# On Fréchet Traveling Salesmen Problems

**Omrit Filtser**
Department of Mathematics and Computer Science, The Open University of Israel, Ra'anana, Israel

**Tzalik Maimon**
Department of Mathematics and Computer Science, The Open University of Israel, Ra'anana, Israel

**Michal Moiseev**
Department of Mathematics and Computer Science, The Open University of Israel, Ra'anana, Israel

## Abstract

The Fréchet distance is a well-studied distance measure between two curves. In this work, we demonstrate that the merit of Fréchet distance extends beyond evaluating similarity, and introduce a new setting in which it proves useful. Consider a situation where two agents are required to visit a given set of sites, while staying close to each other throughout their traversal. In this paper, we study problems where the goal is to construct two curves whose vertices are from a given set of points, under the constraint that the Fréchet distance between the curves is kept as small as possible. This problem can be viewed as a variant of the Traveling Salesman Problem (TSP), and thus may be of interest in routing, network planning and more. We present a near-linear algorithm for this problem under the discrete Fréchet distance, and explore several variants of the problem, including minimizing the lengths of the curves and balancing the number of sites assigned to each agent. Lastly, we prove that the problem is NP-hard under the continuous Fréchet Distance.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Fréchet distance, traveling salesman problem

**Digital Object Identifier** 10.4230/LIPIcs.SWAT.2026.18

**Related Version** *Full Version*: `https://arxiv.org/abs/2606.01147`

**Funding** This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 2135/24).

**Acknowledgements** We would like to thank an anonymous reviewer for helpful comments and references, and in particular for suggesting the expected linear time algorithm in Section 3.

## 1 Introduction

The traveling salesman problem (TSP) asks for the shortest route that visits a given set of points. It is a classic NP-hard problem, and have various approximation algorithms (including a PTAS in the Euclidean plane [4, 23]). In some generalizations of TSP, e.g. the Vehicle Routing Problem or Multiple-TSP (see, e.g. [5, 7, 12, 24]), the goal is to plan short routes for a group of agents to visit all the points together. In these variants there is no restriction on the relations between the routes.

Consider a situation where two agents are asked to visit a given set of sites, they can split but they still need to remain close to each other throughout their motion. This can be required for example in order to keep then in some range of communication, or so that one agent can quickly get to other in case of trouble. Therefore, in this paper, we introduce and study a new set of problems where the goal is to plan routes for two (or more) agents, that together visit a given set of sites, while keeping the agents close to each other throughout the traversal. More precisely, our goal is to construct two curves from a given set of points, such that the (continuous or discrete) Fréchet distance between the curves is as small as possible. We also consider variations of this Fréchet-TSP type problem where the goal is to balance the load on the agents, in different ways.


© Omrit Filtser, Tzalik Maimon, and Michal Moiseev;
licensed under Creative Commons License CC-BY 4.0

The Fréchet distance [18], introduced by *Maurice Fréchet*, is a popular measure of similarity between curves. It is often described by an analogy of a person and a dog connected by a leash, both walking forward along two separate curves, while the leash keeps them at a bounded distance. The Fréchet distance is then the shortest length of a leash that allows them to traverse their respective curves. The Fréchet distance takes into account both the position and ordering of points, distinguishing it from other metrics like the Hausdorff distance. This property makes the Fréchet distance useful across a wide range of applications.

The discrete Fréchet distance focuses solely on the points of the curves rather than the edges between them. This variant can be described analogously by replacing the man and dog by two frogs that are "hopping" along the vertices of the curves while attempting to maintain a small distance between them. This discrete approach allows simpler and (slightly) faster algorithms for computing the distance.

Alt and Godau [3] showed that the Fréchet distance between a curve $P$ of length $n$ and a curve $Q$ of length $m$, can be calculated in $O(nm \log(nm))$ time. Eiter and Mannila [16] showed that the discrete Fréchet distance can be computed in $O(mn)$ time. Only 20 years later, it has been shown that under the Strong Exponential Time Hypothesis (SETH), the Fréchet distance cannot be computed in strongly subquadratic time [9]. Very recently, Cheng, Huang, and Zhang [13], presented a strong subquadratic time algorithm that computes a $(7+\varepsilon)$-approximation for both the discrete and continuous distance.

A plethora of applications and variants of the Fréchet distance have been studied since then (see, e.g. [10, 14, 15, 21]). The variant most relevant to our work is the Fréchet distance between two point sets, recently studied by Buchin and Kilgus [11]. In this variant, the input objects are two sets of points rather than two curves, and the goal is to construct two curves (one for each set of points), to minimize the Fréchet distance between them. The main difference from our version is that we also need to find a partition of the point set into two sets. Buchin and Kilgus showed that under the discrete Fréchet distance, the problem is equivalent to computing the Hausdorff distance between the two sets, and thus can be solved in $O(nm \log(nm))$ time (where $n, m$ are the sizes of the two sets). The continuous version, on the other hand, was shown to be NP-complete. They also provide an exponential time algorithm running in $O(k^a((m+n-a)+a\log a))$ where $a$ denotes the number of points that can be matched only to edges ("floating" points in their terminology) and $k$ is the maximum number of edges that can cover such a point.

Another closely related variant is the Curve-Point-Set Matching Problem (CPSM), where given a polygonal curve $P$, a set of points $S$, and a maximum distance $\delta > 0$, the objective is to find another polygonal curve $Q$, whose set of vertices is either a subset of $S$ or contains all the points of $S$ (also, either with or without repetitions, referred to as the non-unique/unique variant, respectively), such that the continuous or discrete Fréchet distance between the new curve $Q$ and the original curve $P$ is smaller than $\delta$. Maheshwari, Arora, and Smid [22] addressed the continuous non-unique subset version of the problem, and presented an algorithm that runs in $O(nk^2)$ time. Wylie [26] studied CPSM under the discrete Fréchet distance, and proved that the unique version (where each point in $S$ can appear only once in $Q$) is NP-complete for both the subset and all-point versions. This is in contrast to the non-unique cases which were shown to be polynomial-time solvable (in $O(nk)$ time, where $n$ is the size of $S$ and $k$ the size of $P$). Accisano and Ungor [1] also showed NP-completeness under the continuous distance for the all-points variant, both in unique and non-unique settings.

**Overview.** To the best of our knowledge, the Fréchet-TSP problem described above has not been studied before. This gives rise to a rich family of TSP-style questions centered around the Fréchet distance, and we systematically outline several natural variants that capture different aspects of this setting. In Section 3, we present an algorithm for the discrete Fréchet-TSP problem that runs in $O(n \log n)$ time for a set of $n$ points in a constant dimension. We utilize a combination of the properties of Nearest-Neighbor-Graph and Minimum-Spanning Tree (MST) over a Unit-Disk Graph (UDG). By utilizing the properties of UDG, we ensure an upper bound on the distance between the partitions we output. By utilizing the properties of the MST, we obtain a linear time partitioning. The initialization of these constructs are obtained in $O(n \log n)$ time. In Section 4, we consider a variant of the problem aiming to minimize the total length of the curves, and present a constant-factor approximation algorithm that also runs in $O(n \log n)$ time. In Section 5, we focus on balancing the number of vertices among the partitioned components, and present different strategies to achieve almost optimal balance. Here, we utilize the properties of the UDG again. Specifically, the upper bound on the number of kissing number of the graph. This in turn limits the number of cases in the possible partition we wish to balance. We handle these case-by-case showing that we can get the balance as close as at most 1 from optimal. Finally, in Section 6, we show that the continuous variant of Fréchet-TSP is NP-hard. Out proof is an adaptation of the method used by Buchin and Kilgus [11].

## 2 Notations and problem definition

A polygonal curve $P$ in $\mathbb{R}^d$ is a continuous function $P : [1, n] \to \mathbb{R}^d$, such that for any integer $1 \leq i \leq n-1$ the restriction of $P$ to the interval $[i, i+1]$ is a straight line segment. The points $P[0], \ldots, P[n]$ are the vertices of $P$, and the segments $\overline{P[i]P[i+1]}$ are the edges of $P$.

Let $P : [0, n] \to \mathbb{R}^d$ and $Q : [0, m] \to \mathbb{R}^d$ be two polygonal curves of with number of vertices $n$ and $m$, respectively.

