Map-Matching Using Shortest Paths

ERIN CHAMBERS, Saint Louis University
BRITTANY TERESE FASY, Montana State University
YUSU WANG, The Ohio State University
CAROLA WENK, Tulane University

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 → Design and analysis of algorithms**; *Shortest paths*; *Theory and algorithms for application domains.*

Additional Key Words and Phrases: Fréchet distance, graph matching, shortest paths

ACM Reference Format:
Erin Chambers, Brittany Terese Fasy, Yusu Wang, and Carola Wenk. 2018. Map-Matching Using Shortest Paths. *J. ACM* 37, 4, Article 111 (August 2018), 17 pages. https://doi.org/10.1145/1122445.1122456

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. To the best of our knowledge, 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

Authors' addresses: Erin Chambers, Saint Louis University, Department of Computer Science, 220 N. Grand Avenue, Saint Louis, Missouri, 63103, echambe5@slu.edu; Brittany Terese Fasy, Montana State University, School of Computing, 363 Barnard Hall, Bozeman, Montana, 59717, brittany@cs.montana.edu; Yusu Wang, The Ohio State University, Computer Science and Engineering Department, 2015 Neil Avenue, Columbus, Ohio, 43210, yusu@cse.ohio-state.edu; Carola Wenk, Tulane University, Department of Computer Science, 6823 St. Charles Avenue, New Orleans, Louisiana, 70118, cwenk@tulane.edu.

0004-5411/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

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 [18, 19]. Common approaches include the use of Fréchet distance variants [2, 6], ad-hoc incremental methods [6, Sec. 3], matching low-sampling-rate trajectories using spatial-temporal constraints [13], and hidden Markov models [15, 17]. 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. Recently, Gheibi et al. [10] 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.

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. 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 other distance measures such as the Hausdorff distance, since it takes the continuity of the curves and not simply distance between them into account.

In Section 3, we provide an algorithm to match P to the shortest possible path within some Fréchet distance in G. We prove properties of a distance function on the free space diagram, which is the main tool used to compute Fréchet distance this allows us to use an incremental algorithm, which in turn uses less space less space than alternatives [10]. 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 break-points between shortest paths lie on vertices of G, and these break-points are mapped 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-points can 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 systematic study of map-matching algorithms that consider a subset of paths with pre-defined properties in G to be matched to P. Our paper introduces a new perspective on map-matching and provides theoretical foundations for the practically relevant problem, where we consider a restricted set of path classes. We remark that an initial extended abstract of the present paper appeared in the 3rd International Workshop on Interactive and Spatial Computing (IWISC), 2018 [7]. The main difference lies in Section 5, which is a main technical component of the current paper, where no technical proofs were provided for the earlier extended abstract.

Discussion of our model. 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, 16, 20].

Furthermore, we note that it has been observed before that shorter trajectories are more likely to be shortest paths, as do very long routes where the trajectory tends to follow a single route

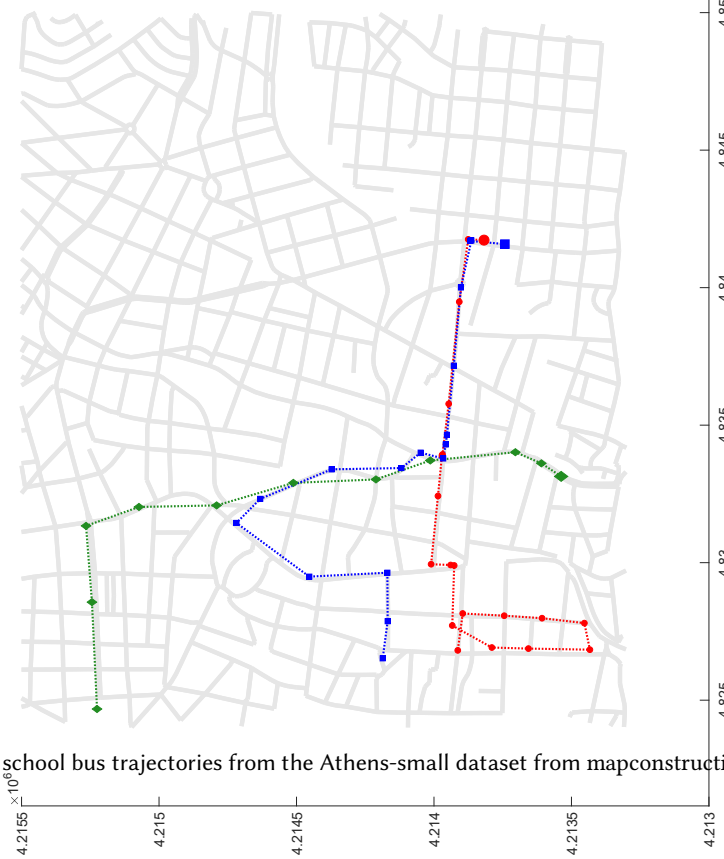


Fig. 1. Three school bus trajectories from the Athens-small dataset from mapconstruction.org.

