Discrete Planar Map Matching

Bin Fu* Robert Schweller*† Tim Wylie*†

Abstract

Route reconstruction is an important application for Geographic Information Systems (GIS) that rely heavily upon GPS data and other location data from IoT devices. Many of these techniques rely on geometric methods involving the Fréchet distance to compare curve similarity. The goal of reconstruction, or map matching, is to find the most similar path within a given graph to a given input curve, which is often approximate location data. This process can be approximated by sampling the curves and using the discrete Fréchet distance. Due to power and coverage constraints, the GPS data itself may be sparse causing improper constraints along the edges during the reconstruction if only the continuous Fréchet distance is used. Here, we look at two variations of discrete map matching: one constraining the walk length and the other limiting the number of vertices visited in the graph. We give an efficient algorithm to solve the question based on walk length showing it is in **P**. We prove the other problem is **NP**-complete and the minimization variant is **APX**-hard while also giving a parameterized algorithm to solve the problem.

1 Introduction

There are many important applications related to GIS systems due to the proliferation of GPS enabled devices and the continued development of IoT devices. Route reconstruction is the process of finding the most likely path of an object based on the GPS data and the possible pathways. For instance, GPS data may indicate a car was driving through buildings, and we want to fit the data to the road network to recreate the most likely path of the car.

Route reconstruction depends greatly on what metric is used to determine how close the reconstructed path is. The two main methodologies are those based on geometric methods and Global Weight Optimization. However, the methodologies can also be classified based on the problem definition where we have local/incremental methods, global methods, and statistical methods. These can be extended to include topological and geological conditions, current weather and traffic conditions, speed limits, and other variables that can produce more optimal routes [17, 23]. Here, we focus on global geometric methods, and assume we have all of the data as input to find an optimal solution.

One popular means of measuring this fit is the Fréchet distance. Finding a path in a graph given a polygonal curve is also referred to as map matching. Map matching with respect to the Fréchet distance was first posed by Alt et. al. [6] as follows: Let $G = (V, E)$ be an undirected connected planar graph with a given straight-line embedding in $\mathbb{R}^2$ and a polygonal line P. Find a path Q in G which minimizes the Fréchet distance between P and Q. They give an efficient algorithm which runs in $\mathcal{O}(pq \log q)$ time and $\mathcal{O}(pq)$ space where p is the number of line segments of P and q is the complexity of G. This allows for vertices and edges to be traversed multiple times. Maheshwari et al. improved the running time for the map matching problem for complete graphs [19]. The original algorithm decides it in $\mathcal{O}(pn^2 \log n)$, where n is the number of vertices in the graph, and their new algorithm solves it in $\mathcal{O}(pn^2)$. We refer to this problem (in a complete graph) as the set-chain matching problem, which was studied in more detail in [1, 2, 3, 25].

There has been work that yields better performance with certain types of curves, with dual simplification for an approximate result, with bounded simplification of one of the chains, and in graphs with certain properties, [8, 11, 12, 14]. With map matching, for the weak Fréchet distance, the bounds have been lowered further to $\mathcal{O}(pq)$ [13], and the problems can be defined with a smaller error bound [24].

All this work has focused on the continuous Fréchet distance, which assumes that every point along the curve is meaningful. In reality, all GPS data is discrete, and these approaches smooth the data. There are some methods optimized for low-sampling-rate data [17], but even these assume some maximum time between samples (less than five minutes). Our goal is to analyze data where samples may be hours apart and can not be reasonably smoothed. There are many instances where you may not have GPS data, such as power constraints (low battery), or coverage issues (no towers), or required disconnects (airline travel). In these cases, you only connect to a cell tower or satellite intermittently, and thus the resulting polygonal curve is only meaningful at the nodes.

Another application where map matching algorithms are useful are discretizations of any continuous data.

*Department of Computer Science, University of Texas - Rio Grande Valley

†This author's research was supported in part by National Science Foundation Grant CCF-1817602.

Examples include cartography applications, schematic maps, or polygon simplification [7, 16, 21].

Our Results. We introduce two additional variants of discrete map matching. We show that minimizing the resulting path is polynomial. We then show that restricting the set of vertices used from the graph is **NP**-complete and the minimization variant is **APX**-hard. We give a positive result based on a separator that yields a polynomial time algorithm under more realistic input assumptions (similar to those in [7]).

