Algorithmica
https://doi.org/10.1007/s00453-024-01267-8

On Flipping the Fréchet Distance

Omrit Filtser[1] · Mayank Goswami[2] · Joseph S. B. Mitchell[3] · Valentin Polishchuk[4]

Received: 30 April 2023 / Accepted: 14 August 2024

Abstract

The classical and extensively-studied *Fréchet distance* between two curves is defined as an *inf max*, where the infimum is over all traversals of the curves, and the maximum is over all concurrent positions of the two agents. In this article we investigate a "flipped" Fréchet measure defined by a *sup min* – the supremum is over all traversals of the curves, and the minimum is over all concurrent positions of the two agents. This measure produces a notion of "social distance" between two curves (or general domains), where agents traverse curves while trying to stay as far apart as possible. We first study the flipped Fréchet measure between two polygonal curves in one and two dimensions, providing conditional lower bounds and matching algorithms. We then consider this measure on polygons, where it denotes the minimum distance that two agents can maintain while restricted to travel in or on the boundary of the same polygon. We investigate several variants of the problem in this setting, for some of which we provide linear-time algorithms. We draw connections between our proposed flipped Fréchet measure and existing related work in computational geometry, hoping that our new measure may spawn investigations akin to those performed for the Fréchet distance, and into further interesting problems that arise.

Keywords Social distancing · Fréchet distance · Conditional lower bounds · Motion planning

✉ Omrit Filtser
omrit.filtser@gmail.com

Mayank Goswami
mayank.goswami@qc.cuny.edu

Joseph S. B. Mitchell
joseph.mitchell@stonybrook.edu

Valentin Polishchuk
valentin.polishchuk@alumni.stonybrook.edu

[1] The Open University of Israel, Ra'anana, Israel

[2] Queens College CUNY, New York, USA

[3] Stony Brook University, Stony Brook, USA

[4] Linköping University, Linköping, Sweden

Published online: 20 September 2024

1 Introduction

The classical Fréchet distance between two curves P and Q is defined as the minimum length of a leash required for a person to walk their dog, with the person and the dog traversing P and Q from start to finish, respectively. Inspired by the challenge of maintaining social distancing among groups and individuals, we consider the question of developing a notion opposite to the Fréchet distance, where instead of keeping the agents close (short leash), we keep them as far apart as possible.

In this paper we propose a new measure, called the *Flipped Fréchet* measure, to capture the amount of social distancing possible while traversing two curves. While Fréchet distance is defined as an *inf max*, where the infimum is over all traversals of the curves, and the maximum is over all concurrent positions of the two agents, the flipped Fréchet measure[1] is defined as a *sup min* – the supremum is over all traversals of the curves, and the minimum is over all concurrent positions of the two agents. How efficiently can this measure be computed, for curves in one or two dimensions, as well as for other domains? Such questions have been considered for Fréchet distance, and in this paper we initiate their study for the flipped Fréchet measure.

We refer to the two agents as "Red" and "Blue" henceforth. Considering the social distancing problem further, what if Blue is not restricted to move along some given curve; rather, it can choose its own path? We now start arriving at a class of problems that have no analogues in the Fréchet version. Of course, if Blue had no restrictions at all, it could just go to infinity and thus be far from Red (on any path). It therefore makes sense to restrict the domain for Blue, e.g. to a simple polygon P. We consider questions regarding the complexity of calculating a strategy for Blue to stay away from Red, when Red is traveling on a given path, which may or not be a geodesic in P. In particular, we define the *social distance width* (SDW for short) of P to be the minimum distance that Blue can maintain, moving anywhere within P, while Red circulates around the boundary of P, and study algorithms to compute the SDW.

In addition to developing algorithms for the general versions of the above problems, we also consider special scenarios which facilitate faster algorithms. For example, while our algorithm for computing the SDW of a polygon runs in quadratic time, we also define the SDW of a plane tree (a "skinny" polygon) and show how to compute it in linear time.

Although we mostly study the case of $k = 2$ agents, in general one may be given k agents and k associated domains; each agent is restricted to move only within its respective domain, and at least one of the agents has some *mission*, e.g., to move from a given start point to a given end point, or to traverse a given path inside the domain. In addition, the domains may be shared or distinct, and different agents may have different speeds. The goal is to find a movement strategy for all the agents, such that the *minimum pairwise* distance between the agents at any time is maximized. Additionally, one may seek to minimize the time necessary to complete one or more missions.

This new class of problems is different from the usual motion planning problems for robots, or disjoint disks: most, if not all, literature on robot motion planning assumes

[1] One observes that this measure is not a metric/distance as it does not satisfy the triangle inequality.

robots are cooperating on some task. One then considers optimizing objectives like makespan, or total distance travelled, etc. However, the kind of movement we consider is far from cooperative – in fact, some agents may not care about social distancing, while others do: some may be "on a mission" while others are just trying to maintain a safe distance.

Related Work. The Fréchet distance is an extensively investigated distance measure for curves, starting with the early work of Alt and Godau [1] in '95. There is a quadratic-time algorithm for computing it [1, 12], and it was recently shown [4] that under the Strong Exponential Time Hypothesis (SETH), no subquadratic algorithm exists, not even in one dimension [5]. Moreover, under SETH, no subquadratic algorithm exists for approximating the Fréchet distance within a factor of 3 [7].

The problem of coordinating collision-free motion of two agents traveling on polygonal curves was considered already in '89 by O'Donnell and Lozano-Perez [18], in the context of robot manipulators. Assuming some additional restrictions on the movements of the agents (e.g. robots are not allowed to simultaneously traverse segments that are too close), they give an $O(n^2 \log n)$ algorithm for minimizing the completion time.

There is an extensive literature on related problems of motion planning in robotics. Perhaps most closely related to our work is that of coordinated motion planning of 2 or more disks; see [10], on nearly optimal (in terms of lengths of motions) rearrangements of multiple unit disks and the related work of [16]. (In our problems, instead of minimizing length of motion for given radius disks, we seek to maximize the radii.)

The problem of computing safe paths for multiple speed-bounded mobile agents that must maintain separation standards arises in air traffic management (ATM) and Aircraft/Train Scheduling applications. [2] studied the problem of computing a large number of "thick paths" for multiple speed-bounded agents, from a source region to a sink region, where the thickness of a path models the separation standard between agents, and the objectives are to obey speed bounds, maintain separation, and maximize throughput. In the Aircraft/Train Scheduling problem (see [10, 20, 21]), given a set of paths on which the agents travel, and a separation parameter, the goal is to find a collision-free motion of the agents while minimizing the time of completion. (In our problems, we are not maximizing a "throughput" or makespan; rather, we maximize a separation standard, for the given agents.)

In the *maximum dispersion* problem, the goal is to place n (static) points within a domain P in order to maximize the minimum distance between two points. (Optionally, one may also seek to keep points away from the boundary of P.) An optimal solution provides maximum social distancing for a set of *static* agents, who stand at the points, without moving. Constant-factor approximations for the problem are known [3, 13]. (The problem is also closely related to geometric packing problems, which is a subfield in itself.) In robotics, the problem of motion planning in order to achieve well dispersed agents has also been studied: move a swarm of robots, through "doorways", into a geometric domain, in order to achieve a set of agents well dispersed throughout the domain. Such movements can be accomplished using local strategies that are provably competitive [17].