like a highway for most of the distance. For example, in [21], the authors analyze GIS data from travel routes over an eight week time frame, finding that nearly 34% of all trips follow the shortest time path exactly; when they relax to allow the route to be "nearly" shortest, then about 40% of all trips follow shortest routes. Routes also tend to revert to something close to a shortest path in the evening [14], as do trips associated with work as opposed to other tasks like shopping [8]. In another work [12], the authors analyze several data sets and show that in fact routes stay relatively close to the shortest path, within an ellipse whose two foci are the start and end points of the trajectory, and even within this area most deviations are small.

While we measure the length of a path by its arclength using Euclidean distances in this paper, we remark that the weight of a shortest path can be based on other quantities, such as travel times.

We also emphasize that assuming that a path consists of k shortest paths is fundamentally broader in scope than assuming that it is a single shortest path. Indeed, in the extreme case, by choosing a sufficiently large k, any path can fit into our model: For example, any path consisting of n road segments can be considered as a concatenation of $k = n$ shortest paths.

In addition, several analyses suggest that breakpoints or anchors - key locations that attract a large number of trajectories - can be a natural way to decompose longer paths into routes that resemble optimal (or shortest) paths [12, 14]. This helps to support investigating algorithms in a model where longer paths can be viewed as concatenations of shortest paths. For example, in Fig. 1, bus route trajectories do not at all resemble shortest paths, but each can be broken into several subtrajectories that do follow a shortest route.

2 PRELIMINARIES

Let $G = (V, E)$ be a geometric graph with polygonal edges, and 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 $p \in G$.

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 [9]. 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.

In order to match the path to the graph, we consider the cell complex $G \times P$; see Fig. 2(a). 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 Fig. 2(b). 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_u^i and b_u^i. Where appropriate, we slightly abuse notation and use a_u^i to also identify the parameter t for which $p_i(t) = a_u^i$. In this way, we say $a_u^i \leq b_u^i$. Likewise, on a horizontal edge $(u, v) \times p_i$, we denote $c_u^i \leq d_u^i$ as the points for which $||c_u^i - p_i|| = ||d_u^i - p_i|| = \varepsilon$; see Fig. 2(b) for an example of a labeled free space cell.

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. That means, we are interested in finding the *shortest matching path*:

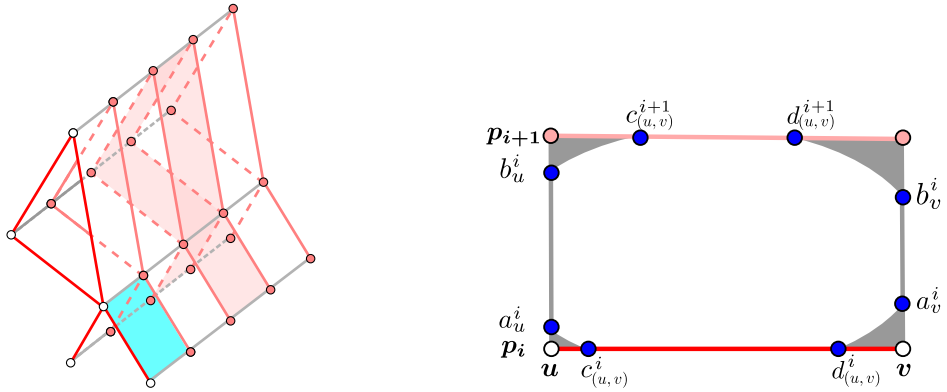
Problem 1 (Shortest Matching Path). *Given a parameter $\varepsilon > 0$ and a path P, find the shortest path in G that is within Fréchet distance ε to P.*

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



(a) Parameter space. (b) A free space cell.

Fig. 2. On the left, 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. A *cell* corresponds to two edges, one of the path and one on G, as shown in cyan. On the right, we illustrate one free space of a 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 $||g - p|| = \varepsilon$.

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 $\varphi_{(v,u),i}(s) = \varphi_{(u,v),i}(1 - s)$.