2 Preliminaries

The discrete Fréchet distance was originally defined by Eiter and Mannila in 1994 [15], and was further expanded on theoretically by Mosig et al. in 2005 [22].

Given two polygonal curves, we define the discrete Fréchet distance as follows. We use $d(a, b)$ to represent the Euclidean distance between two points a and b, but it could be replaced with other distance measures depending on the application.

Definition 1 *The discrete Fréchet distance, d_F, between two polygonal curves $f : [0, m] \to \mathbb{R}^k$ and $g : [0, n] \to \mathbb{R}^k$ is defined as:*

$$d_F(f, g) = \min_{\substack{\sigma:[1:m+n]\to[0:m],\\ \beta:[1:m+n]\to[0:n]}} \max_{s\in[1:m+n]} \left\{ d\Big(f(\sigma(s)),\, g(\beta(s)) \Big) \right\}$$

where σ and β range over all discrete non-decreasing onto mappings of the form $\sigma : [1 : m + n] \to [0 : m], \beta : [1 : m + n] \to [0 : n]$.

The continuous Fréchet distance is typically explained as the relationship between a person and a dog connected by a leash walking along the two curves and trying to keep the leash as short as possible. However, for the discrete case, we only consider the nodes of these curves, and thus the man and dog must "hop" along the nodes. Figure 1 shows this relationship between the two and how with enough evenly sampled points on the two curves, the resulting discrete Fréchet distance can closely approximate the continuous Fréchet distances.

With a dynamic programming solution for finding the discrete Fréchet distance between two polygonal curves with m and n nodes, Eiter and Mannila proved that $\mathcal{O}(mn)$ was possible [15]. Recently, a slightly subquadratic algorithm was discovered by Agarwal et al. showing the discrete Fréchet distance can be computed in $\mathcal{O}(\frac{mn \log\log n}{\log n})$ time [4].

Bringmann and Mulzer [10] recently showed the there is no strongly subquadratic algorithm for the discrete Fréchet distance unless the strong exponential time hypothesis (SETH) fails [9].

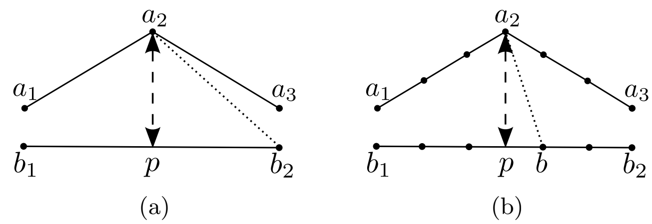


Figure 1: Figures (a) and (b) show the relationship between the discrete and continuous Fréchet distance where p is the point on the line closest to a_2 for the continuous and the dotted line represents the closest discrete distance from a_2 (using only nodes). (a) the curves have fewer nodes and a larger discrete Fréchet distance, while (b) has the same paths with more nodes, and thus provides a better approximation.

2.1 Discrete Map Matching

The definition of discrete map matching follows and we discuss two variants that we consider in this work.

Definition 2 (Discrete Map Matching)
Instance: *Given a simple connected planar graph $G = (V, E)$ embedded in $\mathbb{R}^2$, a polygonal curve P in $\mathbb{R}^d$ ($d \geq 2$), an integer $K \in \mathbb{Z}^+$, and an $\varepsilon > 0$.*
Problem: *Does there exist a walk Q in G with vertices chosen from V' where $V' \subseteq V$, such that $T \leq K$ and $d_F(P, Q) \leq \varepsilon$ where T is defined as either*

- *$T = |Q|$, where the size of the chain is being restricted, or*
- *$T = |V'|$, where the number of vertices in the graph is restricted (a vertex visited multiple times in the walk only counts as one).*

We look at the analogous variants of the set-chain matching problems (rather than a graph they find a path through a set of points) [1, 2, 3, 19, 25]: the Non-unique map matching problem constrains either the curve (NMMC-k) or the set of vertices used (NMMS-k). There is a variant where the vertices in the walk must also form a path, which is Unique map matching (UMM-k), and was shown to be **NP**-complete simultaneously in [20, 26] with extended stronger results in [16]. Note that when the vertices are unique the two decision problems ($|Q|$,$|V'|$) are equivalent. For reference, the naming convention is (U)nique/(N)on-unique (M)ap (M)atching with a k (S)ubset/(C)hain.

