_1,

and π_2 be a shortest path from u_2 to w_2. We need to compute a shortest path starting in some $u \in e_1$ and ending in some $w \in e_2$ whose Fréchet distance to P is at most ε. However, whether the path $u \rightsquigarrow \pi_1 \rightsquigarrow w$ or the path $u \rightsquigarrow \pi_2 \rightsquigarrow w$ is shortest depends on the positions of *both* u and w. Hence, the end point w depends on the starting point u, which makes developing a dynamic programming strategy challenging.

In this section, we focus on approximation algorithms. We say that an algorithm is an (α, β)-*approximation* for the (k, ε)-matching problem, if it computes an $(\alpha k, \beta\varepsilon)$-matching for the path P whenever there exists a (k, ε)-matching for P in G. In what follows, we describe such an approximation algorithm, where the input satisfies the following mild assumption:

Assumption-R: For the optimal k-SP Q, there is no U-turn in the interior of an edge. Equivalently, for a break-point s_i connecting shortest path pieces Q_i and Q_{i+1}, if s_i is in the interior of edge $e = (u, v)$, then $Q_i \cap Q_{i+1} \cap e = \{s_i\}$.

Theorem 5.1 (Approximation Theorem). *Let P be a polygonal path and $G = (V, E)$ be a graph satisfying* Assumption-R*, there is a $(2, 2)$-approximation algorithm for the (k, ε)-matching problem with running time $O(nmN^2)$, where $n = |V|, m = |E|$, and $N = |P|$.*

To prove Theorem 5.1, we solve a version of the k-matching problem for which we require that all break-points in the k-SP Q, *other than the start point and endpoint*, have to be vertices from the graph G. We call this the *G-restricted (k, ε)-matching problem for P*. Theorem 5.1 follows immediately from the two propositions below. The proofs of these propositions are in the full version of this paper.

Proposition 5.2. *If there is a (k, ε)-matching between P and G, where the input satisfies* Assumption-R*, then there is a G-restricted $(2k, \varepsilon)$-matching between P and G.*

Proposition 5.3. *Given a polygonal path P and a graph $G = (V, E)$, there is a $(1, 2)$-approximation algorithm for the G-restricted (k, ε)-matching problem whose running time is $O(nmN^2)$, where $n = |V|, m = |E|$, and $N = |P|$.*

6 DISCUSSION AND FUTURE WORK

In this paper, we present the first algorithms for map matching where we restrict possible matching candidates to consist of shortest paths in the graph. This variant arises naturally given the nature of GPS data, as many routing algorithms prefer certain types of paths; shortest paths are natural in this setting, but similar algorithms could be investigated in more complex settings, such as least costs roads or shortest travel time paths.

We are able to give exact algorithms for the case where shortest paths go between vertices in the graph; however, these techniques will not generalize to give exact algorithms when the shortest paths begin or end in the middle of an edge. Even our approximation for this setting does not allow the two consecutive shortest paths to reverse in the middle of an edge. Further investigation and extensions of these algorithms, as well as improved running time, are perhaps the next natural area of investigation in this work.

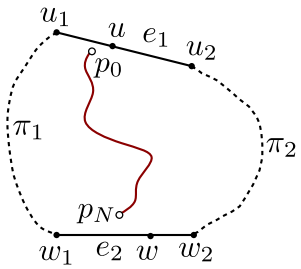

Figure 3: The path $u \rightsquigarrow \pi_1 \rightsquigarrow w$ or the path $u \rightsquigarrow \pi_2 \rightsquigarrow w$ may be shortest, depending on the positions of u and w, where w depends on u.

Acknowledgements

The authors would like to acknowledge the generous support of the National Science Foundation under grants IIS-1319944, CCF-1054779, CCF-1614562, CCF-1618605, CCF-1618247, and CCF-1618469.