Lemma 3.1 (Vertical Monotonicity). *The vertical 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, and 12 then take linear time in the length of the lists. The

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 $\varphi_{v,i}(t) = \min\{\varphi_{v,i}(a_v^i), \min_{u \in Adj(v)} \min_{s \in [0,1]} \{\varphi_{(u,v),i}(s) + (1-s)||u - v||\}\}$
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 $\varphi_{v,i}(t) = \min\{\varphi_{v,i}(t), \varphi_{u,i}(\max\{t, a_v^i\}) + ||u - v||\}$
12 $\varphi_{u,i}(t) = \min\{\varphi_{u,i}(t), \varphi_{v,i}(\max\{t, a_u^i\}) + ||u - v||\}$
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.

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 $\varphi_{v,0}(0) = 0$, and for $t \in (0,1]$:*

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

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:*

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

$$\varphi_{e,i+1}(s) = \min \begin{cases} \varphi_{u,i}(b_u^i) + s||u - v||, \\ \varphi_{v,i}(b_v^i) + (1 - s)||u - v||, \\ \min\limits_{s' \in [0,1]} \{\varphi_{e,i}(s') + |s - s'| \cdot ||u - v||\} \end{cases} \tag{1}$$

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 three 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 the function $\varphi_{e,i}(s)$ is piecewise linear, for $s \in [c_e^i, d_e^i]$, 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 $c_e^i \leq s \leq d_e^i$, 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 $\varepsilon \geq 0$, find a path Q in G that is a concatenation of the smallest number of shortest paths in G, such that $\delta_F(P, Q) \leq \varepsilon$.*

Problem 3 (Min-ε). *Given a parameter $k \geq 1$, 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.

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 $\delta_F(Q_{r,g}, P[p_0, p]) \leq \varepsilon$.*

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) *For all points $g \in T$, $p \in P$, and $x \in (p_0, p_1)$ whether there exists a path $Q_{r,g}$ in G from r to g such that $\delta_F(Q_{r,g}, P[x, p]) \leq \varepsilon$.*
(ii) *If (r, v) is the only edge incident on the root, then it can be computed for all points $g \in T$, $p \in P$, and $x \in (r, v)$ whether there exists a path $Q_{x,g}$ in G from x to g such that $\delta_F(Q_{x,g}, P[p_0, p]) \leq \varepsilon$.*

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 V' are ordered pairs of a vertex in V and a vertex in P; formally, we write: $V' = \{\langle v, p_i \rangle \mid v \in V, i \in \{0, \ldots, N\}\}$. There is an edge in E' connecting $\langle u, p_i \rangle$ 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, we have $E' = \{(\langle u, p_i \rangle, \langle v, p_j \rangle) \mid 0 \leq i \leq j \leq N$, 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 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 $a_u^i = b_v^j$ or $c_e^i = d_e^j$ 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, by removing the vertex-constraint. Let $|G|$ denote the underlying space of G, comprising all points in G, including those in the interior of edges. We say that a path $Q \subset |G|$ 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 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 $P = \{P_1, P_2, \ldots P_k\}$ such that for any $i \in [1, k]$, the Fréchet distance is bounded: $\delta_F(Q_i, P_i) \leq \varepsilon$. (This also implies that $\delta_F(Q, P) \leq \varepsilon$). We refer to endpoints of each path in Q and in P as *break-point*s. Note that the break-points could lie in the interior of edges.

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 Fig. 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.

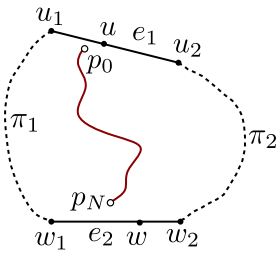


Fig. 3. The path $u \rightsquigarrow \pi_1 \rightsquigarrow w$ or the path $u \rightsquigarrow \pi_2 \rightsquigarrow w$ may be shortest, depending on the positions of u and w, where w depends on u.

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 $Q_i \cap Q_{i+1} \cap e = \{s_i\}$.

Remark: From a technical point of view, this assumption is important in proving Proposition 5.2 below. It may be possible to remove this assumption and still achieves a G-restricted $(2k, \varepsilon)$, as in Proposition 5.2 with a more complicated argument. However, we note that this assumption is in fact natural, or even necessary for a real route planning. Indeed, assuming each edge in the graph represents a road segment connecting two junction nodes, many jurisdictions do not allowed a U-turn between the junction nodes.

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.