In the adversarial setting, in which one or more agents is attempting to move in order to avoid (evade) a pursuer, there is considerable work on pursuit-evasion in geometric domains (e.g., the "lion and man" problem); see the survey [8].

Our results. In this paper, we (mostly) consider the case of $k = 2$, i.e., two agents, "Red" and "Blue", that move inside their given domains. Further, unless stated otherwise, we do not consider speed to be a limiting factor; e.g., when Blue moves in order to maintain distance from Red, we assume that Blue can move at a sufficient speed.

We begin by considering the scenario in which the two domains are polygonal curves R and B. The agents' missions are to traverse their respective curves, from the start point to the end point, in order to maximize the minimum distance between the agents. The Flipped Fréchet measure between the two curves is the maximum separation that can be maintained. In Sect. 2 we consider both the continuous case (agents move continuously along the edges of their curves), and the discrete case (agents "jump" between consecutive vertices of their curves). We first show that the Flipped Fréchet measure between two curves in one dimension (1D) can be computed in linear time. This is in sharp contrast with continuous Fréchet distance, which has quadratic conditional (SETH-based) lower bounds in 1D [7]. We then develop quadratic or near-quadratic time algorithms for computation of discrete Flipped Fréchet measure in 1D and 2D, and for 2D continuous Flipped Fréchet measure. We also complement our quadratic-time algorithms with conditional lower bounds (conditioned on the Orthogonal vectors (OV) problem), even for approximation: we give a quadratic conditional lower bound on approximating SDW for curves in 2D up to a factor better than $\frac{\sqrt{5}}{2\sqrt{2}}$, and a quadratic conditional lower bound on approximating discrete SDW for curves in 1D, with a factor better than $\frac{2}{3}$.

We then restrict the domain for Blue to a simple polygon P, and measure separation using geodesic distance in P. In Sect. 3 we consider several versions. In the first, the Red agent has a mission to walk along a given path inside the polygon. The Blue agent must stay as far as possible from Red, and the only restriction is to move inside P. We give a quadratic-time algorithm for this problem. We then show that under the reasonable assumption that Red moves on a geodesic, one can compute a strategy for Blue in near-linear time.

Next, we consider a simple polygon P where the Red agent is on a mission to traverse the boundary of P, while the Blue agent moves within P (with a starting point of Blue's choice), in order to maximize the minimum Red-Blue distance. We define the *social distance width* (SDW for short) of a polygon P to be the minimum Red–Blue distance that can be maintained throughout the movement, maximized over all possible movement strategies. We develop a quadratic-time algorithm to compute the SDW of a polygon. We also show that when P is a tree (a "skiny" polygon), a strategy for Blue can be computed in linear time.

2 Flipped Fréchet Measure on Polygonal Curves

In this section the domains of Red and Blue are two polygonal curves R and B, respectively. We begin by giving some basic definitions; then, we describe tools that were used in classic algorithms for Fréchet distance and the relation to the social distancing problem for curves.

A polygonal curve P in $\mathbb{R}^d$ is a continuous function $P : [1, n] \to \mathbb{R}^d$, such that for any integer $1 \le i \le n-1$ the restriction of P to the interval $[i, i+1]$ forms a line segment. We call the points $P[1], P[2], \ldots, P[n]$ the vertices of P, and say that n is the length of P. For any real numbers $\alpha, \beta \in [1, n]$, $\alpha \le \beta$, we denote by $P[\alpha, \beta]$ the restriction of P to the interval $[\alpha, \beta]$. Then, for any integer $1 \le i \le n-1$, $P[i, i+1]$ is an edge of P. A continuous, non-decreasing, surjective function $f : [0, 1] \to [1, n]$ is called a *traversal* of P.

Let $P : [1, n] \to \mathbb{R}^d$ and $Q : [1, m] \to \mathbb{R}^d$ be two polygonal curves. A *traversal* of P and Q is a pair $\tau = (f, g)$, with $f : [0, 1] \to [1, n]$ a traversal of P, $g : [0, 1] \to [1, m]$ a traversal of Q.

Definition 1 (Flipped Fréchet Measure) The *Flipped Fréchet measure* (FF) of P and Q is $\mathrm{FF}(P, Q) = \sup_{\tau=(f,g)} \min_{t\in[0,1]} \|P(f(t)) - Q(g(t))\|$, where τ is a traversal of P and Q.

Note that the well-studied *Fréchet distance* between P and Q is $\inf_{\tau=(f,g)} \max_{t\in[0,1]} \|P(f(t)) - Q(g(t))\|$, where τ is a traversal of P and Q.

In the discrete case, we simply define a polygonal curve P as a sequence of n points $P[1], \ldots, P[n]$ (the vertices of P) in $\mathbb{R}^d$. For any $1 \le i \le j \le n$ let $P[i, j] = (P[i], P[i+1], \ldots, P[j])$ be a subcurve of P.

Consider two sequences of points P, Q of length n and m, respectively. A *traversal* τ of P and Q is a sequence, $(i_1, j_1), \ldots, (i_t, j_t)$, of pairs of indices such that $i_1 = j_1 = 1$, $i_t = n$, $j_t = m$, and for any pair (i, j) it holds that the next pair is $(i, j+1)$, $(i+1, j)$, or $(i+1, j+1)$.

Definition 2 (Discrete Flipped Fréchet Measure) The *discrete Flipped Fréchet measure* (dFF) of P and Q is $\mathrm{dFF}(P, Q) = \max_{\tau} \min_{(i,j)\in\tau} \|P[i] - Q[j]\|$, where τ is a traversal of P and Q.

Notice that unlike in the continuous case, the distances between the agents are only calculated at the vertices of the polygonal curves.

The *discrete Fréchet distance* (DFD) between P and Q is $\min_\tau \max_{(i,j)\in\tau} \|P[i] - Q[j]\|$, and it can be computed in $O(nm)$ time [12] using a simple dynamic programming algorithm.

From now on, we assume for simplicity that both curves R and B have length n; however, our algorithms and proofs can be easily adapted to the general case of $m \ne n$.

We give (Sect. 2.1) a linear time algorithm to compute the continuous FF measure in 1D, demonstrating that "flipping" the objective function makes this setting easier: for continuous Fréchet there exist conditional quadratic lower bounds [5, 7]. We give quadratic algorithms and then conditional lower bounds (Sects. 2.2 and 2.3) for computing or approximating other variants (1D discrete, 2D continuous and discrete) of FF measure, specifically:

- A quadratic lower bound, conditioned on the Strong Exponential Time Hypothesis (SETH), on approximating FF measure for curves in 2D, with approximation factor $\frac{\sqrt{5}}{2\sqrt{2}}$.

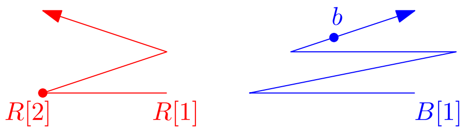


Fig. 1 Instead of τ, move Red to $R[2]$, then Blue to b; then continue τ

- A quadratic lower bound, conditioned on the Strong Exponential Time Hypothesis (SETH), on approximating dFF measure for 1D curves, with approximation factor $\frac{2}{3}$.