**The (continuous) Fréchet distance.** A reparameterization of a curve $P$ is a continuous, non-decreasing, surjective function $f : [0, 1] \to [1, n]$ such that $f(0) = 1$ and $f(1) = n$. The Fréchet distance between $P$ and $Q$ is defined as $d_F(P, Q) = \inf_{f,g} \max_{t \in [0,1]} \|P(f(t)) - Q(g(t)\|$, where $f$ is a reparameterization of $P$ and $g$ is a reparameterization of $Q$.

**The discrete Fréchet distance.** [1] A ***paired-walk*** along $P$ and $Q$ is a sequence of pairs $\pi = \{(P_i, Q_i)\}_{i=1}^{k}$, such that $P_1, ..., P_k$ and $Q_1, ..., Q_k$ partition $P$ and $Q$, respectively, into (disjoint) non-empty subsequences of their vertices, and for any $i$ it holds that either $|P_i| = 1$ or $|Q_i| = 1$. The ***cost*** of a paired walk $W$ along $P$ and $Q$ is

$$\text{cost}(\pi) = \max_i \max_{(p,q) \in P_i \times Q_i} \|p - q\|.$$

For any pair $(p, q) \in P_i \times Q_i$, we say that $p$ is ***matched*** to $q$ in $\pi$.

The ***discrete Fréchet distance*** between $P$ and $Q$ is $\mathrm{d}_{dF}(P, Q) = \min_{\pi \in \Pi} \text{cost}(\pi)$, where $\Pi$ is the set of all possible paired-walks along $P$ and $Q$.

[1] For the simplicity of presentation, we follow the definition given in [6], which is equivalent to the original definition given in [16].

**Partitioning a point set into curves.** Let $\delta$ be a Fréchet-based distance measure for two curves in $\mathbb{R}^d$ (i.e., either the continuous or discrete Fréchet distance). Given a set $S$ of points in $\mathbb{R}^d$, we say that two curves $P, Q$ ***partition*** $S$ if there is a partition of $S$ into two sets $A$ and $B$ such that the set of vertices of $P$ is exactly $A$ and the set of vertices of $Q$ is exactly $B$, and each point in $S$ is used exactly once (either in $P$ or in $Q$). The basic version of the problem that we wish to consider is the following.

▶ **Problem 1** (Fréchet-TSP). *Given a set $S$ of points in $\mathbb{R}^d$, find two curves $P, Q$ that partition $S$ and such that $\delta(P, Q)$ is minimized.*

Note that if we do not require each point in $S$ to be visited by exactly one of the agents, then the problem becomes trivial: we can pick any order on the points in $S$ and set $P = Q$, so the distance between the paths is 0. In the full version of the paper, we discuss a variant in which we allow a point to be used more than once by the same agent, and show that this variant is equivalent to Problem 1 under the discrete Fréchet distance.

Denote by $\varepsilon^*$ the distance between the curves in an optimal solution for the Fréchet-TSP problem. Clearly, there may be many different optimal solutions, all having distance $\varepsilon^*$. However, some solutions may be considered better than others, for example, if each curve covers roughly the same number of points from $S$ (balanced partition – see Figure 1), or if the min-max length (or sum of lengths) of the curves is very small in comparison to other solutions (see Figure 2). We therefore define below different variants of the problem in which the goal is to find a "good" solution among those that achieve the optimal distance $\varepsilon^*$.

**Figure 1** Left: a balanced partition, right: an imbalanced partition. Both have the same distance.

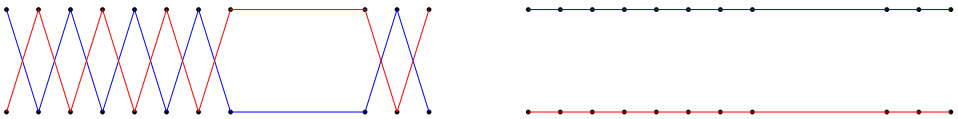

**Figure 2** Left: two long curves, right: two shorter curves. Both have the same distance.

▶ **Problem 2** (balanced-Fréchet-TSP). *Given a set $S$ of $n$ points in $\mathbb{R}^d$, find two curves $P, Q$ that partition $S$, such that $\delta(P, Q) = \varepsilon^*$, and $\max\{|P|, |Q|\}$ is minimized.*

For a curve $P$, denote by $\ell(P)$ the sum of the lengths of the edges of $P$.

▶ **Problem 3** (min-max-Fréchet-TSP). *Given a set $S$ of points in $\mathbb{R}^d$, find two curves $P, Q$ that partition $S$, such that $\delta(P, Q) = \varepsilon^*$, and $\max\{\ell(P), \ell(Q)\}$ is minimized.*

▶ **Problem 4** (min-sum-Fréchet-TSP). *Given a set $S$ of points in $\mathbb{R}^d$, find two curves $P, Q$ that partition $S$, such that $\delta(P, Q) = \varepsilon^*$, and $\ell(P) + \ell(Q)$ is minimized.*

Minimizing the length of the path is NP-hard, similar to the traveling salesman problem (TSP), which is NP-hard: for a reduction, simply double each point in an instance of TSP, leading to the following theorem:

▶ **Corollary 1.** *min-max-Fréchet-TSP and min-sum-Fréchet-TSP are NP-hard.*

We can thus aim to find an approximation algorithm for minimizing the length.

## 3 Discrete Fréchet-TSP

In this section, we focus on the discrete Fréchet distance (i.e. $\delta = \mathrm{d}_{dF}$). We begin by presenting an algorithm for the decision version of the problem: Given a set $S$ of $n$ points in $\mathbb{R}^d$, and a threshold $\varepsilon \geq 0$, decide whether there exist two curves $P$ and $Q$ that partition $S$ and have $\mathrm{d}_{dF}(P,Q) \leq \varepsilon$.

Let $G_\varepsilon = (S, E)$ be the graph whose vertices are the points in $S$, and there is an edge $\{v, u\} \in E$ if and only if $\|v - u\| \leq \varepsilon$. By definition, $G_\varepsilon$ is a unit-disk graph with radius $\varepsilon$. Let $P, Q$ be two curves that partition $S$ s.t. the edges in $P \cup Q$ are edges from $G_\varepsilon$. Any paired walk $\pi$ along $P$ and $Q$ with $\mathrm{cost}(\pi) \leq \varepsilon$ can be reduced to a set of disjoint stars in $G_\varepsilon$ simply by removing edges from $\pi$. Therefore, if there is such a walk $\pi$, then there is such a set of disjoint starts and vice versa. Thus, we can search for such a set.

▶ **Lemma 2.** *There exist two curves $P, Q$ that partition $S$ and have $d_{dF}(P,Q) \leq \varepsilon$ if and only if $G_\varepsilon$ does not contain a vertex of degree 0.*

**Proof.** If $G_\varepsilon$ contains a vertex $v$ of degree 0, then there is no $u \in S$ with $\|v - u\| \leq \varepsilon$. Therefore, $v$ cannot be matched to any other point in $S$ in a paired-walk of cost at most $\varepsilon$. For the other direction, let $C$ be a connected component of $G_\varepsilon$. We now show that if $C$ contains more than one vertex, then we can construct two curves on the set of vertices of $C$ as required. This finishes the proof, as we can concatenate the curves that were constructed for all the connected components, and get two curves with vertices from $S$ such that each point in $S$ is used exactly once. Let $T_C$ be some spanning tree of $C$. We color the vertices of $T_C$ red and blue, as follows: first, color all the leaves in red, and then, color the parents of all those leaves in blue. This coloring defines a set of stars, each has a blue center node and at least one red node. By removing these stars from $T_C = T_0$, we are left with a smaller tree, $T_1$. If $T_1$ is a single vertex, then color it red and add it to one of the stars of its child nodes. Otherwise, $T_1$ contains at least one edge. We then repeat the process on $T_1$ and obtain another set of stars, remove them from $T_1$ and get a smaller tree $T_2$. We continue this process until all the vertices are colored. This process results in a partition of the nodes of $T_C$ into stars $S_1, \ldots, S_k$, each star contains exactly one blue node $\mathbf{blue}(S_i)$, and a non empty sequence of red nodes $\mathbf{red}(S_i)$. Let $P = \{\mathbf{blue}(S_1), \ldots, \mathbf{blue}(S_k)\}$ and $Q = \{\mathbf{red}(S_1), \ldots, \mathbf{red}(S_k)\}$. Since for each $i \in [k]$ we have $\|\mathbf{blue}(S_i) - v\| \leq \varepsilon$ for every $v \in \mathbf{red}(S_i)$, the sequence of stars corresponds to a paired-walk with cost at most $\varepsilon$, and therefore we get $\mathrm{d}_{dF}(P,Q) \leq \varepsilon$. ◀

Note that Lemma 2 provides an algorithm for computing a partition of $S$ into two curves $P, Q$ such that $\mathrm{d}_{dF}(P,Q) \leq \varepsilon$. Given a spanning forest of $G_\varepsilon$, the running time for constructing $P, Q$ is $O(n)$, since each tree can be colored using a a simple BFS traversal. We conclude this in the following corollary.