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 $N = |P|$.*

Proof of Proposition 5.2. Assume G has a (k, ε)-matching and let $Q^* = Q_1 \circ Q_2 \circ \cdots \circ Q_k$ be the k-SP and $P = \{P_1, P_2, \cdots, P_k\}$ be the k-partition in this matching. We now show that we can modify Q to a G-restricted $2k$-SP $\widehat{Q}$ forming a $(2k, \varepsilon)$-matching with some $2k$-partition of P. Our modification *re-partitions* P and Q^*. Note that for any oriented path π, given an ordered sequence of points $\{\alpha_0, \ldots, \alpha_\ell\}$ along this path with start point α_0 and endpoint α_ℓ, it induces a unique partition $\pi[\alpha_0, \alpha_1] \circ \pi[\alpha_1, \alpha_2] \circ \cdots \circ \pi[\alpha_{\ell-1}, \alpha_\ell]$ of π. (Recall that $\pi[\alpha, \beta]$ is the subcurve of π between two points $\alpha \leq \beta$, meaning that α has a smaller preimage than β under the parametrization of π.) Hence, in what follows, we simply specify such sequences of break-points to describe (re-)partitioning of the paths Q^* and P.

Definition 5.4. Repartitions of Q^* and P induced by a sequence of break-points $S = \{s_0, \ldots, s_\ell\}$ and $\Pi = \{b_0, \ldots, b_\ell\}$ are called *valid* if for all $i \in [0, \ell - 1]$,

(i) each piece $Q^*[s_i, s_{i+1}]$ is a shortest path in G, and
(ii) $\delta_F(Q^*[s_i, s_{i+1}], P[b_i, b_{i+1}]) \leq \varepsilon$.

Let $S^* = \{s_0^*, s_1^*, \ldots, s_k^*\}$ be the sequence of break-points of the optimal k-SP Q^*; and $\Pi^* = \{b_0 = p_0^*, b_1^*, \ldots, b_{k-1}^*, b_k^* = p_{n_p}\}$ be the sequence of break-points for the optimal k-partition of P. We

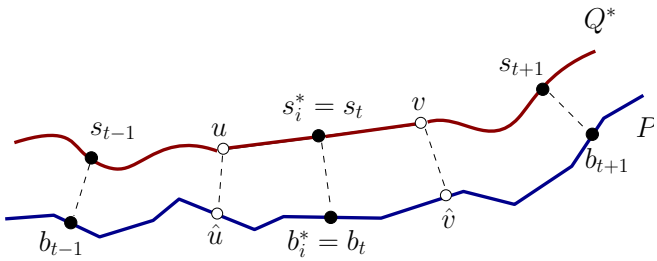


Fig. 4. Black dots are break-points, and dashed segments indicate aligned points.

now process each s_i^* in order from $i = 1$ to $i = k - 1$. In the beginning, $S_0 = S^*$ and $\Pi_0 = P^*$. In the i-th iteration, we obtain S_i from S_{i-1} such that $|S_i| \leq |S_{i-1}| + 1$, and we obtain Π_i from Π_{i-1} such that $|\Pi_i| \leq |\Pi_{i-1}| + 1$. We also maintain the invariant that the partitions S_i and Π_i are valid.

Specifically, in the ith iteration, suppose the break-point s_i^* from the optimal k-SP is still present in S_{i-1}. Assume $s_i^* \in e = (u, v) \in E$. By Assumption-R, the entire edge (u, v) must be covered by the path Q^* and $Q^*[u, v] = (u, v)$. Let $S_{i-1} = \{s_0, \ldots, s_\ell\}$ and $\Pi_{i-1} = \{b_0, \ldots, b_\ell\}$. Since S_{i-1} and Π_{i-1} are valid, there exist Fréchet matching F between Q^* and P composed by the union of Fréchet matchings between $Q^*[s_j, s_{j+1}]$ and $P[b_j, b_{j+1}]$ for all $j \in [1, \ell)$, such that $\delta_F(Q^*[s_j, s_{j+1}], P[b_j, b_{j+1}]) \leq \varepsilon$. Let $\hat{u}$ and $\hat{v}$ be two points aligned to u and v under this matching F; obviously, $\delta_F((u, v), P(\hat{u}, \hat{v})) \leq \varepsilon$. See Fig. 4 for an illustration, where suppose $s_i^* = s_i \in S_{i-1}$.

We obtain S_i by removing s_i^* as a break-point from S_{i-1} and adding u and v as new break-points to S_{i-1}. (In general, the edge (u, v) may contain more break-points than s_i^*, and we need to remove all of them from S_{i-1}). Similarly, we add $\hat{u}$ and $\hat{v}$ as new break-points, and remove any existing break-points of Π_{i-1} contained in $P[\hat{u}, \hat{v}]$ (the break-point b_i that s_i is matched to will necessarily be removed). This gives rise to a pair of new partitions S_i and Π_i, where the number of pieces can increment by at most one.