2.1 A Linear Time Algorithm for FF in 1D

Let R and B be two paths on the x-axis, each specified by n points. We will make a series of simplifying assumptions about the paths, arguing that each assumption can be made without loss of generality. In the end, each path will have monotonically growing extreme points, i.e., each vertex v will be an extreme point of the subpath to v; for such paths, a simple linear-time greedy algorithm will find the flipped Fréchet distance.[2]

First, assume that $R[1] < B[1]$. Next, assume that none of the paths has subdivided edges, i.e., that each (internal) vertex v is a local extremum (both incident edges lie on the same side of the vertical line through v). The following lemma allows us to assume that $R[1] < R[2]$ (and symmetrically that $B[1] > B[2]$):

Lemma 3 *If* $R[2] < R[1]$, *then* $FF(R, B) = \min\{\, B[1] - R[1])\ ,\ FF(R[2, n], B)\,\}$

Proof Obviously, $\text{FF}(R, B) \leq B[1] - R[1]$. Consider an optimal traversal τ of R, B, achieving $\text{FF}(R, B)$. Let $b \in B$ be the point where Blue is when Red is on $R[2]$ (Fig. 1). Since $R[2]$ is left of any point of the first segment $R[1, 2]$ of R, the configuration $(R[2], b)$ can be reached, without decreasing the minimum Red–Blue distance as follows (instead of τ): move Red to $R[2]$ while Blue sits on $B[1]$, then move Blue to b. Then the traversals of R, B can be finished using τ. □

The proof of Lemma 3 modified a traversal so that Red moved while Blue was sitting on a vertex, and Blue moved while Red was sitting on a vertex. This is true in general:

Lemma 4 *There is an optimal traversal such that at any time either Red or Blue is on a vertex.*

Proof The argument is similar to the proof of Lemma 3: Suppose that in some traversal τ, Red and Blue passed through a configuration $(R[i], B[j])$ for some vertices $R[i] \in R$, $B[j] \in B$, but later reached a configuration $(\bar{r}, \bar{b})$ with $\bar{r}, \bar{b}$ being interior to edges $R[i, i+1]$ and $B[j, j+1]$ resp. (Fig. 2); as in the proof of Lemma 3, we will consider the location $r' \in R$ where τ has Red when Blue is at $B[j+1]$, and the location $b' \in B$ where τ puts Blue when Red is at $R[i+1]$ (i.e., $(r', B[j+1])$ and $(R[i+1], b')$ are

[2] We thank an anonymous reviewer for suggesting this linear-time algorithm; the conference version [14] of this paper reported an $O(n \log^2 n)$-time solution.

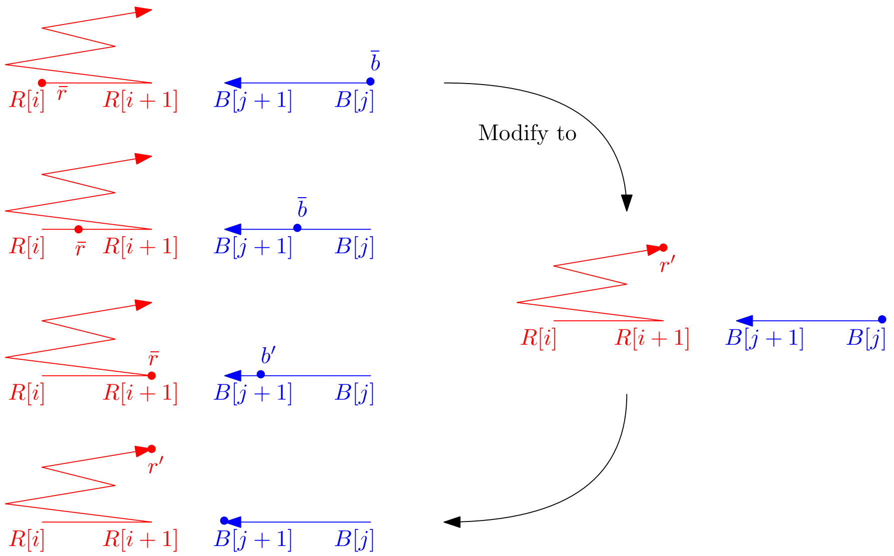


Fig. 2 The red and blue dots show configurations through which Red and Blue go, the time is increasing from top to bottom. Left: original traversal. Right: modified traversal which always keeps one agent on a vertex

configurations from τ) – we will modify τ depending on whether the edges "face each other" (i.e., $R[i] < R[i+1]$, $B[j+1] < B[j]$), or at least one of the edges is not "directed towards the other agent" (w.l.o.g. assume that $B[j] < B[j+1]$):

- $R[i] < R[i+1]$, $B[j+1] < B[j]$ (Fig. 2). Assume w.l.o.g. that Red reaches $R[i+1]$ before Blue reaches $B[j+1]$ (the other case is symmetric). Modify τ so that Blue sits at $B[j]$ while Red goes to r'; then move Blue to $B[j+1]$. Since $B[j+1] \leq b \ \forall b \in B[j, j+1]$, in the modified traversal, Red and Blue maintain at least as large a separation as in τ.
- $B[j] < B[j+1]$. Let Red sit on $R[i]$ until Blue reaches $B[j+1]$; then let Red go to r'. It can be seen by inspection that in the modified traversal, Red and Blue maintain at least as large a separation as in τ.

□

Of course, there is no need for Blue to sit on a local minimum (and for Red – on a local maximum):

Lemma 5 *There is an optimal traversal such that at any time either Red is on a local minimum of R or Blue is on a local maximum of B.*

Proof Consider an optimal traversal τ in which at any time one of the agents is on a vertex (Lemma 4). Suppose that τ has Blue on a local minimum b while Red is at a point r in the interior of an edge. It follows that Red should reach a vertex r' before Blue moves. Since the configuration (b, r') is part of τ, we have that $b - r' \geq \text{FF}(R, B)$. If r' is a local maximum, then Red can pass r' and get to the next vertex (a local minimum) while only increasing the separation from Blue. □

Fig. 3 Edges of the simplified curves are dotted. The subpath of B between two consecutive vertices of B' lies fully between the vertices

After signifying the importance of local minima and maxima of R and B, we look at global extrema. Let $b_{\max}$ be the rightmost point of B and let $r_{\min}$ be the leftmost point of R.

Lemma 6 *If $FF(B, R) > 0$, then:*

(i) All of R must be to the left of $b_{\max} - FF(B, R)$, and all of B must be to the right of $r_{\min} + FF(B, R)$.
(ii) There exists an optimal traversal (achieving $FF(B, R)$) such that at some point Blue is at $b_{\max}$ and Red is at $r_{\min}$.