▶ **Corollary 3.** *Given a spanning forest $T$ of $G_\varepsilon$ such that no vertex in $T$ has degree 0, a partition $P, Q$ of $S$ with $d_{dF}(P,Q) \leq \varepsilon$ can be constructed in $O(n)$ time.*

We wish to find the partition $P, Q$ of $S$ that minimizes $\mathrm{d}_{dF}(P,Q)$. Denote by $\varepsilon^*$ the distance between the curves in an optimal partition, i.e., there exists a partition $P^*, Q^*$ of $S$ with $\mathrm{d}_{dF}(P^*, Q^*) = \varepsilon^*$, and for any partition $P, Q$ of $S$ it holds that $\mathrm{d}_{dF}(P,Q) \geq \varepsilon^*$. In Claim 4 below, we show that $\varepsilon^*$ is the furthest nearest neighbor distance, i.e., the length of the longest edge in the Nearest Neighbor Graph of $S$.

Denote by $\mathrm{NNG}(S)$ the Nearest Neighbor Graph (NNG) of $S$, i.e., the graph whose vertices are the points of $S$, and there is an edge $\{u, v\}$ is the graph if and only if $v$ is a nearest neighbor of $u$ in $S$. Note that a point $u$ can have more than a single nearest neighbor.

In this case, we break ties by taking the point with the largest index to be the unique nearest neighbor. It is well-known that when applying such a tie-breaking rule, the NNG is a forest, and a subgraph of the Euclidean minimum spanning tree.

▷ Claim 4. Let $L$ be the longest edge in $\mathrm{NNG}(S)$. Then $\varepsilon^* = L$.

Proof. Notice that $\mathrm{NNG}(S)$ is a spanning forest of $G_L$ that does not contain any vertex of degree 0 (every point in $S$ has a unique nearest neighbor). Therefore, by Corollary 3 we get that there exists a partition $P, Q$ of $S$ with $\mathrm{d}_{dF}(P, Q) \leq L$.

Assume by contradiction that there is a partition $P', Q'$ of $S$ with $\mathrm{d}_{dF}(P', Q') = \varepsilon^* < L$, and consider a paired-walk $\pi$ along $P$ and $Q$ with cost $\varepsilon^*$. Then for any pair of points $v, u \in S$ that are matched in $\pi$, we have $\|v - u\| < L$. Let $\{w, x\}$ be the longest edge in $\mathrm{NNG}(s)$, so $\|w - x\| = L$, and $w, x$ are not matched in $\pi$. Therefore, there exists $w', x' \in S$ such that $w, w'$ are matched in $\pi$ and $x, x'$ are matched in $\pi$. We get that $\|w - w'\| < L$, so $x$ is not a nearest neighbor of $w$, and $\|x - x'\| < L$, so $w$ is not a nearest neighbor of $x$, a contradiction to $\{w, x\}$ being an edge of $\mathrm{NNG}(s)$. ◁

By Claim 4, all the edges of $\mathrm{NNG}(S)$ have length smaller or equal to the optimal distance $\varepsilon^*$, and therefore it is a spanning forest of $G_{\varepsilon^*}$. Thus, given $\mathrm{NNG}(S)$ as an input, the algorithm from Corollary 3 runs in $O(n)$ time. For $d = 2$, computing $\mathrm{NNG}(S)$ can be dome in $O(n \log n)$ time [17]. For general dimension $d$, computing $\mathrm{NNG}(S)$ can be done in $O(2^{O(d)} n \log n)$ time [25], so the overall running time for our problem is $O(n \log n)$ for any fixed dimension $d$.

We, therefore, obtain the following theorem.

▶ **Theorem 5.** *Given a set $S$ of $n$ points in $\mathbb{R}^d$, for any fixed dimension $d$, one can find two curves $P, Q$ that partition $S$ such that $d_{dF}(P, Q)$ is minimized in $O(n \log n)$ time.*

In the two remarks below, we suggest other ways to use Claim 4 and Corollary 3 for computing an optimal partition of $S$.

▶ Remark 6 (Using the Net and Prune framework). Har-Peled and Raichel [19] show that the furthest nearest neighbor distance (which, by Claim 4, equals $\varepsilon^*$) can be computed in expected linear time for points in any dimension $d$. Then, to apply Corollary 3, we need to compute a spanning forest $T$ of $G_\varepsilon$ such that no vertex in $T$ has degree 0. For this we can use the Net and Prune framework of [19] as follows. Put $S$ in a grid with cell diameter $\varepsilon^*$. Points that belong to the same grid cell form a clique in $G_\varepsilon$, and can be connected, for example, by some star graph. The lonely points, i.e., points that are alone in their cell, can be connected to a point in a neighbor cell (such a point must exist by the way we chose $\varepsilon^*$). Since the number of neighbor cells is $2^{O(d)}$, this results in an $O(2^d \cdot n)$ time algorithm for computing the spanning forest, and $O(2^{O(d)} \cdot n)$ expected time for computing the partition.

▶ Remark 7 (Using the MST). For points in dimension $d > 2$, we can use the the Minimum Spanning Tree of $S$ ($\mathrm{MST}(S)$) instead of $\mathrm{NNG}(S)$ as follows. We compute $\mathrm{MST}(S)$, and then iterate the edges from the longest to shortest. If removing an edge from $\mathrm{MST}(S)$ does not create a vertex of degree 0, remove it, and otherwise stop. Since $\mathrm{NNG}(S) \subseteq \mathrm{MST}(S)$, and because in $\mathrm{NNG}(S)$ there is no vertex of degree 0, the resulting graph contains $\mathrm{NNG}(S)$ and the length of its edges is at most $\varepsilon^*$. We can then execute the algorithm from Corollary 3 on this residue graph. Since $\mathrm{MST}(S)$ can be computed in $O(n^{2-2/\lceil d/2 \rceil + 1} + \varepsilon)$ time [2], this is also the total running time of the algorithm. Notice that for $d \leq \log \frac{n^k}{\log n}$, where $0 < k < 1 - 1/O(\log \frac{n}{\log n})$, we prefer to run the algorithm that finds the residue graph over computing $\mathrm{MST}(S)$. This is because the running time for finding the residue graph is $O(n^{2-2/\lceil k \cdot \log \frac{n}{\log n} / 2 \rceil + 1} + \varepsilon)$ while computing $\mathrm{NNG}(S)$ is in $O(n^{1+k})$.

## 4 Minimizing the lengths

In this section, we focus on problems 3 and 4. Let $S$ be a set of $n > 2$ points in the plane. Our goal is to find a partition $P, Q$ with $\mathrm{d}_{dF}(P, Q) = \varepsilon^*$. In problem 3, the partition also minimizes $\max\{\ell(P), \ell(Q)\}$. In problem 4 it also minimizes $\ell(P) + \ell(Q)$.

Let $\mathrm{TSP}(S)$ be a path on $S$ of minimum length. Denote by $\ell(G)$ the sum of edge lengths of a graph $G$ embedded in the plane.

### 4.1 Min-max discrete Fréchet-TSP

In this section, we prove the following theorem.