3 Non-unique Map Matching With Restricted Length (NMMC)

Here, we discuss discrete map matching concerned with the length of the path through the graph. As the NMMC problem restricts the length of Q, the problem is similar to the set-chain variant (NSMC) [25] and has a similar optimal substructure. The recurrence to find the minimum size of Q (in number of vertices), is given in Equation 1. The recurrence uses a 2D table M of size $|V| \times |P|$ where the first column is initialized to one if $d(v_k, p_1) \leq \varepsilon$ where $1 \leq k \leq |V|$, and the values are set to ∞ otherwise. $N(v)$ stands for the neighborhood of vertex v, which is the set of adjacent vertices in G.

The recurrence minimizes the number of vertices used while going from p_1 to $p_{|P|}$. This is done by ensuring that for each p_i, $1 \leq i \leq |P|$, we mark all vertices v with $d(v, p_i) \leq \varepsilon$ and that v is adjacent to at least one vertex used in the walk so far, i.e., there is a v' where $d(v', p_{i-1}) \leq \varepsilon$ and there is an edge between v and v'.

$$M[i,j] = \min \begin{cases} M[i,j\text{-}1], & \text{if } d(v_i, p_j) \leq \varepsilon, M[i,j\text{-}1] \neq \infty \\ \min\limits_{v_k \in N(v_i)} M[k,j\text{-}1]+1, & \text{if } d(v_i, p_j) \leq \varepsilon \\ \infty, & \text{if } d(v_i, p_j) > \varepsilon \end{cases} \quad (1)$$

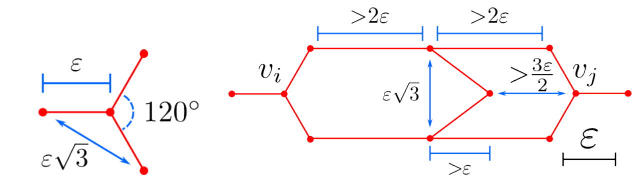
This algorithm works for any graph with the worst case being a complete graph, which is equivalent to the discrete set-chain matching variant [25] and has complexity $\mathcal{O}(|P|(|V| + |E|))$. Each vertex v only looks at its neighbor set, $N(v)$, and since a planar graph has fewer edges, the algorithm has a faster runtime. A planar graph has $|E| = 3(|V| - 2) = \mathcal{O}(|V|)$, yielding a run time of $\mathcal{O}(|P||V|)$ for planar graphs.

Theorem 1 *The discrete Non-unique Map Matching (NMMC) problem restricting the number of nodes in the output polygonal curve Q (vertices in the walk) can be solved in $\mathcal{O}(|P|(|V| + |E|))$ time for general graphs, and $\mathcal{O}(|P||V|)$ time for planar graphs.*

The optimal walk can be extracted by a simple backtracking algorithm. Find the minimum value in the last column, and the index of that row is the last vertex of the walk. Then, continually look at the previous column and find either the same row (same value), or look at all neighbors of that vertex and find a row with a value that is one less than the current value.

4 Discrete Non-unique Map Matching with Restricted Set (NMMS)

Discrete map matching concerned with restricting the number of vertices of the graph that the walk uses is an interesting problem related to coverage. Imagine a route reconstruction problem looking at cellphone tower coverage. If we wanted to know whether it was possible that the driver connected to fewer than k towers, this is equivalent to NMMS. On a complete graph, this is the same as discrete unit disk cover [25], and thus NMMS is asking a DUDC question related to planar connectivity



(a) Vertex Gadget (b) Connecting Edge

Figure 2: (a) The vertex gadget replaces a vertex with 4 vertices and three connecting locations. (b) Edge Connecting two vertex gadgets. Note that there is no restriction requiring the edges to be straight, just that the distance bounds are maintained.

between the disks. Section 4.1 shows NMMS is **NP**-complete, and Section 4.2 shows the minimization variant is **APX**-hard. We give a polynomial result related to real-world application constraints in Section 4.3.