We now argue that S_i and Π_i are also valid. We already know that $S_{i-1} = \{s_0, \ldots, s_\ell\}$ and $\Pi_{i-1} = \{b_0, \ldots, b_\ell\}$ are valid. Consider any two consecutive break-points in S_i. Then, one of the following three cases must hold:

(1) Both break-points $s_j, s_{j+1} \in S_{i-1} \cap S_i$. Then as S_{i-1} and Π_{i-1} are valid, we have that the two conditions (i) and (ii) in Definition 5.4 hold for s_j and s_{j+1}.

(2) We have a pair of new consecutive break-points u and v in S_i, corresponding to new break-points $\hat{u}$ and $\hat{v}$ in Π_{i-1}. However, we know that $Q[u, v] = (u, v)$ thus it is a shortest path between u and v; and also we know from above that $\delta_F((u, v), P(\hat{u}, \hat{v})) \leq \varepsilon$.

(3) The two consecutive break-points in S_i are either of the form s_j, u ($j = t - 1$ in Fig. 4); or symmetrically, they are $v, s_{j'}$ from S_i ($j' = t + 1$ in Fig. 4). Consider s_j and u from S_i, which correspond to consecutive break-points $b_j, \hat{u}$ from Π_i. By construction, $Q^*[s_j, u] \subseteq Q^*[s_j, s_{j+1}]$; thus $Q^*[s_j, u]$ is necessarily a shortest path of G as well. Furthermore, $P[b_j, \hat{u}] \subseteq P[b_j, b_{j+1}]$ and u is aligned to $\hat{u}$ under the Fréchet matching F between $Q^*[s_j, s_{j+1}]$ and $P[b_j, b_{j+1}]$ mentioned above. Thus, we have $\delta_F(Q^*[s_j, u], P[b_j, \hat{u}]) \leq \varepsilon$ under the same matching F. Symmetrically, we can argue that $\delta_F(Q^*[v, s_{j'}], P[\hat{v}, b_{j'}]) \leq \varepsilon$.

It then follows that S_i and Π_i are valid. Furthermore, after each iteration, if break-point s_i^* is in the interior of an edge in G, then we remove at least this break-point and add two new break-points u and v, which are vertices of graph G. Thus, in each iteration, the number of break-points can increase by at most one; implying that $|S_k| \leq 2k + 1$. After processing all s_i^*'s, all break-points from Q^* that are in the interior of graph edges are removed, and all newly added break-points are

graph nodes. Hence, the new partitions S_k of Q^* and Π_k of P witness a G-restricted $(2k, \varepsilon)$-matching for P in G, which proves Proposition 5.2. □

Proof of Proposition 5.3. We now describe a $(1, 2)$-approximation algorithm for the G-restricted case, which would then prove Proposition 5.3. To make the main idea clear, we first assume that in the G-restricted (k, ε)-matching, *all break-points* of the k-SP Q have to be vertices in G (in our earlier definition, the start point and endpoint may not be). At the end of this proof, we will describe how to remove this assumption.

The high-level framework is similar to the approach in Section 4.2. Given parameter ε, we build an auxiliary graph $G' = (V', E')$ as follows. The node set V' consists of $\{\langle v, e_i\rangle \mid v \in V, e_i = (p_i, p_{i+1})$ is the i-th edge in $P\}$. The edge $(\langle v, e_i\rangle, \langle v', e_j\rangle) \in E'$ is in the auxiliary graph G' if and only if there is a subpath $P[a, b] \subseteq P$ with $a \in e_i$ and $b \in e_j$, as well as a shortest path Q from $v \in V$ to $v' \in V$ in G, such that $\delta_F(Q, P[a, b]) \leq \varepsilon$; if $i = 0$, then we require $a = p_0$, and if $j = N - 1$, then we require $b = p_N$. The latter two conditions are to guarantee that the first and last break-points for the partition of P have to be p_0 and p_N, respectively. Our algorithm returns 'yes' if a path of at most k links exists from $\langle s, e_0\rangle$ to $\langle t, e_N\rangle$ in the auxiliary graph G' for some $s, t \in V$.