▶ **Theorem 8.** *Given a set $S$ of $n$ points in the plane, two curves $P, Q$ that partition $S$ such that $d_{dF}(P, Q) = \varepsilon^*$ and $\max\{\ell(P), \ell(Q)\} \leq 2.75 \cdot \ell(\mathrm{TSP}(S))$ can be found in $O(n \log n)$ time.*

By Theorem 5, computing the value $\varepsilon^*$ can be done in $O(n \log n)$ time. Let $\mathrm{MST}(S)$ be a minimum spanning tree of $S$, and observe that $\ell(\mathrm{MST}(S)) \leq \ell(\mathrm{TSP}(S))$. We thus focus on computing two curves $P, Q$ that partition $S$ such that $\mathrm{d}_{dF}(P, Q) = \varepsilon^*$ and $\max\{\ell(P), \ell(Q)\} \leq 2.75 \cdot \ell(\mathrm{MST}(S))$. For simplicity, we say that two points (or vertices) $u, v$ are ***close*** if $\|u - v\| \leq \varepsilon^*$, and otherwise we say that $u, v$ are ***far***.

▶ **Observation 9.** *Every vertex in $\mathrm{MST}(S)$ has at least one neighbor in $\mathrm{MST}(S)$ that is close to it.*

**Proof.** Assume by contradiction that all the neighbors of a vertex $v$ in $\mathrm{MST}(S)$ are far from $v$. By Lemma 2, $v$ has a neighbor $u$ in $G_{\varepsilon^*}$, and therefore $u$ is close to $v$. This contradicts the fact that the MST contains a nearest neighbor for each vertex (by Kruskal's algorithm). ◀

We now show how to construct two curves $P$ and $Q$ that partition $S$ by traversing $\mathrm{MST}(S)$ in a DFS order, starting from a leaf vertex of $\mathrm{MST}(S)$ as a root, and coloring the vertices **red** and **blue**. The blue vertices will be in $P$, and the red vertices in $Q$. The order of the points along the curves would be the same order in which they where colored during the algorithm.

Consider a vertex $v$ in the rooted tree $\mathrm{MST}(S)$. Let $I_{\mathrm{close}}(v)$ be the set of children of $v$ that are close to $v$, and similarly, $I_{\mathrm{far}}(v)$ will be the set of children of $v$ that are far from $v$. In addition, let $L(v) = \{u \mid u \in I_{\mathrm{close}}(v), I_{\mathrm{close}}(u) = \emptyset\}$. In other words, $L(v)$ is the set of ***lonely children*** of $v$ – those that are either leaves in $\mathrm{MST}(S)$, or that do not have close children – and therefore must be in $v$'s star. Notice that by Observation 9, all the nodes in $L(v)$ are close to $v$.

Let $v$ be the current vertex visited by the DFS algorithm. The invariant of our recursive DFS algorithm is that if $L(v)$ is empty, then there must be at least one vertex in $I_{\mathrm{close}}(v)$. The algorithm has two main steps: in the first step we color vertices in red and blue, and the second step contains the recursive calls.

**Coloring.** In this step, we color the entire star that $v$ belongs to using Algorithm 1. Note that $v$ can be either a center or a leaf in that star. Also, Algorithm 1 colors exactly one vertex in **blue** and the rest in **red**. We progress in recursion on the vertices of the tree. In each recursive call, we activate the coloring algorithm on newly colored vertices in the following order:

1. First, activate the algorithm on children of vertices that are colored **red**, in reversed order (last colored first called).
2. then, activate the algorithm on children of the **blue** vertex.

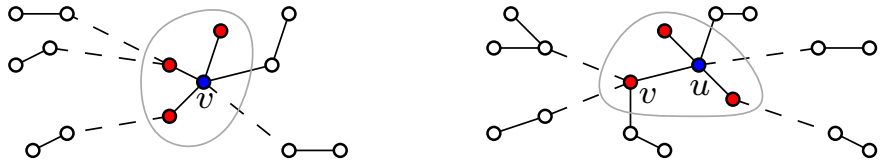


**Figure 3** Coloring of MST($S$). Dashed edges correspond to far neighbors. The star to which $v$ belongs is marked in gray.

**Algorithm 1** Coloring for Min-Length.

**Input:** Vertex $v$
**Output:** Color updates
**if** $L(v) \neq \emptyset$ **then**
  color the leaf nodes in $L(v)$ **red**;
  color all the other nodes in $L(v)$ **red**;
**else**
  Let $u$ be an arbitrary vertex in $I_{\text{close}}(v)$;
  color $v$ **red**;
  color $u$ **blue**;
  color the leaf nodes in $L(u)$ **red**;
  color all the other nodes in $L(u)$ **red**;

First, notice that when the algorithm is called with a vertex $v$ in a recursive step, then $v$ and its entire subtree were not colored yet. Moreover, we show that the following invariant holds in each step of the algorithm.

▷ Claim 10. In each step of the algorithm, if $L(v)$ is empty, then there must be at least one vertex in $I_{close}(v)$.

Proof. Assume by contradiction that in some step of the algorithm both $L(v)$ and $I_{\text{close}}(v)$ are empty. Then, by Observation 9, $v$ must have a parent $t$ in MST($S$) such that $v \in I_{\text{close}}(t)$, and $t$ was already colored by the algorithm. Moreover, $v$ is in $L(t)$, because $I_{\text{close}}(v) = \emptyset$. Therefore, it is not possible that $t$ was colored **blue**, because then $v$ would have been already colored **red** and would not be called recursively. In addition, $t$ is not a lonely child of its parent, because $I_{\text{close}}(t) \neq \emptyset$. Therefore, if $t$ was colored **red**, then it must be in step 2(a) of the algorithm, which means that $L(t)$ is empty, but this is not possible because $v \in L(t)$. ◁

Let $P = \{p_1, \ldots p_k\}$ (resp. $Q = \{q_1, \ldots q_m\}$) be the set of vertices that were colored **blue** (resp. **red**), in the order in which they were colored during the DFS scan.

▷ Claim 11. The running time for computing $P$ and $Q$ is $O(n \log n)$.

Proof. Calculating the Euclidean MST for the set $S$ takes $O(n \log n)$ time. A standard DFS traversal over the MST also requires $O(n)$ time. The additional overhead in the algorithm comes from recursively scanning the $L(v)$ children of each vertex $v$. Since $|L(v)| \leq 5$ in the EMST, this adds at most $O(n)$ additional operations. Thus, the total running time remains $O(n \log n)$. ◁

▷ Claim 12. $\mathrm{d}_{dF}(P, Q) \leq \varepsilon^*$.

Proof. We show that in each step of the algorithm, we color a star in $\mathrm{MST}(S)$ with edges of length at most $\varepsilon^*$. The center is colored **blue**, and the leaves **red**. Let $v$ be the current vertex. There are two cases:

- If $L(v)$ is not empty, then $v$ is colored **blue** and the nodes in $L(v)$ are colored **red**. Since $L(v) \subseteq I_{\mathrm{close}}(v)$ (by Observation 9), this red-blue star has edges of length at most $\varepsilon^*$.
- If $L(v) = \emptyset$, then the algorithm picks a vertex $u \in I_{\mathrm{close}}(v)$ and color it **blue**. This vertex becomes the center of a star with its children $v \cup L(u)$ which are colored **red**. Since $L(v) \subseteq I_{\mathrm{close}}(u)$, we again obtain a red-blue star with edges of length at most $\varepsilon^*$.

Since the vertices of $P$ and $Q$ are ordered by the time they were colored, we get that the sequence of starts corresponds to a paired-walk of cost at most $\varepsilon^*$ between $P$ and $Q$, as required. ◁

Next, we bound the lengths of the curves $P$ and $Q$ in relation to $\mathrm{MST}(S)$. Notice that the order in which we color the **blue** vertices (the vertices of $P$) follows a classic DFS preordering, and therefore we clearly have $\ell(P) \leq 2 \cdot \ell(\mathrm{MST}(S))$. However, the order in which we color the **red** vertices slightly differs from a classic DFS preordering. However, since all the children of $v$ that are colored in this step are close to $v$, the additional traversal overhead is small. The following claim together with Claim 12 implies Theorem 8.