Proof (i) holds by continuity: if R has a point to the right of $b_{\max} - FF(B, R)$, there must be a time when Red is at distance smaller than $FF(B, R)$. The claim for B is symmetric. For (ii), we use the same ideas as in the proof of Lemma 4. Consider a traversal, and assume that Blue reaches $b_{\max}$ before Red reached $r_{\min}$. Let r' be the location of Red when Blue is at $b_{\max}$; r' does not have to be a vertex of R, but in any case r' precedes $r_{\min}$ along R. Let $b' \in B$ be Blue's location at the time Red reaches $r_{\min}$ (b' is after $b_{\max}$). By (i), while Blue is at $b_{\max}$, Red can go from r' to $r_{\min}$ – so $(b_{\max}, r_{\min})$ becomes part of the traversal. Again by (i), while Red is at $r_{\min}$, Blue can go from $b_{\max}$ to b'. From $(b', r_{\min})$, Blue and Red can follow τ to complete the traversal. □

Lemma 6 allows us to assume, without loss of generality, that $b_{\max}$ and $r_{\min}$ are the final points of B and R, respectively (i.e., $B[n] = b_{\max}$, $R[n] = r_{\min}$): for arbitrary B, R we can separately solve the problem for the subpaths of Blue from $B[1]$ to $b_{\max}$ and Red from $R[n]$ to $r_{\min}$, and the problem for the (reversed) subpaths of Blue from $B[n]$ to $b_{\max}$ and Red from $R[n]$ to $r_{\min}$.

Our final simplification of R and B is inspired by the linear-time algorithm for computing the weak Fréchet distance in 1D [6, Section 8.2]. Given a path P on the x-axis, consider the following algorithm that constructs a simplified path P' which is a subsequence of P (refer to Fig. 3). First, for $i = 1$ to n, add $P[i]$ to P' if either $P[i] \geq \max_{1 \leq j \leq i-1} P[j]$ or $P[i] \leq \min_{1 \leq j \leq i-1} P[j]$. Intuitively, this process creates a 'zig-zag' path P', with monotonically growing extreme points. Then, remove from P' any vertex $P'[j]$ such that either $P'[j-1] \leq P'[j] \leq P'[j+1]$ or $P'[j-1] \geq P'[j] \geq P'[j+1]$. This removes from P any subdivided edges. Let R', B' be such simplified path computed for R, B, respectively. Since $B[n] = b_{\max}$, $R[n] = r_{\min}$, we get that b_{max} is the final point of B' and r_{min} is the final point of R'.

Lemma 7 $FF(B, R) = FF(B', R')$.

Proof Consider an optimal traversal τ' of R', B', in which at any time (at least) one of the agents is on a vertex (Lemma 4). Suppose that while Red remains on a vertex

$r' \in R'$, Blue traverses an edge $B'[j, j+1]$ of the simplified curve B'. Then both $B'[j] - r' \geq \mathrm{FF}(B', R')$ and $B'[j+1] - r' \geq \mathrm{FF}(B', R')$. Since the subpath of B from $B'[j]$ to $B'[j+1]$ lies between the endpoints of $B'[j, j+1]$, we get that $b - r' \geq \mathrm{FF}(B', R')$ for any $b \in B$ between $B'[j]$ and $B'[j+1]$. We can add the subpath to τ'; after adding all such subpaths (both for B and for R) we obtain a traversal of R, B, achieving $\mathrm{FF}(B', R')$,

Conversely, consider a traversal τ of R, B, in which at any time (at least) one of the agents is on a local extremum (Lemma 5). We modify τ so that the agents pause only at maxima of B' and minima of R'. For this, whenever Blue (resp. Red) pauses, modify the traversal to pause at the previous vertex of B' (resp. R') instead: (as in the proofs of the lemmas above,) this vertex gives more freedom for Red (resp. Blue) to move; moreover, the next time Red pauses will be a global minimum of its prefix, so Blue will not have lost any freedom by having paused early. □

Finally, when the curves R, B have been simplified, the flipped Fréchet distance between them may be found by the greedy algorithm which always selects the "cheapest" possible next move, i.e., either moves Red or Blue depending on what leads to a larger distance between the vertices. The full algorithm is as follows:

- if $R[i+1] \geq B[j]$ and $B[j+1] \leq R[i]$, return 0.
- else, if $R[i+1] \geq B[j]$ and $B[j+1] > R[i]$, set $j \leftarrow j+1$ and $\delta \leftarrow \min\{\delta, B[j+1] - R[i]\}$.
- else, if $R[i+1] < B[j]$ and $B[j+1] \leq R[i]$, set $i \leftarrow i+1$ and $\delta \leftarrow \min\{\delta, B[j] - R[i+1]\}$.
- else, if $R[i+1] < B[j]$ and $B[j+1] > R[i]$:
  - if $B[j] - R[i+1] \geq B[j+1] - R[i]$, set $i \leftarrow i+1$ and $\delta \leftarrow \min\{\delta, B[j] - R[i+1]\}$,
  - if $B[j] - R[i+1] < B[j+1] - R[i]$, set $j \leftarrow j+1$ and $\delta \leftarrow \min\{\delta, B[j+1] - R[i]\}$.

Finally, if $i = n$ we return $\min\{\delta, \min_{j \leq k \leq n}(B[k] - R[i])\}$, and if $j = n$ we return $\min\{\delta, \min_{i \leq k \leq n}(B[k] - R[j])\}$.

Claim 8 *The algorithm above returns FF(B, R) (for simplified paths R, B).*

Proof First notice that if $R[i]$ is a local minimum of R, and $B[j]$ is a local minimum of B, then the next step of the algorithm is Blue moving to a local maximum $B[j+1]$. Similarly, if $B[j]$ is a local maximum of B, and $R[i]$ is a local maximum of R, then the next step of the algorithm is Red moving to a local minimum $R[i+1]$. In both cases, δ does not change. In addition, if $R[i]$ is a local maximum of R and $B[j]$ is a local minimum of B, then in the next step of the algorithm either Red or Blue will "step back" to a local minimum or local maximum, respectively, and again δ will not change.

In fact, δ changes only at a step when Red is on a local minimum and Blue is on a local maximum. In this case, the algorithm chooses the next move that decreases δ as little as possible (if at all). Since one of the agents needs to move, the decrease is necessary.

□

Since the simplification of the curves and the greedy algorithm can be performed in linear time, we obtain our main result of this section:

Theorem 9 *Given two polygonal curves P, Q with n vertices in 1D, their social distance width, $FF(P, Q)$, can be computed in $O(n)$ time.*

2.2 Quadratic-Time Algorithms

In this section we describe algorithms for computing continuous and discrete $\mathrm{FF}(R, B)$; the algorithms are based on similar algorithms for computing the continuous and discrete Fréchet distances, in times $O(n^2 \log n)$ and $O(n^2)$, respectively. As with Fréchet distance, our algorithms use the notion of a free space diagram, appropriately adapted.

The free space diagram. The *δ-free space diagram* [1] of two curves P and Q represents all locations on P and Q with distance at most δ. We adapt this notion to our new setting.

Let $\mathcal{C}_{ij} = [i, i+1] \times [j, j+1]$ be a unit square in the plane, for integers $1 \le i \le n-1$ and $1 \le j \le m-1$. Let $\mathcal{B} = [1, n] \times [1, m]$ be the square in the plane that is the union of the squares C_{ij}. Given $\delta > 0$, the *δ-free space* is $\mathcal{F}_\delta = \{(p, q) \in \mathcal{B} \mid \|P(p) - Q(q)\| \ge \delta\}$. In other words, it is the set of all red-blue positions for which the distance between the agents is at least δ. A point $(p, q) \in \mathcal{F}_\delta$ is a *free point*, and the set of *non-free points* (or *forbidden points*) is then $\mathcal{B} \setminus \mathcal{F}_\delta$. Note that for Fréchet distance, these definitions are reversed ("flipped"). We call the squares $\mathcal{C}_{ij}$ the *cells* of the free space diagram; each cell may contain both free and forbidden points. An important property of the free space diagram is that the set of forbidden points inside a cell $\mathcal{C}_{ij}$ (i.e., $\mathcal{C}_{ij} \cap \mathcal{F}_\delta$) is convex [1].