We compute the edge set E' for this auxiliary graph as follows. We add the edge $(\langle v, e_i\rangle, \langle v', e_j\rangle)$ to the auxiliary graph G' as long as there is a monotone path in the free space D_v starting from some point, say (v, a), along the vertical edge $v \times e_i$, to some point, say (v, b), within the vertical edge $v' \times e_j$. We further associate the pair (a, b) with the edge $(\langle v, e_i\rangle, \langle v', e_j\rangle) \in E'$, and say that (a, b), with $a \in e_i$ and $b \in e_j$, *witnesses the existence of the edge* $(\langle v, e_i\rangle, \langle v', e_j\rangle)$. Using Corollary 4.3 for the shortest path DAG T_v for $v \in V$, this can be computed in $O(|T_v|N) = O(mN)$ time for fixed v and i.

Overall, this auxiliary graph has $|V'| = O(nN)$ nodes, and $|E'| = O(n^2N^2)$ edges. Constructing all edges takes $O(|V'|mN) = O(nmN^2)$ total time. We need to test whether there is a path from $\langle s, e_0\rangle$ to $\langle t, e_N\rangle$ of at most k links in G' for some $s, t \in V$. This can be done in $O(|V'| + |E'|) = O(nmN^2)$ time by adding a super-source node as in the algorithm for Theorem 4.5. Hence, the claimed time complexity in Proposition 5.3 follows.

To prove the correctness of our algorithm, we will show how to construct a G-restricted $(k, 2\varepsilon)$-matching from the k-link path mentioned above from the auxiliary graph. Let

$$\begin{aligned} u_0 = \langle s = s_0, p_0\rangle, u_1 = \langle s_1, e_{i_1}\rangle, \ldots, \\ u_{k-1} = \langle s_{k-1}, e_{i_{k-1}}\rangle, u_k = \langle s_k = t, p_N\rangle \end{aligned} \tag{2}$$

be the sequence of nodes for a path of k links from $u_0 = \langle s, p_0\rangle$ to $u_k = \langle t, p_{N-1}\rangle$ in the auxiliary graph G'. Obviously, this gives rise to a k-SP $Q = Q_1 \circ Q_2 \circ \cdots Q_k$, where for each $j \in [0, k-1]$, Q_j is a shortest path from s_j to s_{j+1} in G. We now construct a k-partition of P as follows. Consider the edge (u_j, u_{j+1}) from the path in Equation (2). Let $a_j \in e_{i_j}$ and $b_j \in e_{i_{j+1}}$ be the two points witnessing the existence of edge (u_j, u_{j+1}) in E' for any $j \in [1, k-1]$. Note that both b_{j-1} and a_j are from the same edge e_{i_j} of P; and we set $c_j = \frac{b_{j-1}+a_j}{2}$ to be the mid-point of $b_{j-1}a_j$. Obviously, c_j is also contained in e_{i_j}. Now simply consider the sequence of break-points $\{p_0, c_1, c_2, \ldots, c_{k-1}, p_N\}$ and the corresponding k-partition of P. We next prove that $\delta_F(Q_j, P[c_j, c_{j+1}]) \leq 2\varepsilon$ for each $j \in [1, k-2]$.

Indeed, by construction, we know that:

$$\delta_F(Q_{j-1}, P[a_{j-1}, b_{j-1}]) \leq \varepsilon$$

$$\delta_F(Q_j, P[a_j, b_j]) \leq \varepsilon$$

and $a_j, b_{j-1} \in e = e_{i_j}$. See Fig. 5 for an illustration. Since the start point of Q_j and the endpoint of Q_{j-1} are the same, which is s_j, we then have that $\|a_j - s_j\| \leq \varepsilon$ and $\|b_{j-1} - s_{j+1}\| \leq \varepsilon$. By the triangle inequality, $\|a_j - b_{j-1}\| \leq 2\varepsilon$. Since c_j is the mid-point of $b_{j-1}a_j$, we have that $\|a_j - c_j\|, \|c_j - b_{j-1}\| \leq \varepsilon$.

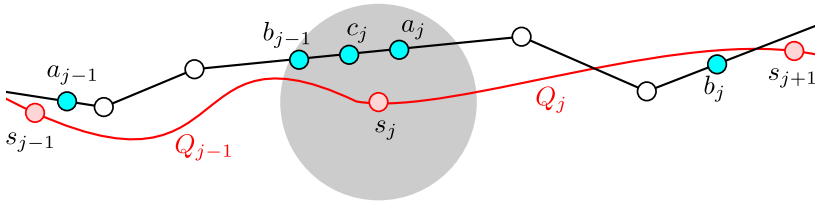


Fig. 5. Since c_j is within the segment $b_{j-1}a_j$, $||s_j - c_j|| \leq \varepsilon$ by convexity of the ε-ball around s_j (shaded region).