4.1 Reduction Overview

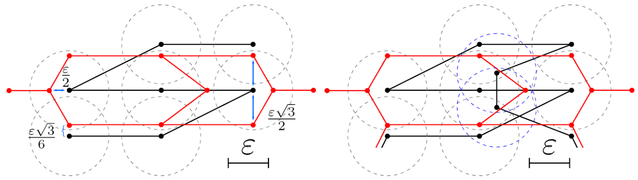
NMMS on a complete graph is equivalent to Non-Unique Set Matching with a fixed set for some k (NSMS), which is **NP**-complete [25]. On a planar graph the problem is different since our walk through the graph is limited by its neighbors and the planarity of the graph. We show that the problem is **NP**-complete via a reduction from Planar Vertex Cover with max degree three (PVC3), which was shown to be **NP**-complete in [18] and shown to be **APX**-hard in [5]. For Planar Vertex Cover we are given a planar graph $G = (V, E)$ and an integer K_{vc} as input. For this special case we know that $deg(v) \leq 3 \; \forall \; v \in V$. We want to know if there is a vertex cover of G of size at most K_{vc}.

We use several gadgets to transform an instance of PVC3 into an instance of NMMS. Since we are moving from a graph to a geometry problem, we use a planar embedding of the graph. Let G_s be a planar embedding of the graph G where each edge has length greater than 5ε. This ensures our geometric gadgets work correctly for any given $\varepsilon > 0$.

The reduction is going to make a new graph and a polygonal curve that will visit each edge of G exactly twice by creating a doubly-connected edge list (Section 4.1.2). The gadgets that replace each vertex and edge of G (Section 4.1.1) ensure that this walk is possible.

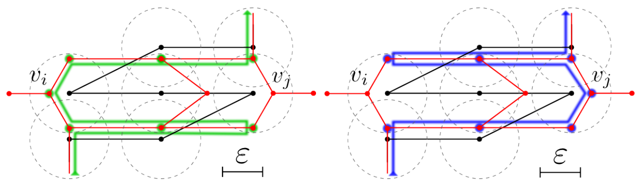
4.1.1 Gadgets

Each vertex in G_s is replaced with the vertex gadget shown in Figure 2a, which consists of four vertices and three edges connecting them. The edge lengths are exactly ε in length. The central vertex represents the original vertex while the other three will be used to connect edges. We replace each edge by three additional vertices and six edges that connect two vertex gadgets (Figure



(a) Edge Gadget (b) Edge-Cross Gadget

Figure 3: (a) An edge gadget includes the two connected vertex gadgets and the connecting edges between them, and then also has a piece of the polygonal curve. The circles represent an ϵ-ball around each node of the curve to show which vertices in the graph are within ϵ. It begins on one "side" of the edge and ends on the other "side." (b) The edge-cross gadget is an edge gadget that has the polygonal curve ending on the same side of the edge gadget that it began.



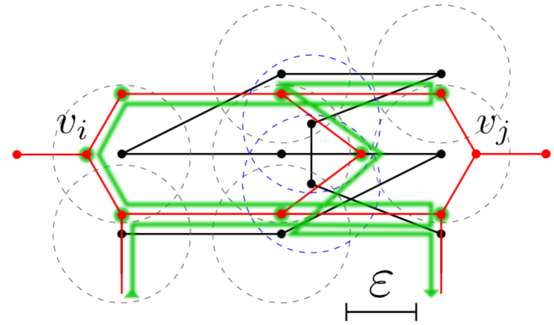
(a) Walk through v_i (b) Walk through v_j

Figure 4: The two possible walks through an edge gadget connecting vertices v_i and v_j. (a) A walk through the edge gadget that uses the center vertex of v_i. (b) A walk through the edge gadget that uses the center vertex of v_j.

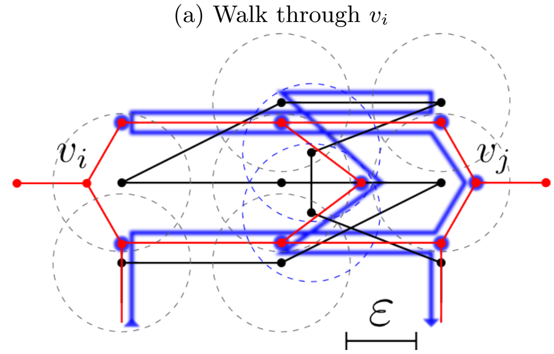
2b). Note that the new edges can be arbitrarily placed in the plane as necessary as long as the relationship between the three center vertices is maintained. Since we know the max degree of any vertex in G_s is three, we can connect up to three edge gadgets to each vertex gadget. We now have a new graph $G'_s = (V', E')$ with exactly $|V'| = 4|V| + 3|E|$, and $|E'| = 3|V| + 6|E|$.