Notice that a monotone path through the free space $\mathcal{F}_\delta$ between two free points (p, q) and (p', q') corresponds to a traversal of $P[p, p']$ and $Q[q, q']$. Thus, $\mathrm{FF}(P, Q) \ge \delta$ if and only if there exists a monotone path through the free space $\mathcal{F}_\delta$ between $(0, 0)$ and (n, n) (i.e. (n, n) is "reachable" from $(0, 0)$). The Fréchet distance between P and Q can be computed in $O(n^2 \log n)$ time [1] as follows. For a given value of δ, the reachability diagram is defined to be the set of points in $\mathcal{F}_\delta$ reachable from $(0, 0)$. As the set of free points in each cell is convex, the set of "reachable" points on each of the boundary edges of a cell is a line segment. Thus, one can construct in constant time the reachable boundary points of a cell, given the reachable boundary points of its bottom and left neighbor cells (see Fig. 4). Therefore, computing the reachability diagram (and hence solving the decision version of the problem) takes $O(n^2)$ time using a dynamic programming algorithm. For the optimization, there are $O(n^3)$ critical values of δ, which are defined by (i) the distances between starting points and endpoints of the curves, (ii) the distances between vertices of one curve and edges of the other, and (iii) the common distance of two vertices of one curve to the intersection point of the bisector with some edge of the other. Then, parametric search, based on sorting, can be performed in time $O((n^2 + T_{dec}) \log n)$, where T_{dec} is the running time for the decision algorithm.

In the case of FF, we can again compute the reachability diagram in $O(n^2)$ time, as in each cell the set of forbidden points is convex, and thus the set of "reachable" points

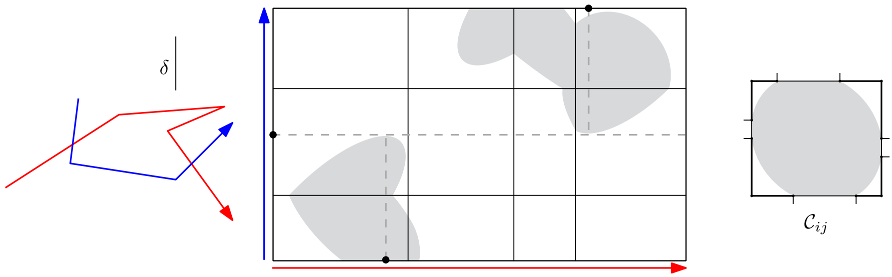


Fig. 4 Right: a free space cell $\mathcal{C}_{ij}$. For FF, the free space is white, while for Fréchet distance, the free-space is gray. The gray region within a cell is convex. Left: the free space diagram of two curves. The black points and dashed lines indicate critical values of type (iii), which are openings in the free space diagram defined by two red edges and one blue edge

on each of the boundary edges of a cell is at most two line segments; moreover, in each cell the set X of infeasible points has constant description complexity, implying that reachable points on the sides of a cell can be computed in constant time – these are the points that can be reached from feasible points in the adjacent cells by x- and y-monotone paths that avoid X. The set of critical values is similar, except that the third type can occur between three edges (see Fig. 4, left). Thus, by arguments similar to [1], we have a $O(n^2 \log n)$ time algorithm for computing FF(P, Q).

Theorem 10 *There is an $O(dn^2 \log n)$ time exact algorithm for computing the Flipped Fréchet measure of two polygonal n-vertex curves in $\mathbb{R}^d$.*

For the discrete version of FF, a simple dynamic programming algorithm (similar to the one known for discrete Fréchet distance) gives an $O(n^2)$ solution. In short, let $OPT[i, j]$ be the FF of $P[1, i]$ and $Q[1, j]$; then, by the definition of dFF we have $OPT[i, j] = \min\{\|P[i] - Q[j]\|, \max\{OPT[i-1, j], OPT[i, j-1], OPT[i-1, j-1]\}\}$.

Theorem 11 *There exists an $O(dn^2)$ time exact algorithm for computing the dFF measure of two polygonal n-vertex curves in $\mathbb{R}^d$.*

2.3 Quadratic Lower Bounds

Bringmann and Mulzer [5] give a lower bound (conditioned on SETH) for computing the discrete Fréchet distance between two curves in 1D, using a reduction from the Orthogonal Vectors (OV) problem. We prove similar results below for FF in 2D and dFF in 1D.

The Orthogonal Vectors problem is defined as follows. Given two sets $U = \{u_1 \dots u_N\}$ and $V = \{v_1 \dots v_N\}$, each consisting of N vectors in $\{0, 1\}^D$, decide whether there are $u_i \in U$, $v_j \in V$ orthogonal to each other, i.e., $u_i(k) \cdot v_j(k) = 0$ for every $k = 1, \dots, D$ (where $u_i(k)$ denotes the kth coordinate of u_i). Bringmann and Mulzer [5] showed that if OV has an algorithm with running time $D^{O(1)} \cdot N^{2-\varepsilon}$ for some $\varepsilon > 0$, then SETH fails.

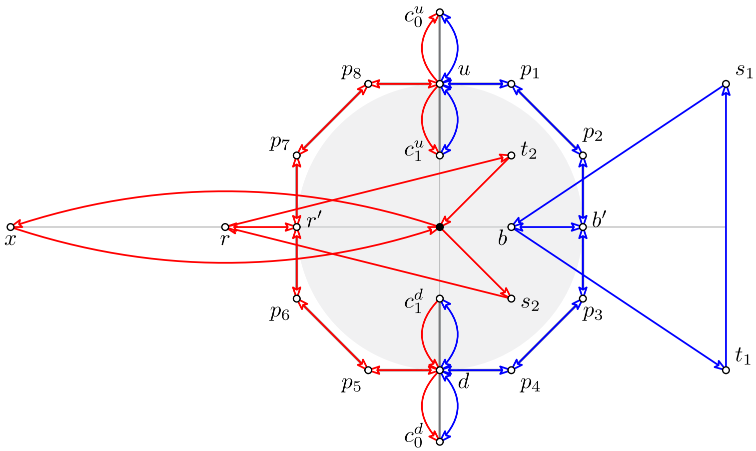


Fig. 5 The curves R and B for the continuous FF lower bound

In the following, by an algorithm with approximation factor $\alpha < 1$ we mean an algorithm that outputs a traversal whose maintained separation distance is *at least* α times the FF, given by an optimal traversal.

In Theorem 12 we show that for continuous FF, increasing the dimension from 1D (where we gave a linear-time algorithm) to 2D likely rules out subquadratic algorithms. Then, in Theorem 13 we show a quadratic lower bound for the discrete version in 1D.