Now, consider the Fréchet matching $\mathcal{F}$ between Q_j and $P[a_j, b_j]$: If $c_j \leq a_j$ (the case illustrated in Fig. 5), then we can extend this matching to a matching between Q_j to $P[c_j, a_j]$ by simply matching all points in $P[c_j, a_j]$ to the point $s_j \in Q_j$. By convexity of the distance function, we have that $\|p - s_j\| \leq \varepsilon$ for any point p in the segment $P[c_j, a_j]$. Hence, we obtain a Fréchet matching between Q_j and $P[c_j, b_j]$ with error still at most ε.

Otherwise if $c_j > a_j$, then let q be the first point in Q_j that is matched to c_j under the Fréchet matching between Q_j and $P[a_j, b_j]$. Consider the subcurve $Q_j[s_j, q]$. We construct a Fréchet matching $\mathcal{F}'$ between Q_j and $P[c_j, b_j]$ by keeping the matching for all points in $Q_j[q, s_{j+1}]$ the same as in $\mathcal{F}$, but re-matching all points in $Q_j[s_j, q]$ to c_j. Note that each $q' \in Q_j[s_j, q]$ is matched to some point $p \in P[a_j, c_j]$ under $\mathcal{F}$. Hence, $\|q' - c_j\| \leq \|q' - p\| + \|p - c_j\| \leq 2\varepsilon$. Thus, the new matching $\mathcal{F}'$ has error at most 2ε.

By a symmetric argument, we can further modify the Fréchet matching $\mathcal{F}'$ between Q_j and $P[c_j, b_j]$ to a new matching $\mathcal{F}''$ between Q_j and $P[c_j, c_{j+1}]$ with error at most 2ε. After performing this modification for all j, we have $\delta_F(Q_j, P[c_j, c_{j+1}]) \leq 2\varepsilon$ for all $j \in [1, k-2]$. Finally, we have that $\delta_F(Q_1, P[p_0, c_1]) \leq 2\varepsilon$ and $\delta_F(Q_k, P[c_{k-1}, p_N]) \leq 2\varepsilon$ by a similar argument. The proposition then follows.

We now describe how to remove the assumption that all break-points in the G-restricted (k, ε)-matching have to be graph vertices. So, we must explain how to allow the start point and endpoint of Q to lie in the interior of graph edges.

In the argument above, we assumed that all break-points of the k-SP path Q have to be graph vertices. In our definition of G-restricted (k, ε)-matching, the first and last break-points of Q could be points from the interior of graph edges. To allow this, we modify the auxiliary graph $G' = (V', E')$ to also add nodes of the form $\langle e, e_0 \rangle$ or $\langle e, e_{N-1} \rangle$ to V', where $e \in E$ is any edge in input graph G, and e_0 (resp., e_{N-1}) is the first (resp., the last) edge of the input path P. In order to check whether an edge $(\langle e, e_0 \rangle, \langle v, e_i \rangle)$ is in the edge set E' of the auxiliary graph G', we perform the following.

Let $e = (s, t)$ with $s, t \in V$. For any point $x \in |e|$, the shortest path from v to x either passes through vertex s, or through vertex t. In fact, there exists a point $w \in |e|$ such that for any point in segment sw, the shortest path from v to it passes through s; while for any point in wt, the shortest path from v to it passes through t. We break e into two segments $e_s = sw$ and e_t. First consider e_s. For any $x \in |e_s|$, a shortest path from v to x in G is the concatenation of a shortest path from v to s and segment sx. We thus construct a DAG T_v containing all shortest paths from v to s, plus the edge e_s. We then build the free space diagram $D_v = T_v \times \overline{P}[p_{i+1}, p_0]$, where $\overline{P}$ respresents the path P with the reverse orientation. We add an edge $(\langle e, e_0 \rangle, \langle v, e_i \rangle)$ to the auxiliary graph if there is a shortest path $\pi_{v \rightsquigarrow x}$ in G from v to some point x in e_s and a point $b \in e_i = p_i p_{i+1}$ such that $\delta_F(\pi_{v \rightsquigarrow x}, \overline{P}[b, p_0] \leq \varepsilon$. This decision problem can be answered by checking whether any point on the grid edge $e_s \times p_0 \in D_v$ is reachable by a monotone path from the free-region of D_v starting from any point in the grid edge $v \times e_j \in D_v$. Such reachability can be maintained by a similar dynamic programming procedure as used earlier in time $O(|D_v|N) = O(mN)$. Similarly, we also

check whether there is a shortest path from v to a point in subedge $e_t \subseteq e$ that is within Fréchet distance ε to the sub-path $\overline{P}[b, p_0]$ for some $b \in e_i = p_i p_{i+1}$, using the same approach. We add edge $(\langle e, e_0\rangle, \langle v, e_i\rangle)$ to the auxiliary graph if the answer is yes.