REFERENCES

[1] Mahmuda Ahmed and Carola Wenk. 2012. Constructing Street Networks from GPS Trajectories. In *Algorithms–ESA 2012*. Springer, 60–71.
[2] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. 2003. Matching Planar Maps. *Journal of Algorithms* 49 (2003), 262–283.
[3] Helmut Alt and Michael Godau. 1995. Computing the Fréchet Distance between Two Polygonal Curves. *Int. J. Comput. Geometry Appl.* 5 (1995), 75–91.
[4] Theo A Arentze and Harry J.P Timmermans. 2004. A learning-based transportation oriented simulation system. *Transportation Research Part B: Methodological* 38, 7 (2004), 613 – 633. https://doi.org/10.1016/j.trb.2002.10.001
[5] Michael Balmer, Kay Axhausen, and Kai Nagel. 2006. Agent-Based Demand-Modeling Framework for Large-Scale Microsimulations. *Transportation Research Record: Journal of the Transportation Research Board* 1985 (2006), 125–134. https://doi.org/10.3141/1985-14
[6] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On Map-matching Vehicle Tracking Data. In *Proc. 31st VLDB Conf.* 853–864.
[7] Wilner Ciscal-Terry, Mauro Dell'Amico, Natalia Selini Hadjidimitriou, and Manuel Iori. 2016. An analysis of drivers route choice behaviour using GPS data and optimal alternatives. *Journal of Transport Geography* 51 (2016), 119–129.
[8] Maurice Fréchet. 1906. Sur quelques Points du Calcul Fonctionnel. *Rendiconti del Circolo Matematico di Palermo* 22, 1 (1906), 1–74.
[9] Amin Gheibi, Anil Maheshwari, and Jörg-Rüdiger Sack. 2016. Minimizing Walking Length in Map Matching. In *Topics in Theoretical Computer Science: The First IFIP WG 1.8 International Conference*, M. T. Hajiaghayi and M. R. Mousavi (Eds.). 105–120.
[10] Sariel Har-Peled and Benjamin Raichel. 2014. The Fréchet Distance Revisited and Extended. *ACM Trans. Algorithms* 10, 1 (Jan. 2014), 3:1–3:22.
[11] Mithilesh Jha, Samer Madanat, and Srinivas Peeta. 1998. Perception updating and day-to-day travel choice dynamics in traffic networks with information provision. *Transportation Research Part C: Emerging Technologies* 6, 3 (1998), 189 – 212. https://doi.org/10.1016/S0968-090X(98)00015-1
[12] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-Matching for Low-Sampling-Rate GPS Trajectories. In *Proc. 17th ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems*. 352–361.
[13] Paul Newson and John Krumm. 2009. Hidden Markov Map Matching Through Noise and Sparseness. In *Proc. 17th ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems*. 336–343.
[14] Carlo Prato and Shlomo Bekhor. 2006. Applying Branch-and-Bound Technique to Route Choice Set Generation. *Transportation Research Record: Journal of the Transportation Research Board* 1985 (2006), 19–28. https://doi.org/10.3141/1985-03
[15] Mohammed Quddus and Simon Washington. 2015. Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies* 55 (2015), 328–339.
[16] Piotr Szwed and Kamil Pekala. 2014. An Incremental Map-Matching Algorithm Based on Hidden Markov Model. In *Artificial Intelligence and Soft Computing*. 579–590.
[17] Muhammad Reaz Uddin, Chinya Ravishankar, and Vassilis J. Tsotras. 2011. A System for Discovering Regions of Interest from Trajectory Data. In *Advances in Spatial and Temporal Databases*. Springer, 481–485.
[18] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, and Yan Huang. 2010. T-Drive: Driving Directions Based on Taxi Trajectories. In *Proc. 18th ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems*. 99–108.

[19] Lei Zhang, David M. Levinson, and Shanjiang Zhu. 2008. Agent-Based Model of Price Competition, Capacity Choice, and Product Differentiation on Congested Networks. *Journal of Transport Economics and Policy* 42, 3 (2008), 435–461.

[20] L. Zhu, J.R. Holden, and J.D Gonder. 2017. Trajectory Segmentation Map-Matching Approach for Large-Scale, High-Resolution GPS Data. *Transportation Research Record: Journal of the Transportation Research Board* 2645, 67-75 (2017).

[21] Shanjiang Zhu and David Levinson. 2015. Do People Use the Shortest Path? An Empirical Test of Wardrop's First Principle. *PLOS ONE* 10 (08 2015), 1–18. https://doi.org/10.1371/journal.pone.0134322