Theorem 12 *(FF Lower Bound in 2D) There is no algorithm that computes the FF measure between two polygonal curves of length n in the plane, up to an approximation factor at least* $\frac{\sqrt{5}}{2\sqrt{2}}$*, and runs in time* $O(n^{2-\delta})$ *time for any* $\delta > 0$*, unless OV fails.*

Proof Set $\alpha = \frac{\sqrt{5}}{2\sqrt{2}} \approx 0.79$. Given an instance $U = \{u_1 \dots u_N\}$, $V = \{v_1 \dots v_N\}$ of OV, we show how to construct two curves R and B of length $O(N \cdot D)$ in the plane, such that if $U \times V$ contains an orthogonal pair then $\mathrm{FF}(R, B) \geq 1$, and if $U \times V$ does not contain an orthogonal pair then $\mathrm{FF}(R, B) \leq \alpha$. Therefore, if there exists an algorithm with running time $O(n^{2-\varepsilon})$ for computing FF of two curves of length n in the plane, then $\mathrm{FF}(R, B)$ can be computed in $O((DN)^{2-\varepsilon})$, which means that OV can be decided in $D^{O(1)} \cdot N^{2-\varepsilon}$ time, and SETH fails. Moreover, if FF can be approximated up to a factor of α, then again we get that OV can be decided in $D^{O(1)} \cdot N^{2-\varepsilon}$ time, and SETH fails. As in [5], we assume that D is even (otherwise add a 0 coordinate to each vector).

The construction of R (resp. B) is such that for each vector $u_i \in U$ (resp. $v_j \in V$), we construct a vector gadget curve A_i (resp. B_j), such that if u_i and v_j are orthogonal then $\mathrm{FF}(A_i, B_j) = 1$, and otherwise $\mathrm{FF}(A_i, B_j) \leq \alpha$. Then, we connect the vector gadgets into curves R and B.

Consider the following set of points (see Fig. 5):

- Points on the x-axis: $x = (-1.5, 0)$, $r = (-0.75, 0)$, $r' = (-0.5, 0)$, $b = (0.25, 0)$, $b' = (0.5, 0)$

- Points on the y-axis: $c_0^u = -c_0^d = (0, 0.75)$, $c_1^u = -c_1^d = (0, 0.25)$, $u = -d = (0, 0.5)$
- Octagon points: $p_1 = -p_5 = (0.25, 0.5)$, $p_2 = -p_6 = (0.5, 0.25)$, $p_3 = -p_7 = (0.5, -0.25)$, $p_4 = -p_8 = (0.25, -0.5)$
- Other points: $s_1 = (1, 0.5)$, $t_1 = (1, -0.5)$, $s_2 = (-0.25, -0.25)$, $t_2 = (0.25, 0.25)$

Notice that all the points are located on a regular grid with side length 0.25, and that $\|s_1 - t_2\| = \|t_1 - s_2\| = \|c_1^u - p_4\| = \|c_1^u - p_5\| = \|c_1^d - p_1\| = \|c_1^d - p_8\| = \|r - c_1^u\| = \|r - c_1^d\| = \alpha$.

For each $u_i \in U$, the gadget A_i is constructed as follows:

$$r' \bigcirc_{k=1,\dots,\frac{D}{2}} \Big(p_7 \circ p_8 \circ u \circ c^u_{u_i(2k-1)} \circ u \circ p_8 \circ p_7 \\ \circ p_6 \circ p_5 \circ d \circ c^d_{u_i(2k)} \circ d \circ p_5 \circ p_6\Big) \circ r' \ .$$

Similarly, for $v_j \in V$, the gadget B_i is constructed as follows:

$$b' \bigcirc_{k=1,\dots,\frac{D}{2}} \Big(p_3 \circ p_4 \circ d \circ c^d_{v_i(2k-1)} \circ d \circ p_4 \circ p_3 \circ p_2 \\ \circ p_1 \circ u \circ c^u_{v_i(2k)} \circ u \circ p_1 \circ p_2\Big) \circ b' \ .$$

It is easy to see that if u_i, v_j are orthogonal then $\mathrm{FF}(A_i, B_j) = 1$, because the traversal that uses "antipodal" points maintains distance 1 between Red and Blue. For the other direction, we claim that if $\mathrm{FF}(A_i, B_j) > \alpha$, then u_i, v_j are orthogonal. Now assume that $u_i(1) = 1$, so R starts with r', p_7, p_8, u, c_1^u. Notice that when Blue traverses the subcurve b', p_3, p_4, d, Red cannot reach c_1^u. If $v_j(1) = 1$, then the next move of Blue is toward c_1^d, and the distance between the agents becomes at most α (if Red in on p_8 while Blue is on c_1^d then their distance is exactly α). This means that $u_i(1) = v_j(1) = 1$ is not possible. Therefore, at least one of $u_i(1), v_j(1)$ is 0. Notice that Blue will visit d for the first time before Red visits d for the first time. Similarly, Red will visit u for the first time before Blue visits u for the first time. Moreover, Red cannot reach r' before Blue leaves d, and Blue cannot reach b' before Red leaves u. Thus, the movement of Red and Blue is synchronized in the sense that when Red is moving from u towards d, Blue is moving from d towards u, and vice versa. Hence, by similar (symmetric) arguments, we get that $u_i(k) = v_j(k) = 1$ is not possible for all $k = 2, \dots, D$ as well, and u_i, v_j are orthogonal.

The gadgets A_i and B_j are connected into R and B as follows:

$$R = x \circ (0,0) \circ s_2 \bigcirc_{i=1,\dots,N} (r \circ A_i) \circ r \circ t_2 \circ (0,0) \circ x \ , \\ B = \bigcirc_{j=1,\dots,N} \big(s_1 \circ b \circ B_j \circ b \circ t_1\big) \ .$$

If $u_i \in U, v_j \in V$ are orthogonal, then Blue traverses $\bigcirc_{k=1,\dots,j-1} (s_1 \circ b \circ B_k \circ b \circ t_1)$ while Red stays at x, then Blue moves to s_1. Now Red moves from x to $(0, 0)$, to s_2, then traverses $\bigcirc_{k=1,\dots,i-1} (r \circ A_k)$, and moves to

r just before A_i. Now Blue moves to b just before B_j, and they traverse A_i and B_j in sync, keeping distance ≥ 1. Now Red moves to r while Blue moves to b, then Blue moves to t_1. While Blue is on t_1, Red traverses $\bigcirc_{k=i+1,\ldots,N}(r \circ A_k) \circ r' \circ t_2 \circ x$. Finally, Blue traverses $\bigcirc_{k=j+1,\ldots,N}(s_1 \circ b \circ B_k \circ b \circ t_1)$ while Red is on x.