For the full edge gadget, we add a polygonal curve to each edge, as shown in Figure 3a. We discuss connecting these into one curve later; for now we focus on a single edge. If we walk through the graph following the polygonal curve, and minimize the number of vertices used, then there are only two possible walks. Assume we begin at the lower left edge vertex (on the variable gadget). We must either follow the walk shown in Figure 4a or the one in 4b. Both walks use only 7 vertices, and this is minimal for any edge gadget. This means we must use the center vertex representing v_i or v_j for every edge e_{ij}. We could use both, but we only have to use one, and either will work. In this way, we have a vertex "covering" that edge. The same vertex could be used for all three edge gadgets.

The edge gadget ends the walk on the opposite side of the edge from where it started, which can be a problem (explained later). Thus, we finish the edge gadget with a



(a) Walk through v_i



(b) Walk through v_j

Figure 5: The two possible walks through an edge-cross gadget connecting vertices v_i and v_j. (a) A walk through the edge-cross gadget that uses the center vertex of v_i. (b) A walk through the edge-cross gadget that uses the center vertex of v_j.

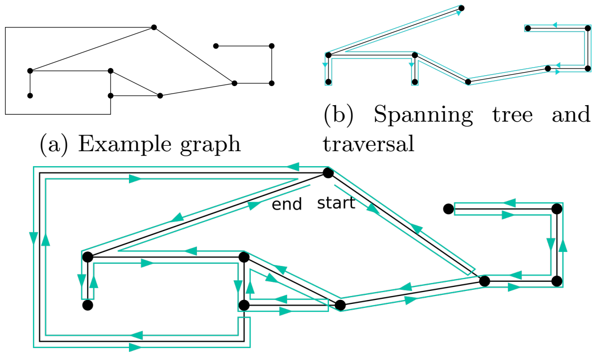
crossover that allows the walk to cross back to the other side of the edge. Figure 3b shows this construction. Figures 5a and 5b show the two possible walks through the edge. Thus, every edge gadget needs a minimum of 8 vertices for a walk following the curve.

4.1.2 Connecting Gadgets

To connect all of the small polygonal curves into a single curve, we need a walk that traverses every edge in our original graph at least once, and it may traverse an edge multiple times. The edge-traversal algorithm (Algorithm 1) ensures we go through every edge exactly twice (an example is shown in Figure 6). This generates a doubly connected edge-list (DCEL).

Algorithm 1 (Generate DCEL)
Input: Graph $G = (V, E)$
Output: Sequence of edges

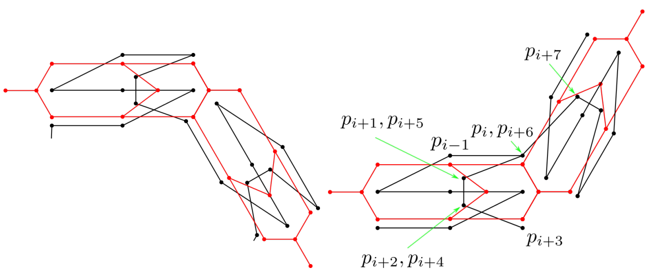
- *Compute a minimum spanning tree M of G and pick a vertex v.*
- *Create a path P around M by visiting each edge twice with the path in a counter-clockwise manner.*
- *At each vertex $v_i \in P$, add any edge $e_{\{v_i,v_j\}}$ twice to the edge sequence if it is not part of P.*



(a) Example graph (b) Spanning tree and traversal

(c) Final edge traversal

Figure 6: (a) An example planar graph with max vertex degree three. (b) A random MST with a traversal going through all vertices and going through all edges of the MST twice. (c) Adding in all other edges by following the path and inserting the missing edges into the path at each vertex.



(a) Right Turn (b) Left Turn

Figure 7: Assuming we are coming from the left-most vertex gadget. (a) Two edge-cross gadgets connected by a vertex gadget turning right. (b) Two edge-cross gadgets connect its left and thus making a left turn. After we cross the first edge, we must cross back to the other side in order to turn left, so those three nodes in the curve are repeated in reverse order: $\langle p_1, \ldots, p_i, p_{i+1}, \ldots, p_{i+6}, \ldots p_F \rangle$ s.t. the locations of p_i and p_{i+6}, p_{i+1} and p_{i+5}, and p_{i+2} and p_{i+4} are the same.