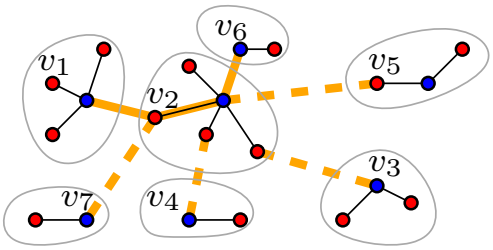


**Figure 4** An illustration of the algorithm. The root vertex is $v_1$, and the algorithm runs on $v_1, \ldots, v_7$ in this order. The stars are marked in gray, and the dashed edges mark far vertices. The orange edges are the set $W$ of edges that connect the stars.

▷ Claim 13. $\max\{\ell(P), \ell(Q)\} \leq 2.75 \cdot \ell(\mathrm{MST}(S))$.

Proof. Let $T$ be the path on all the points in $S$, which is obtained by traversing the edges of $\mathrm{MST}(S)$ in the order in which the algorithm colors the vertices, regardless of their color. Clearly $\ell(T) \geq \max\{\ell(P), \ell(Q)\}$. We show that $\ell(T) \leq 2.75 \cdot \ell(\mathrm{MST}(S))$.

Recall that the degree of any vertex in $\mathrm{MST}(S)$ is at most 5. Since we choose the root of $\mathrm{MST}(S)$ to be a leaf, then for every vertex $v$ in the rooted tree, $L(v)$ may contain at most 4 vertices. Consider a step of the algorithm were the current vertex is $v$. If $v$ was colored **blue**, denote $L(v) = u_1, \ldots, u_k$ for $1 \leq k \leq 4$. Then, the subpath of $T$ that was added in this step is $T_v = \{v, u_1, v, u_2, v, \ldots, u_k\}$. This is because, in $T$, we are moving from $u_i$ to $u_{i+1}$ through $v$, since $\{u_i, u_{i+1}\}$ is not an edge in $\mathrm{MST}(S)$. If $v$ was colored **red**, the algorithm picked a vertex $u \in I_{close}(v)$. Denote $L(u) = u_1, \ldots, u_k$ for $1 \leq k \leq 4$. The subpath of $T$ that was added in this step is $T_v = \{v, u, u_1, u, u_2, u, \ldots, u_k\}$.

We show how to charge the edges of $\mathrm{MST}(S)$ for each such star-subpath and for each of the edges that are connecting between star-subpaths. First, notice that in each star-subpath $T_v$, each edge of the star is traversed at most twice. Let $U = \cup_{x \in P} T_x$ be the set of edges that are charged for the star-subpaths themselves. Because the edges of the stars are disjoint, each edge in $U$ is charged at most twice.

Let $W$ be the set of edges that are traversed in $T$ when connecting between any two star-subpaths. Each edge of $W$ is charged at most twice because we traverse the stars following the classic DFS order. Therefore, in both sets $U, W$ each edge is charged at most twice. Notice that $U, W$ are distinct.

The edges in $U$ have length at most $\varepsilon^*$, so edges of length larger than $\varepsilon^*$ can only appear in $W$. In addition, edges that are incident to leaves in $\mathrm{MST}(S)$ appear only in $U$, because they do not connect between star-subpaths. We conclude that leaf-edges are charged only twice, and edges of length larger than $\varepsilon^*$ are also charged only twice.

Consider an edge $\{v, u\} \in T_v$ such that $L(v)$ is empty and $u \in I_{close}(v)$. Notice that the edge $\{v, u\}$ is charged only once in $U$ (it appears once in $T_v$) and once in $W$ (to connect $u$ with the centers of stars in $v$'s subtree), so in total it is charged at most twice.

The only case left to handle is edges $\{x, u_i\}$ in a star such that $u_i \in L(x)$ ($x$ can be either the current vertex or its child $u \in I_{\mathrm{close}}(v)$). In this case, $I_{close}(u_i) = \emptyset$. If $u_i$ is not a leaf, then there exists an edge $\{u_i, w_i\}$ such that $w_i \in I_{far}(u_i)$. Since the algorithm recursively runs on $w_i$, the edges $\{u_i, w_i\}$ is in $W$. In other words, for every $\{x, u_i\} \in U$ there exists an edge $\{u_i, w_i\} \in W \setminus U$ such that $\|u_i - w_i\| > \varepsilon^*$, which is charged only twice. Therefore, if $\{x, u_i\}$ is charged four times (twice in $U$ and twice in $W$), we charge it three times, and transfer the forth charge to $\{u_i, w_i\}$, so that both edges are charged only three times in total. Note that no other edge can transfer its charge to $\{u_i, w_i\}$, because $w_i$ has to come after $u_i$ in the order of the traversal.

This gives a bound of $3 \cdot \ell(\mathrm{MST}(S))$. To further improve the bound, notice that the last edge $\{x, u_k\}$ is charged only once in $U$. Moreover, it is also charged only once in $W$, because the children in its subtree are the first to be called recursively in this step, so $\{x, u_k\}$ only appears on the subpath that goes back to $x$. In the worst case, when $k = 4$, we charge the first three edges $\{x, u_1\}, \{x, u_2\}, \{x, u_3\}$ a total of 3 times (by transferring one charge to the corresponding edge in $W$), and $\{x, u_4\}$ is charged only twice. By choosing the farthest child to be colored last, i.e., $\|x - u_4\| \geq \max\{\|x - u_1\|, \|x - u_2\|, \|x - u_3\|\}$, each of the first three edges can transfer a charge of $\frac{1}{4}$ to $\{x, u_4\}$, resulting in a total charge of at most $2\frac{3}{4}$ per edge of $\mathrm{MST}(S)$, as claimed. ◁

Theorem 8 follows from Claim 11, Claim 12, and Claim 13.

## 4.2 Min-sum discrete Fréchet-TSP

In this section we consider the case where the goal is to minimize the sum of the lengths. The following theorem is a corollary of Theorem 8.

▶ **Theorem 14.** *Given a set $S$ of $n$ points in the plane, two curves $P, Q$ that partition $S$ such that $d_{dF}(P, Q) = \varepsilon^*$ and $\ell(P) + \ell(Q) \leq 4.75 \cdot \ell(\mathrm{MST}(S))$ can be found in $O(n \log n)$ time.*

**Proof.** Let $P$ and $Q$ be the curves obtained by the algorithm in the previous section. Then by Theorem 8 $\ell(Q) \leq 2.75 \cdot \ell(\mathrm{MST}(S))$. For $P$, note that it follows a DFS traversal of the $\mathrm{MST}(S)$ restricted to the centers. Since each edge is used at most twice and no vertex is repeated as in $Q$, this gives $\ell(P) \leq 2 \cdot \ell(\mathrm{MST}(S))$. Combining the two bounds yields $\ell(P) + \ell(Q) \leq 4.75 \cdot \ell(\mathrm{MST}(S))$. ◀

This result can be improved by using a $(1 + \varepsilon)$ approximation for TSP (for $\varepsilon > 0$) on one of the curves. We make use of the algorithm of Kisfaludi-Bak, Nederlof and Węgrzycki [20] which computes in $2^{O(\varepsilon^{1-d})} n \log n$ time. We get the following result.

▶ **Theorem 15.** *Given a set $S$ of $n$ points in $\mathbb{R}^d$, two curves $P, Q$ that partition $S$ such that $\ell(P) + \ell(Q) \leq (4+\varepsilon) \cdot \ell(\mathrm{MST}(S))$ can be found in $O(n \log n)$ time for constant $d, \varepsilon$.*

**Proof.** Given the set of stars $S_1, \ldots, S_k$ that were obtained in the proof of Lemma 2, each star $S_i$ has a **blue** vertex that is connected in $S_i$ to a set of **red** vertices. We run an algorithm that computes a $(1+\varepsilon)$ approximation for the TSP. On the **blue** vertices, and obtain a curve $P$ with the length $(1+\varepsilon) \cdot \ell(\mathrm{TSP}(S))$.