For the converse, assume that $\mathrm{FF}(R, B) > \alpha$. When Red reaches $(0, 0)$ for the first time, Blue must be on s_1 or t_1 (or on the edge between them). If Blue is on t_1, then it must move towards s_1 before Red can continue to s_2; moreover, when Red reaches s_2 Blue can only be strictly above the x-axis either on the edge (t_1, s_1) or (s_1, b). Thus we can assume that when Red is on s_2, Blue is on s_1 immediately before some vector gadget B_j. Now consider the first time when Red reaches t_2. At this time, Blue can be either near t_1 (strictly below the x-axis) or near c_0^d. However, if Blue is near c_0^d, it is not possible for the agents to continue their movement: Blue cannot reach d and Red cannot reach $(0, 0)$. Thus we can assume that when Red reaches t_2, Blue is near t_1 and strictly below the x axis. We conclude that between the first time that Red visited s_2 and the first time that Red visited t_2, Blue had to traverse the vector gadget B_j in order to get from a point strictly above the x axis to a point strictly below it. Before Blue starts traversing B_j, it first visits b, but when Blue is on b, the only possible location of Red is near r (more precisely, on the edge between r and r', but not on r'). Notice that Red cannot be near x since it has not visited t_2 yet. Now there are two options: (1) Red is immediately before some vector gadget A_i, and thus by previous arguments u_i and v_j are orthogonal, or (2) Red has finished all its vector gadgets, and it waits on r while Blue traverses B_j. In this case, notice that since $\|r - c_1^u\| = \|r - c_1^d\| = \alpha$, Blue cannot reach any of c_1^u, c_1^d while Red is on r; thus, v_j must be a 0-vector, implying u_i and v_j are orthogonal. □

We now show that the discrete FF likely requires quadratic time, even in 1D.

Theorem 13 *(Discrete 1D Lower Bound) There is no $O(n^{2-\varepsilon})$ time α-approximation algorithm for dFF in 1D, for any $\varepsilon > 0$ and $\alpha > 2/3$, unless OV fails.*

Proof Given an instance $U = \{u_1 \ldots u_N\}$, $V = \{v_1 \ldots v_N\}$ of OV, we show how to construct two curves R and B of length $O(N \cdot D)$ on the line, such that if $U \times V$ contains an orthogonal pair then $\mathrm{dFF}(R, B) \geq 1$, and if $U \times V$ does not contain an orthogonal pair then $\mathrm{dFF}(R, B) \leq 2/3$. Therefore, if there exists an algorithm with running time $O(n^{2-\varepsilon})$ for computing dFF of two curves of length n on the real line, then $\mathrm{dFF}(R, B)$ can be computed in $O((DN)^{2-\varepsilon})$, which means that OV can be decided in $D^{O(1)} \cdot N^{2-\varepsilon}$ time, and SETH fails. Moreover, if dFF can be approximated up to a factor of 2/3, then again we get that OV can be decided in $D^{O(1)} \cdot N^{2-\varepsilon}$ time, and SETH fails. Again as in [5] we assume that D is even.

Consider the following set of points on the line (see Fig. 6): $w_1 = -w_2 = 5/3$, $x_2 = -x_1 = 1$, $a_0^e = b_0^o = -a_0^o = -b_0^e = 2/3$, $a_1^e = b_1^o = -a_1^0 = -b_1^e = 1/3$, $s = 0$.

We first construct vector gadgets. For each $u_i \in U$, we create a subsequence A_i of R: for odd (resp. even) k, the kth point in A_i is $a^o_{u_i(k)}$ (resp. $a^e_{u_i(k)}$). Similarly, for $v_j \in V$, we create a subsequence B_j of B, using bs instead of as. It is easy to see that Red and Blue can traverse B_j and A_i while maintaining distance 1 if and only if u_i, v_j are orthogonal (they jump between odd and even points in sync, "opposite"

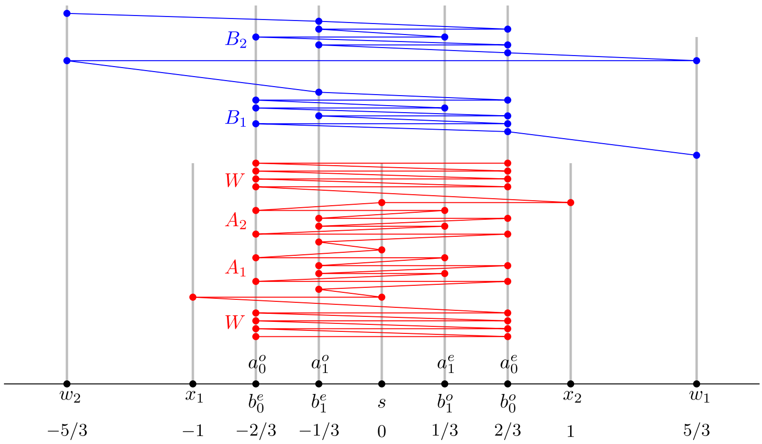


Fig. 6 For a pair of orthogonal vectors, Red and Blue jump in sync between as and bs, with at least one of them at a "far" point (indexed with 0)

each other, and at least one of them is at "far" point, indexed with 0). Note that any such vector gadget has length D.

The gadgets B_j are connected into B as follows: $B = w_1 \circ B_1 \circ w_2 \circ w_1 \circ B_2 \circ w_2 \circ \cdots \circ w_1 \circ B_N \circ w_2$. Let W be the sequence of $D(N-1)$ points that alternate between a_0^o and a_0^e starting with a_0^o (Red can traverse each of $N-1$ length-D subpaths of W in sync with Blue on any B_j). We construct R as follows: $R = W \circ x_1 \circ s \circ A_1 \circ s \circ A_2 \circ \cdots \circ s \circ A_N \circ s \circ x_2 \circ W$.

The proof that OV is a Yes instance if and only if $\mathrm{dFF}(R, B) \geq 1$ is similar to that in [5], with an important change: here, we also show that if OV is a No instance, then $\mathrm{dFF}(R, B) \leq 2/3$. Indeed, notice that since $\mathrm{dFF}(R, B)$ is determined by the distance between two vertices, one from R and one from B, we get that if $\mathrm{dFF}(R, B) > 2/3$, then necessarily $\mathrm{dFF}(R, B) \geq 1$ (as there are no two points in our construction with distance in the range $(2/3, 1)$). Therefore, it is enough to show that if OV is a No instance, then $\mathrm{dFF}(R, B) < 1$.

If $u_i \in U$, $v_j \in V$ are orthogonal, then Red traverses $D(N-j)$ points on W while Blue stays at w_1, then Blue traverses $B_1 \dots B_{j-1}$ in sync with Red traversing the rest of W. Now, while Blue stays at w_1 before B_j, Red goes to x_1 and traverses $A_1 \dots A_{i-1}$, then goes to s before A_i. Then, A_i, B_j are traversed in sync, Blue stays at w_2 while Red completes the traversal of $A_{i+1}, \dots, A_N$ and goes to x_2, and finally Blue can complete the traversal of $B_{j+1}, \dots, B_N$ in sync with Red traversing the second W gadget. When Blue goes to w_2, Red is able to complete the traversal of W.