With this walk, connect the polygonal curve segments and adjust each segment whenever we cross the same edge twice (the number of times we will visit the same edge) or we need to cross to the other side of an edge (to finish on the opposite side). With the crossovers, Figure 7 shows how to make right and left turns.

4.1.3 Complexity

The new graph has $|V'| = 4|V| + 3|E|$, and the number of edges $|E'| = 3|V| + 6|E|$ with an optimal walk using $K = 7|E| - |d_2| - 3|d_3| + K_{vc}$ where $d_i = \{v_j | v_j \in V, deg(v_j) = i\}$, i.e., d_i is the set of all vertices from V with degree i. Since adjacent edges on the vertex gadget share vertices, we must subtract these for vertices of degree 2 and 3 in the original graph.

Theorem 2 *Discrete non-unique map matching with $T = |V'|$ (NMMS) is **NP**-complete.*

Proof. The graph G has a vertex cover of size K_{vc} if and only if there exists a walk Q in G'_s with P such that Q only passes through $K = 7|E| - |d_2| - 3|d_3| + K_{vc}$ vertices and $d_F(P, Q) \leq \varepsilon$.

Given the graph G, the given construction allows a walk to pass through an edge gadget with a minumum of 7 vertices. Since six of these must be used for every edge giving $7|E| - |d_2| - 3|d_3|$ vertices, the crossing vertex comes from one of the two vertex gadgets, but either can be used. Thus, every edge must have one of the two center vertex gadget vertices, which can be used for the adjacent edges as well. Thus, it is a vertex cover equivalent meaning K_{vc} vertices are sufficient.

Given an instance of G'_s and P. A walk must pass through $7|E| - |d_2| - 3|d_3|$ vertices. The additional vertex needed for each edge-cross gadget constitutes a single vertex associated with an adjacent edge in G. Thus, it would be a vertex cover.

For membership in NP, if given a set V' of vertices, take the induced subgraph of G of the vertices from V' and let it be G'. The problem then becomes equivalent to NMMC over G' with $k = |P|$, which is polynomial. Finally, we check the discrete Fréchet distance in polynomial time. □

4.2 APX-hardness

Here we show that the minimization variant of the problem is **APX**-hard with an L-reduction from PVC3. An L-reduction is an approximation-preserving reduction when both problems are minimization problems.

Definition 3 (L-reduction) *Let A and B be optimization problems and c_A and c_B their respective cost functions. A pair of functions f and g is an L-reduction if all of the following conditions are met:*

- *f and g are computable in polynomial time,*
- *if x is an instance of problem A, then $f(x)$ is an instance of problem B,*
- *if y' is a solution to $f(x)$, then $g(y')$ is to x,*
- *there exists a positive constant α such that* $\mathrm{OPT_B}(f(x)) \leq \alpha \mathrm{OPT_A}(x)$,
- *there exists a positive constant β such that for every solution y' to $f(x)$,* $|\mathrm{OPT_A}(x) - c_A(g(y'))| \leq \beta |\mathrm{OPT_B}(f(x)) - c_B(y')|$.

Theorem 3 *Minumum discrete non-unique map matching with $T = |V'|$ (Min NMMS) is **APX**-hard.*

Proof. We show this via an L-reduction. Let f be the L-reduction from PVC3 to NMMS using the described construction above. For every vertex cover V_c of size

K_{vc} of graph $G = (V, E)$ for PVC3, there is a vertex walk Q in our new graph $G'_s = (V', E')$ using $K' = 7|E| - |d_2| - 3|d_3| + K_{vc}$ vertices such that $d_F(P, Q) \leq \varepsilon$.

We construct our walk Q as a sequence of vertices from G'_s. Let $\{Q\}$ denote the vertices in the walk, i.e., $|\{Q\}| = K'$. Then for every vertex cover $V_c \subset V$ of G we construct the walk Q with $\{Q\} \subset V'$ of $G'_s = f(G)$ of size K'. Since G has bounded degree 3, it is clear that $3K_{vc} \geq \sum_{v \in V_{vc}} deg(v) \geq |E| \geq |V|$. We can see that $K' = 7|E| - |d_2| - 3|d_3| + K_{vc} \leq 7|E| + K_{vc} \leq 22K_{vc}$. We can replace $|E|$ with $3K_{vc}$. Thus, the first property of an L-reduction is satisfied with $\alpha = 22$. $K' \leq 22K_{vc}$.