The total number of extra edges we need to check for is $O(mnN)$, and checking for the existence of each such edge takes $O(mN)$ time. Hence, we need $O(nm^2N^2)$ extra time to build the auxiliary graph.

This time complexity however can be improved to $O(nmN^2)$ by batching the testing for all edges of the form $(\langle e, e_0\rangle, \langle v, e_i\rangle)$ for a fixed $v \in V, e_i \in P$, but all $e \in E$. We sketch the argument here. Let D'_v be the shortest path DAG rooted at v to all other *graph nodes* in V. Next, for each $s \in V$, consider the set of edges E_s incident on s but not in D'_v. For each such edge $e = (s, t) \in E_s$, we compute the furthest point $w \in |e|$ from s such that the shortest path in G from v to w passes through graph node s. We then add the partial edge ws to the DAG D'_v (note that it is possible that $w = s$. We do this for all edges in E_s for all nodes $s \in V$. The resulting modified DAG is denoted by D_v. We call edges in $D_v \setminus D'_v$ *partial edges*. We then run the same dynamic programming procedure to compute the reachability on all grid edges $e' \times p_0$ for all partial edges e' in $O(|D_v|N) = O(mN)$ as before. For any partial edge e' which is a subsegment of edge $(s', t') \in E$, if any point is reachable, then we add the edge $(\langle (s', t'), e_0\rangle, \langle v, e_i\rangle)$ to the auxiliary graph. Overall, we need to perform this construction $O(nN)$ times for all $v \in V$ and $e_i \in P$. Hence, the total construction time is $O(nmN^2)$. This completes the proof of Proposition 5.3. □

Recall that our main result, Theorem 5.1 then follows from Propositions 5.2 and 5.3.

Remark: We conjecture that we can develop an algorithm, with approximation factors better than (2, 2), for example, by constructing an approximate matching directly, instead of going through an intermediate G-restricted matching. We leave this as a direction for future research.

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.

ACKNOWLEDGMENTS

The authors would like to acknowledge the generous support of the National Science Foundation under grants IIS-1319944, CCF-1054779, CCF-1614562, DBI-1759807, CCF-1618605, CCF-1618247, and CCF-1618469.

REFERENCES

[1] Mahmuda Ahmed and Carola Wenk. 2012. Constructing Street Networks from GPS Trajectories. In *Proc. 20th Annual European Symposium on Algorithms*. 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 Conference on Very Large Data Bases (VLDB)*. 853–864.

[7] Erin W. Chambers, Brittany Terese Fasy, Yusu Wang, and Carola Wenk. 2018. Map-matching using shortest paths. In *Proc. 3rd International Workshop on Interactive and Spatial Computing*. 44–51.

[8] Ron Dalumpines and Darren M. Scott. 2017. Determinants of route choice behavior: A comparison of shop versus work trips using the Potential Path Area - Gateway (PPAG) algorithm and Path-Size Logit. *Journal of Transport Geography* 59 (2017), 59 – 68. https://doi.org/10.1016/j.jtrangeo.2017.01.003

[9] Maurice Fréchet. 1906. Sur quelques Points du Calcul Fonctionnel. *Rendiconti del Circolo Matematico di Palermo* 22, 1 (1906), 1–74.

[10] Amin Gheibi, Anil Maheshwari, and Jörg-Rüdiger Sack. 2016. Minimizing Walking Length in Map Matching. In *Proc. International Conference on Topics in Theoretical Computer Science*, M. T. Hajiaghayi and M. R. Mousavi (Eds.). 105–120.

[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] Antonio Lima, Rade Stanojevic, Dina Papagiannaki, Pablo Rodriguez, and Marta C. González. 2016. Understanding individual routing behaviour. *Journal of The Royal Society Interface* 13 (2016). Issue 116. https://doi.org/10.1098/rsif.2016.0021

[13] 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.

[14] E. J. Manley, J. D. Addison, and T. Cheng. 2015. Shortest path or anchor-based route choice: a large-scale empirical analysis of minicab routing in London. *Journal of Transport Geography* 43 (2015), 123 – 139. https://doi.org/10.1016/j.jtrangeo.2015.01.006

[15] 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.

[16] 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

[17] Piotr Szwed and Kamil Pekala. 2014. An Incremental Map-Matching Algorithm Based on Hidden Markov Model. In *Artificial Intelligence and Soft Computing*. 579–590.

[18] 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.

[19] 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.

[20] 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.

[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