We then construct the curve $Q$ on the **red** vertices following the order of the **blue** vertices in $P$, i.e., for each vertex $v \in P$ that corresponds to the star $S_i$, we connect the red vertices of $S_i$, and then connect these red paths according to the TSP order on the **blue** vertices. We can bound the length of $Q$ by the length of a curve traversing $P$ with detours for traversing the **red** vertices. The sum of lengths of these detours is bounded by traversing each edge of a star twice, which is bounded by $2 \cdot \ell(\mathrm{MST}(S))$. We therefore get that $\ell(Q) \leq \ell(P) + 2 \cdot \ell(\mathrm{TSP}(S))$ and thus $\ell(Q) + \ell(P) \leq (4+\varepsilon) \cdot \ell(\mathrm{TSP}(S))$. ◀

## 4.3 Comparing to the optimal solution

In previous sections, we compared the solution obtained from our algorithm to $\mathrm{TSP}(S)$. We now show that it gives a constant approximation comparing to the optimal solution.

▶ **Observation 16.** *Let $P, Q$ be two curves that partition $S$, such that $d_{dF}(P, Q) = \varepsilon$ for some $\varepsilon > 0$. Then $\ell(\mathrm{TSP}(S)) \leq \ell(P) + \ell(Q) + \varepsilon$.*

**Proof.** By concatenating $P$ and $Q$, we obtain a path of at most $\ell(P) + \ell(Q) + \varepsilon$ that traverses all the points of $S$, which is a feasible solution for TSP on $S$. ◀

Denote by $P^*, Q^*$ two curves that partition $S$ with $\mathrm{d}_{dF}(P^*, Q^*) = \varepsilon^*$ such that $\max\{\ell(P^*), \ell(Q^*)\}$ is minimized.

▶ **Lemma 17.** *Let $P, Q$ be two curves that partition $S$, such that $d_{dF}(P, Q) = d_{dF}(P^*, Q^*) = \varepsilon^*$. Denote by $L_P$ (reps. $L_Q$) the maximum length of an edge in $P$ (resp. $Q$). If $|P|, |Q| \geq 2$, then $\varepsilon^* \leq L = \max\{L_P, L_Q\}$.*

**Proof.** Assume by contradiction that $\varepsilon^* > L$. That means that all edges of both $P$ and $Q$ are of strictly smaller length than $\varepsilon^*$. Denote $P = \{p_1, \ldots, p_k\}$ and $Q = \{q_1, \ldots, q_m\}$. If $k \geq 2$, then the discrete Fréchet distance between $P_{\mathrm{odd}} = \{p_1, p_3, p_5, \ldots\}$ and $P_{\mathrm{even}} = \{p_2, p_4, p_6, \ldots\}$ is at most $L$. Similarly, if $m \geq 2$, then the Fréchet distance between $Q_{\mathrm{odd}} = \{q_1, q_3, q_5, \ldots\}$ and $Q_{\mathrm{even}} = \{q_2, q_4, q_6, \ldots\}$ is at most $L$. Therefore, the discrete Fréchet distance between $P_{\mathrm{odd}} \circ Q_{\mathrm{odd}}$ and $P_{\mathrm{even}} \circ Q_{\mathrm{even}}$ is at most $L$, in contraction to the optimality of $\varepsilon^*$. ◀

Note that if one of $P, Q$ is a single vertex, the above lemma may not be correct. Let $S = \{p, q_1, q_2, \ldots\}$ such that $\|p - q_i\| = \varepsilon$ for some $\varepsilon > 0$. Let $\|q_i - q_j\| \ll \varepsilon$ for all $q_i, q_j \in Q$. Then $\varepsilon$ is the optimal Fréchet distance. And yet, the lemma does not hold for $P = \{p\}$ and $Q = \{q_1, q_2, \ldots\}$.

An immediate corollary of Observation 16 and Lemma 17 is that $\ell(\mathrm{TSP}(S)) \leq \ell(P) + \ell(Q) + \varepsilon \leq 3 \cdot \max\{\ell(P), \ell(Q)\}$, and $\ell(\mathrm{TSP}(S)) \leq 2 \cdot (\ell(P) + \ell(Q))$. Therefore, by Theorem 8 we have the follwiing corrolary.

▶ **Corollary 18.** *Given a set $S$ of $n$ points in the plane, we can find in $O(n \log n)$ time two curves $P$, $Q$ that partition $S$ such that $d_{dF}(P, Q) = \varepsilon^*$ and $\ell(P) + \ell(Q) = O(\ell(P^*) + \ell(Q^*))$, or $\max\{\ell(P), \ell(Q)\} = O(\max\{\ell(P^*), \ell(Q^*)\})$.*

## 5 Balancing the number of vertices

In this section, we address Problem 2 (balanced-Fréchet-TSP) under the discrete Fréchet distance. Clearly, it is possible that there is no pair of curves that partition $S$ and have both $\mathrm{d}_{dF}(P,Q) \leq \varepsilon^*$ (the optimal distance) and $\max\{|P|,|Q|\} = \lceil n/2 \rceil$ (see, e.g. Figure 5).

We show that for points in the plane our solution to discrete-Fréchet-TSP can be adjusted such that $\max\{|P|,|Q|\} \leq \lfloor n/2 \rfloor + 2$. In other words, we prove the following theorem.

▶ **Theorem 19.** *Given a set $S$ of points in the plane, there always exists two curves $P,Q$ that partition $S$, such that $d_{dF}(P,Q) \leq \varepsilon^*$, $|P| \geq |Q|$ and $|P| - |Q| \leq 4$ (and in case that $n$ is odd, $|P| - |Q| \leq 3$). Moreover, such curves can be found in $O(n \log n)$ time.*

**Proof.** Consider the set of stars $S_1, \ldots, S_k$ that were obtained in the proof of Lemma 2, and that can be computed in $O(n \log n)$ time by Theorem 5). Each star $S_i$ has a blue vertex $\mathbf{blue}(S_i)$ which is connected in the spanning tree $T_C$ to a set of red vertices $\mathbf{red}(S_i)$. For Theorem 5 we are using the nearest neighbor graph $\mathrm{NNG}(S)$, applying the unique nearest neighbor rule, and therefore the maximum degree in $\mathrm{NNG}(S)$ is at most 5. Therefore, for every $1 \leq i \leq k$, we have $\mathbf{red}(S_i) \leq 5$.

Notice that flipping the colors of the vertices in a star $S_i$ does not change the correctness of the algorithm, and we still obtain two curves that partition $S$ and have distance at most $\varepsilon^*$. Therefore, we can perform the following procedure. Let $P$ and $Q$ be the curves obtained from the algorithm of Theorem 5, then by construction $|Q| \geq |P|$. Set $w = |Q| - |P|$, and iterate over the stars $S_1, \ldots, S_k$: if $w \geq 5$, flip the colors in the current star, and update $w$. The algorithm terminates when $w \leq 4$, or after $S_k$ is flipped.

Since initially each star $S_i$ has exactly one vertex in $P$ (its center $\mathbf{blue}(S_i)$) and at most 5 vertices in $Q$ ($\mathbf{red}(S_i)$), a flip adds $c = |\mathbf{red}(S_i)| - 1 \leq 4$ vertices to $P$, and removes $c$ vertices from $Q$, which in total reduces $w$ by $2c \leq 8$. Therefore, after a flip we have $w \geq -3$. Assume by contradiction that the algorithm terminates with $w \geq 5$. Then all the stars $S_1, \ldots, S_k$ were flipped, but this is a contradiction because in the first step we had $w = |P| - |Q| > 0$, and thus after flipping all the stars we have $w = |Q| - |P| < 0$.

We conclude that when the algorithm terminates, we have $-3 \leq w \leq 4$. If $w < 0$, we flip all the stars, and get $0 \leq w = |Q| - |P| \leq 4$, as required. Finally, note if $n$ is odd, then $w$ must be odd, and therefore in this case we have $0 \leq w = |Q| - |P| \leq 3$. ◀

▶ Remark 20. In fact, the maximum degree of the NNG for points in $d$ dimensions is equal the kissing number of spheres in $d$ dimensions, and therefore Theorem 19 can be generalized to higher dimensions accordingly.

### 5.1 Relaxing the distance requirement

The example in Figure 5 shows a set $S$ for which there is no balanced partition with distance $\varepsilon^*$. However, notice that if we relax the requirement on the distance between the curves and allow it to be up to $2\varepsilon^*$, then we can split each star into smaller stars by connecting pairs of leaves, so that the maximum degree of a star becomes two, which allows for a balanced partition. Below we show that by allowing an even smaller relaxation (the discrete Fréchet distance will be at most $\sqrt{3} \cdot \varepsilon^*$), we can always obtain optimally balanced curves.