Conversely, given a walk Q with $\{Q\} \subset V'$ of $G'_s = f(G)$ of size K', we transform it back into a vertex cover $V_{vc} \subset V$ of size K_{vc} of graph G as follows. We look at each variable gadget (any subgraph with one vertex with exactly 3 adjacent vertices of distance ε in the embedding as shown in Figure 2a). For every one of these central vertices, if the vertex is included in the walk Q, then we include it in V_{vc}. Observe this will give a vertex cover for G since the walk must include either v_i or v_j from the two variable gadgets on either side of an edge gadget. Note that $K_{vc} \leq K' - 7|E| + |d_2| + 3|d_3|$. Given any walk in the graph requires all the vertices except those that would be part of a vertex cover, we get that f is an L-reduction with $\beta = 1$. □

4.3 Positive Result Based on a Separator

Here, we develop an FPT algorithm for the NMMS problem based on a divide and conquer approach with a graph separator, which is a set of vertices that, if removed, disconnect the graph. Let $\text{ball}_d(p, \epsilon)$ be the ball in $\mathbb{R}^d$ with center $p \in \mathbb{R}^d$ and radius ϵ.[1]

Definition 4 ((ϵ, c)-local property) *A polygonal curve $P = \langle p_1, p_2, \ldots, p_n \rangle$ on a plane satisfies the (ϵ, c)-local property if for every p_i, the circle with center at p_i and radius ϵ does not contain any p_j with $|i - j| > c$.*

Definition 5 ((ϵ, c, u)-local property) *An input polygonal curve $P = \langle p_1, p_2, \ldots, p_n \rangle$ and planar graph $G = (V, E)$ satisfy the (ϵ, c, u)-local property if*

1. *The curve P satisfies the (ϵ, c)-local property, and*
2. *For every point $p \in \mathbb{R}^d$, the ball with center at p and radius ϵ contains at most u points in V.*

Theorem 4 *Assume that c and u are integer parameters. There is a $\mathcal{O}(n^{1+u \log c})$ time algorithm for the Discrete Map Matching with restricted set problem (NMMS) when the input $P = \langle p_1, p_2, \ldots, p_n \rangle$ and $G = (V, E)$ satisfy the (ϵ, c, u)-local property.*

[1]Given we are looking at planar embeddings, if we restrict the polygonal curve to be in $\mathbb{R}^2$, it is equivalent to look at the circle with radius ϵ centered at $p \in \mathbb{R}^2$.

Proof. Consider the case that the input is a polygonal curve satisfying the (ϵ, c)-local property, and an arbitrary graph. Let $S = \{p_{\lfloor \frac{n}{2} \rfloor + i} : i \in [0, c-1]\}$.

By brute force, iterate over all possible matchings between the c points in S, and at most u possible points in $V \cap \text{ball}(p_i, \epsilon)$. The number of possible matchings between the points in S and the points in V is at most c^u. For each such matching, we recurse on the two separated portions of P induced by the respective separator. This yields a divide and conquer algorithm that has time complexity $T(n) = 2c^u T(\frac{n}{2})$. Solving this recurrence equations yields a time complexity of $\mathcal{O}(n(c^{u \log n})) = \mathcal{O}(n^{1+u \log c})$. □

Corollary 1 *Assume that c and ϵ are fixed. Then there is a polynomial time algorithm if input P is a polygonal curve satisfying the (ϵ, c)-local property, and input G is a grid graph on the plane.*

Proof. Suppose the input graph is a grid graph and ϵ is fixed. Each vertex v_i in the curve P can only select at most $u = \pi(\epsilon + \sqrt{2}/2)^2$ grid points with distance ϵ to match. The result follows from Theorem 4. □

5 Conclusion and Future Work

In this work we introduce variants of discrete map matching and show that given different constraints the problem is tractable or **APX**-hard. When the length of the walk is restricted, we show the problem is decidable in $\mathcal{O}(|P||V|)$ time. For the variant restricting the number of vertices usable in the graph, after proving it is **APX**-hard, we give an FPT algorithm based on the structure of P and its relationship to the graph. Given the application of route reconstruction, this is a reasonable practical constraint. Our work leads to many open questions such as FPT algorithms based on different parameters of the input curve or graph beyond ours, and possible approximation algorithms. Is there a good constant factor approximation for minimum NMMS?