▶ **Theorem 21.** *Given a set $S$ of points in the plane, there always exists two curves $P,Q$ that partition $S$, such that $d_{dF}(P,Q) \leq \sqrt{3} \cdot \varepsilon^*$ and $\max\{|P|,|Q|\} = \lceil n/2 \rceil$. Moreover, such curves can be found in $O(n \log n)$ time.*

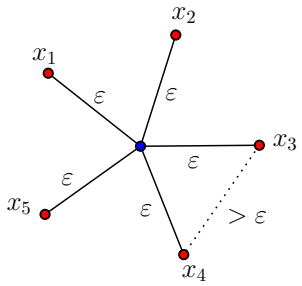


**Figure 5** A set of 6 points for which the optimal solution for Fréchet-TSP is $\varepsilon$, $|P| = 1$ and $|Q| = 5$.

**Proof.** Consider the set of stars $S_1, \ldots, S_k$ that were obtained in the proof of Lemma 2. If the degree of each star is at most two (i.e. $|\mathbf{red}(S_i)| \leq 2$ for every $1 \leq i \leq k$), then by applying arguments similar to the proof of Theorem 19, we get that $\max\{|P|, |Q|\} = \lceil n/2 \rceil$ as required.

Otherwise, let $S_i$ be a star such that $|\mathbf{red}(S_i)| \geq 3$. Let $c = \mathbf{blue}(S_i)$, then there is at least one pair of nodes $u, v \in \mathbf{red}(S_i)$ such that the smaller angle $\angle ucv$ at at most $\frac{2\pi}{3}$. Thus, by the law of cosines, the distance between them is

$$\|v - u\| = \sqrt{\|c - v\|^2 + \|c - u\|^2 - 2 \cdot \cos(\frac{2\pi}{3}) \cdot \|c - v\| \cdot \|c - u\|}.$$

Since the length of any edge in $S_i$ is at most $\varepsilon^*$, we have

$$\|v - u\| \leq \sqrt{2(\varepsilon^*)^2 - 2(\varepsilon^*)^2 \cos(\frac{2\pi}{3})} = \sqrt{2(\varepsilon^*)^2 + (\varepsilon^*)^2} = \sqrt{3} \cdot \varepsilon^*.$$

We now split $S_i$ into two stars: remove $v, u$ from $\mathbf{red}(S_i)$ and create a new star, $S_{k+1}$, with $\mathbf{red}(S_{k+1}) = \{v\}$ and $\mathbf{blue}(S_{k+1}) = u$.

We continue this process until all our stars have degree at most two. Notice that any star that we add consists of a single pair of points with distance at most $\sqrt{3} \cdot \varepsilon^*$. Therefore, when applying the arguments from the proof of Theorem 19 as before, we obtain $\max\{|P|, |Q|\} = \lceil n/2 \rceil$, and $\mathrm{d}_{dF}(P, Q) \leq \sqrt{3} \cdot \varepsilon^*$. ◀

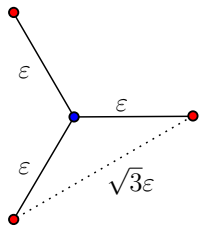


**Figure 6** A set of 4 points for which the optimal solution for Fréchet-TSP is $\varepsilon$, $|P| = 1$ and $|Q| = 3$. By allowing the distance to be $\sqrt{3}\varepsilon$, we can obtain a solution where $|P| = |Q| = 3$.

The example in Figure 6 shows that the bounds in Theorem 19 and Theorem 21 are tight, as we conclude in the observation below.

▶ **Observation 22.** *There exists a set $S$ of points in the plane such that: (i) for any two curves $P, Q$ that partition $S$ and have $d_{dF}(P, Q) \leq \varepsilon^*$ it holds that $|P| - |Q| \geq 4$, and (ii) for any two curves $P, Q$ that partition $S$ and have $|P| - |Q| = 0$ it holds that $d_{dF}(P, Q) \leq \sqrt{3} \cdot \varepsilon^*$.*

## 5.2 Tighter balancing

In the proof of Theorem 19 we flip the colors of some stars in order to obtain a difference of four between $|P|$ and $|Q|$. However, for some instances there might still be a different way to arrange the stars (i.e., a different paired-walk) that allows for a more balanced partition. Therefore, to further improve the difference, we now show how to split some of the starts (as we do in the proof of Theorem 21) in order to reduce the difference between $|P|$ and $|Q|$, while still having $\mathrm{d}_{dF}(P,Q) \leq \varepsilon^*$.

Given a star subgraph $H$ of $G_{\varepsilon^*}$, define its ***balance*** as $\mathbf{b}(H) = \mathbf{deg}(H) - 1$, where $\mathbf{deg}(H)$ is the degree of the center node of $H$. We call a star $H$ with balance $\mathbf{b}(H)$ a $\mathbf{b}(H)$-star (e.g., a 0-star, a 1-star, etc.). For a set $X$ of star subgraphs of $G_{\varepsilon^*}$, denote $\mathbf{b}(X) = \sum_{H \in X} \mathbf{b}(H)$.

Let $\alpha = \{H_1, \ldots, H_t\}$ be a set of disjoint stars $H_1, \ldots, H_t$ in $G_{\varepsilon^*}$ that together cover $S$. We call such a set a ***star cover*** of $S$. Consider a valid coloring of the stars in $\alpha$ in two colors, then as in the proof of Lemma 2 this coloring yields two curves that partition $S$ and a paired-walk along them with cost at most $\varepsilon$. Intuitively, $\mathbf{b}(S_i)$ is the amount that a star contributes to either $|P|$ and $|Q|$. Therefore, the following problem is equivalent to Problem 2 (balanced-Fréchet-TSP).

▶ **Problem 5.** *Given a set $S$ of $n$ points in the plane and a value $\varepsilon > 0$, find a star cover $\alpha = H_1, \ldots, H_t$ of $S$, that can be partitioned into two sets of stars $\rho$ and $\beta$, that minimizes the difference*

$$\Delta_\alpha(\rho, \beta) = |\boldsymbol{b}(\rho) - \boldsymbol{b}(\beta)| .$$

We thus focus on solving the problem defined above. For a given star cover $\alpha$, an ***optimally balanced partition*** of $\alpha$ is a partition of $\alpha$ into two sets of stars $\rho$ and $\beta$ that minimizes $\Delta_\alpha \coloneqq \Delta_\alpha(\rho, \beta)$. In the full version of the paper, we show that when $|S| = n$ is even, any star cover $\alpha$ of $S$ with $\Delta_\alpha = 4$ consists of only 0-stars and 4-stars. We then present an algorithm that runs in $O(n^2)$ time, and either returns a star cover $\alpha$ of $S$ with $\Delta_\alpha \leq 2$, or reports (correctly) that the optimal star cover has $\Delta_\alpha = 4$.

▶ **Theorem 23.** *Given a set $S$ of $n$ points in the plane, where $n$ is even, there exists an algorithm that runs in $O(n^2)$ time, which returns a star cover $\alpha$ of $S$ s.t. either:*

1. *$\Delta_\alpha = 4$ and it is the minimum possible value of any star cover of $S$ (i.e. $\alpha$ must be optimal).*
2. *$\Delta_\alpha \leq 2$.*

# 6 Continuous Fréchet-TSP is hard

As mentioned in the introduction, Buchin and Kilgus [11] show that the continuous Fréchet distance between 2 point sets is NP-hard. Similarly, we show that the Fréchet-TSP problem under the continuous Fréchet distance is also NP-hard. The reduction is quite similar to the reduction described in [11], and for completeness, we include a sketch of the proof. We reduce from a restricted version of 3-SAT, called $(3, B2)$-SAT, in which each clause contains exactly three literals, and each literal (that is, a variable or its negation) appears exactly twice in the entire formula. As in [11], we assume that in the $(3, B2)$-SAT instance no two clauses share more than one literal (this is possible by the construction in [8]).

Given a $(3, B2)$-SAT formula, we construct a set $S$ of points in the plane such that the formula is satisfiable if and only if there exist two curves $P, Q$ that partition $S$ and have $d_F(P,Q) \leq \varepsilon$. Our construction is as follows. Each clause in the formula is represented by a

single point, where three line segments intersect, with one segment for each literal in the clause. This is possible because no two clauses share more than one literal. Since each literal appears at most twice in the formula, the line segment corresponding to a literal passes through the two points representing the clauses containing that literal. The variable gadgets are then created in a way for each clause point, one curve must visit it, and the other curves must pass close to it. This is possible because no two clauses share more than one literal. Since each literal appears in at most two clauses, the line segment corresponding to a literal passes through the two points representing the clauses containing that literal. This ensures that each clause is "covered" by the appropriate combination of curves according to the truth assignment encoded by the variable gadgets.

To make this construction concrete, we now describe the variable gadgets in detail. For every input variable $x$ in the formula, we build a gadget similar to the one described in [11]. Each gadget has two points $x_1, x_2$ that represent the positive literal, and two points $\bar{x}_1, \bar{x}_2$ that represent the negative literal, see Figure 7 below.

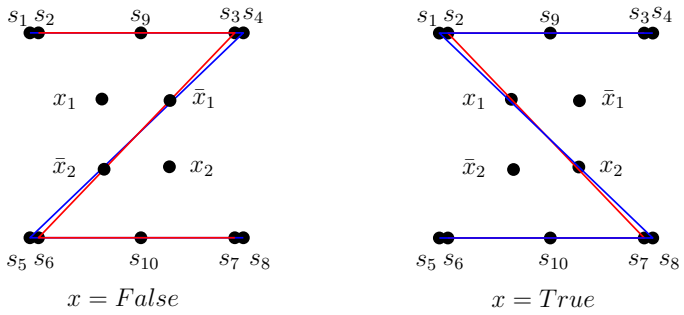


**Figure 7** Variable $x$ gadget.

The key difference for the 1-set case, compared to the original two–point-set case in [11], is that the vertices are not yet assigned to the curves. In order to force the partition to be as in [11], we place the points at the corners of the variable gadgets at distance $\varepsilon > 0$, for small enough $\varepsilon$. If the formula is stisfiable, then the optimal Fréchet distance will be $\varepsilon$,because each pair of corner points can be matched while still passing through all the clause points, as illustrated in Figure 7.

More precisely, the gadgets consists of four pairs of points $(s_1, s_2)$, $(s_3, s_4)$, $(s_5, s_6)$ and $(s_7, s_8)$, each pair of points is placed so the distance between them is very small $\varepsilon$. This ensures that both curves are forced to pass through all four corners, and, in order to preserve the uniqueness of each point along the curves, they cannot complete a full loop around the gadget and return to their starting corner. As a result, the traversal of the gadget is restricted, so that each curve may follow only one of the two diagonals. This restriction allows the gadget to enforce that exactly one of the line segments corresponding to $x$ or $\bar{x}$ is visited by one of the curves (and the other one passes close enough to it), corresponding to the chosen assignment of the 3SAT variable.

For the clauses in the $(3, B2)$-Linear-SAT formula, we intersect the 3 different gadgets in one of the $x_1, x_2, \bar{x}_1, \bar{x}_2$ representing the variables in the clause, this is constructed in the same way as [11], as once the variable gadget are built the difference of the points being preassigned a curve or not does not affect the construction anymore, Figure 8.

▶ **Theorem 24.** *There is no polynomial-time solution for uniquely partitioning $S$ into curves $P, Q$ with minimal Fréchet distance, unless $P = NP$.*

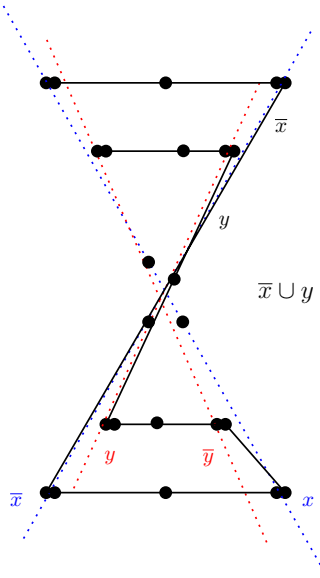


**Figure 8** $(3, B2)$-SAT to 1 set continuous Fréchet problem.

▶ Remark 25. Partitioning $S$ into curves $P, Q$ with minimal weak Fréchet distance, also cannot be done in polynomial time unless $P = NP$. This follows directly from the proof over the Fréchet distance that also applies for the week case. Notice that the proof does not apply for the non-unique case, where we allow the same point to repeat on the same curve, but not in both because then we can just take identical curves over all set $S$.

## References


1 Paul Accisano and Alper Üngör. Hardness results on curve/point set matching with fréchet distance. *CoRR*, abs/1211.2030, 2012. `arXiv:1211.2030`.

2 Pankaj K Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 203–210, 1990. `doi:10.1145/98524.98567`.

3 Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995. `doi:10.1142/S0218195995000064`.

4 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998. `doi:10.1145/290179.290180`.

5 Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006. `doi:10.1016/j.omega.2004.10.004`.

6 Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3d polygonal chains under the discrete fréchet distance. In *LATIN 2008: Theoretical Informatics*, pages 630–641, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. `doi:10.1007/978-3-540-78773-0_54`.

7 Mark de Berg, Kevin Buchin, Bart MP Jansen, and Gerhard Woeginger. Fine-grained complexity analysis of two classic tsp variants. *ACM Transactions on Algorithms (TALG)*, 17(1):1–29, 2020. `doi:10.1145/3414845`.

8 Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electron. Colloquium Comput. Complex.*, TR03-049, 2003. URL: `https://eccc.weizmann.ac.il/eccc-reports/2003/TR03-049/index.html`.

**9** Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 661–670. IEEE, 2014. doi:10.1109/FOCS.2014.76.

**10** Kevin Buchin, Maike Buchin, Wouter Meulemans, and Bettina Speckmann. Locally correct fréchet matchings. *Comput. Geom.*, 76:1–18, 2019. doi:10.1016/J.COMGEO.2018.09.002.

**11** Maike Buchin and Bernhard Kilgus. Fréchet distance between two point sets. *Comput. Geom.*, 102:101842, 2022. doi:10.1016/J.COMGEO.2021.101842.

**12** Omar Cheikhrouhou and Ines Khoufi. A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy. *Computer Science Review*, 40:100369, 2021. doi:10.1016/j.cosrev.2021.100369.

**13** Siu-Wing Cheng, Haoqiang Huang, and Shuo Zhang. Constant approximation of fréchet distance in strongly subquadratic time. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 2329–2340. ACM, 2025. doi:10.1145/3717823.3718157.

**14** Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013. doi:10.1137/120865112.

**15** Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *J. Math. Imaging Vis.*, 27(3):203–216, 2007. doi:10.1007/S10851-006-0647-0.

**16** Thomas Eiter, Heikki Mannila, et al. Computing discrete fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, Technische Universität Wien, 1994.

**17** David Eppstein, Michael S Paterson, and F Frances Yao. On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282, 1997. doi:10.1007/PL00009293.

**18** Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, 1906.

**19** Sariel Har-Peled and Benjamin Raichel. Net and prune: A linear time algorithm for euclidean distance problems. *Journal of the ACM (JACM)*, 62(6):1–35, 2015. doi:10.1145/2831230.

**20** Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Węgrzycki. A gap-ETH-tight approximation scheme for Euclidean TSP. *Journal of the ACM*, 72(6):1–48, 2025.

**21** Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. Fréchet distance with speed limits. *Comput. Geom.*, 44(2):110–120, 2011. doi:10.1016/J.COMGEO.2010.09.008.

**22** Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. Staying close to a curve. In *CCCG*, 2011.

**23** Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

**24** David Mount and Mary Monroe. A ptas for the min-max euclidean multiple tsp, December 2021. doi:10.48550/arXiv.2112.04325.

**25** Pravin M Vaidya. An o (n log n) algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 4:101–115, 1989. doi:10.1007/BF02187718.

**26** Tim Wylie. Discretely following a curve. In *International Conference on Combinatorial Optimization and Applications*, pages 13–24. Springer, 2013. doi:10.1007/978-3-319-03780-6_2.