Another direction of research is to extend and generalize discrete map matching. Our problem definitions ignore all nodes in the graph outside the reach of the polygonal curve. To fully realize route reconstruction on large graphs, the problem should only ensure that at least one node is visited within the range of each node of the curve, but also accounts for and attempts to minimize paths through vertices that are not in range. This is similar to TSP with neighborhoods, except that the order of the neighborhoods is given. This is also similar to Group Steiner Tree and facility location problems.

Finally, what are the problem complexities under the continuous Fréchet distance, and are there better approximations or algorithms?

References

[1] P. Accisano and A. Üngör. Hardness results on curve/point set matching with fréchet distance. *CoRR*, abs/1211.2030, 2012.

[2] P. Accisano and A. Üngör. Approximate matching of curves to point sets. In *Proceedings of the 26th Canadian Conference on Computational Geometry*, CCCG'14, 2014.

[3] P. Accisano and A. Üngör. Finding a curve in a point set. *CoRR*, abs/1405.0762, 2014.

[4] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014.

[5] P. Alimonti and V. Kann. Some apx-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123 – 134, 2000.

[6] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, Nov 2003.

[7] Q. W. Bouts, I. I. Kostitsyna, M. van Kreveld, W. Meulemans, W. Sonke, and K. Verbeek. Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance. In *Proceedings of the 24th Annual European Symposium on Algorithms*, volume 57 of *ESA 2016*, pages 22:1–22:16, 2016.

[8] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. of the 31st Int. Conf. on Very Large Data Bases*, VLDB'05, pages 853–864. VLDB Endowment, 2005.

[9] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 661–670, 2014.

[10] K. Bringmann and W. Mulzer. Approximability of the discrete fréchet distance. *JoCG*, 7(2):46–76, 2016.

[11] K. Buchin, M. Buchin, and Y. Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. of the 20th Annual ACM-SIAM Sym. on Discrete Algorithms*, SODA'09, pages 645–654, 2009.

[12] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the Fréchet distance. In *Proc. of the 13th Workshop on Algorithm Engineering and Experiments*, ALENEX'11, pages 75–83. SIAM, 2011.

[13] D. Chen, L. J. Guibas, Q. Huang, and J. Sun. A faster algorithm for matching planar maps under the weak Fréchet distance. Unpublished, December 2008.

[14] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Comp. Geom.*, 48(1):94–127, 2012.

[15] T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Information Systems Dept., Technical University of Vienna, 1994.

[16] M. Löffler and W. Meulemans. Discretized approaches to schematization. In *Proceedings of the 29th Canadian Conference on Computational Geometry*, CCCG'17, pages 220–225, 2017.

[17] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proc. of the 17th ACM SIGSPATIAL Int. Conf. on Advances in G.I.S.*, GIS'09, pages 352–361, New York, NY, 2009. ACM.

[18] D. S. J. M. R. Garey. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[19] A. Maheshwari, J.-R. Sack, K. Shahbaz, and H. Zarrabi-Zadeh. Staying close to a curve. In *Proc. of the 23rd Annual Canadian Conf. on Computational Geometry*, CCCG'11, 2011. August 10-12, 2011.

[20] W. Meulemans. Map matching with simplicity constraints. *CoRR*, abs/1306.2827, 2013.

[21] W. Meulemans. Discretized approaches to schematization. *CoRR*, abs/1606.06488, 2016.

[22] A. Mosig and M. Clausen. Approximately matching polygonal curves with respect to the Fréchet distance. *Comp. Geom.: Theory and Appl.*, 30(2):113–127, Feb 2005.

[23] H. Wei, Y. Wang, G. Forman, and Y. Zhu. Map matching by Fréchet distance and global weight optimization. Technical Report SJTU_CS_TR_201302001, Department of Computer Science, Shanghai Jiao Tong University, 2013.

[24] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *18th Int. Conf. on Scientific and Statistical Database Management*, SSDBM'06, pages 379–388, 2006.

[25] T. Wylie and B. Zhu. Following a curve with the discrete Fréchet distance. *Theoretical Computer Science*, 556:34–44, oct 2014.

[26] T. Wylie and B. Zhu. Intermittent map matching with the discrete Fréchet distance. *CoRR*, abs/1409.2456:1–21, 2014.