For the converse, assume that $\mathrm{dFF}(R, B) \geq 1$. When Red is on x_1, Blue must be to the right of s, but if Blue is not on w_1, then they cannot take the next step – so Blue must be on w_1, right before some vector gadget B_j. Immediately after leaving w_1, Blue gets to b_0^o or b_1^o, implying that Red must be at a_0^o or a_1^o, i.e., either in some vector gadget A_i or on the second W (since it already passed x_1). However, if it is on W, it must have gone through x_2 which is too close to w_1, so Red is in A_i. Now while they are on A_i and B_j, Red and Blue must jump in sync, until one of them reaches the end point of their respective vector gadget. Therefore, if Red did not start on the first point of A_i, then it will finish A_i and appear at s before Blue has finished B_j. This means that A_i and B_j were traversed simultaneously, and since $\mathrm{dFF}(R, B) \geq 1$, the respective vectors have to be orthogonal. □

2.4 More than 2 Agents

Our algorithm for continuous FF in 1D (more specifically, the decision version: deciding whether $\mathrm{FF}(R, B) \geq \delta$) generalizes to any number $k \geq 2$ of agents, with a running time of $O(kn)$, as follows. Let $A_1, A_2, \ldots, A_k$ be the polygonal paths of k agents in 1D. Given a distance value δ, our goal is to decide whether the k agents can traverse their respective paths while maintaining distance δ from one another. Since we are on a line and agents cannot cross paths, we only need to maintain distances for neighboring (along the line) agents. Therefore, we can translate each A_i by $-(i-1)\delta$ (for $2 \leq i \leq k$), and then our goal is to find non-crossing traversals, or corresponding paths on the space-time graph. This can be done by fixing a path for A_1, then using our algorithm for two agents to align a corresponding path for A_2, then consider the path of A_2 as fixed, and align a path for A_3 and so on (recall that we can have an agent walk in infinite speed).

The question of whether or not there exists an algorithm in 2D with running time fully polynomial in k remains open (for the Fréchet distance of a set of curves, the best known running time is roughly $O(n^k)$; see [11]).

3 Social Distancing in a Simple Polygon

In this section we consider distancing problems in which the given domain (for both Red and Blue) is a simple polygon. Since the two agents are moving inside the same polygon, it is natural to consider geodesic distance (i.e., the shortest path inside the polygon) instead of Euclidean distance to measure separation.

Consider a scenario in which Red and Blue have to traverse two polygonal paths R and B, both inside a given polygon P, and their goal is to find a movement strategy (a traversal) that maintains geodesic distance of at least δ between them. For the analogous Fréchet problem (Red and Blue have to maintain geodesic distance of at most δ), Cook and Wenk [9] presented an algorithm that runs in $O(n^2 \log N)$ time, where N is the complexity of P and n is the complexity of R and B. Their algorithm is based on the fact that the free space in a cell of the diagram is x-monotone, y-monotone, and connected. Then the geodesic decision problem can be solved by propagating the reachability information through a cell in constant time, as for the Euclidean Fréchet distance. Thus, we can apply a similar “flipped” algorithm for computing the $\mathrm{FF}(R, B)$ under geodesic distance in nearly quadratic time.

When both Red and Blue are restricted to traverse a given path, it seems that the Fréchet-like nature of the problem leads to near-quadratic time algorithms. Thus, in this section we consider the scenario where Blue has more freedom, and it is not required to traverse a given path. We first consider the case when Red is walking on an arbitrary path in the polygon; we show that while the naive solution takes at least cubic time, there exists a quadratic time algorithm for this problem. We then describe *two variants for which we present linear time algorithms*; the first is where Red is walking along a shortest path in the polygon, and second is where Red is traversing the boundary of a skinny polygon (a tree).

Throughout this section, we will use SDW to denote social distancing width, to differentiate from the Flipped Fréchet versions where both the Red and Blue curves are given as input.

3.1 Red on an Arbitrary Path Mission

Consider the case when Red moves along some given path R in P, and Blue may wander around in P, starting from some given point b. The free-space diagram can be adapted to the case of a path and a polygon, by partitioning the polygon into a linear number of convex cells (for example, a triangulation). This is a three-dimensional structure, which contains $O(n^2)$ cells (assuming that the complexity of both R and P is $O(n)$). However, for maintaining geodesic separation, building the free-space may be cumbersome, because it would involve building the parametric shortest path map in a triangle as the source moves along a segment, and such SPM may have $\Omega(n)$ combinatorial changes. Instead, we show that Blue may stay on the boundary, thus reducing the problem to the standard free-space diagram between a path and a closed curve. We will prove:

Theorem 14 *Let P be a polygon with n vertices, b a point in P, and R a path between two points r and r' in P. There exists an $O(n^2 \log n)$-time algorithm to decide whether there exists a path B in P starting from b, such that $SDW(R, B) > 1$ under geodesic distance.*

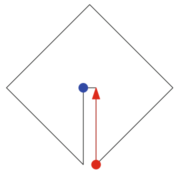
The proof follows from the following observation and lemma, and by using the $O(n^2 \log n)$-time algorithm of [9] for geodesic Fréchet distance between two curves.

Observation 15 *Let p be an arbitrary point inside a polygon P, and D be the closed geodesic disk of any radius centered at p, then D splits both P and ∂P into the same number of connected components, and there is a natural one-to-one correspondence between them.*

Lemma 16 *Assume that the point b is on the boundary. If there exists a path B in P starting from b, such that $SDW(R, B) > 1$ under geodesic distance, then there exists such a path B' that is entirely on the boundary of P.*

Proof Let r_1 be the first point of R, and denote by $C_1, C_2, \ldots, C_k$ the set of connected components of $\partial P \backslash D_{r_1}$ where D_{r_1} is the unit geodesic disk around r_1, i.e., the set of points within geodesic distance 1 from r_1. Assume that b lies in C_1. When Red moves continuously along the segments of R, some of the connected components may disappear, split, or merge with other connected components. In addition, some new connected components may appear. These events may occur in the case in which, for some point p on R, either (i) a segment of ∂P becomes tangent to D_p, or (ii) a vertex of P is on the boundary of D_p. By Observation 15, as long as none of these events occur, Blue can stay on the boundary of C_1, because, as Red moves, C_1 contains a single connected piece of ∂P on which Blue can walk (Blue may need to move along C_1 in the case that C_1 "shrinks"). Consider the first time when Red reaches a point r_2 on R such that either (i) or (ii) occur. If this event causes C_1 to disappear, then Blue had no way to escape. If C_1 splits, then by Observation 15, Blue can move on the boundary of C_1 to any of the new connected components, right before the split occurs. If C_1 merges with another connected component, say C_2, then again by Observation 15, Blue can move to the boundary of C_2 via the boundary of C_1. In any of these cases, Blue can move via ∂P to the connected component in which it would be if it had walked along B. □

Remark. Observation 15 and hence the above lemma apply only to geodesic distance separation – if Blue is maintaining separation using Euclidean distance, Blue may need to go inside the polygon; see the figure on the right for an example. In this case, we build the 3D free-space, but note that its complexity is quadratic, because the complexity of each free-space cell that corresponds to a triangle of P and a red edge is constant, and thus the free-space can be searched in quadratic time.



3.2 Red on a Shortest Path Mission

Assume that Red moves along a geodesic path R in P (Red is on a mission and does not care about social distancing) while Blue may wander around anywhere within P starting from a given point b. We show that the decision problem, whether Blue can maintain (geodesic) social distance at least 1 from Red, can be solved in linear time.

Theorem 17 *Let P be a polygon with n vertices, b a point in P, and R a geodesic shortest path between two points r and r' in P. There exists an $O(n)$-time algorithm to decide whether there exists a path B in P starting from b, such that $SDW(R, B) > 1$ under geodesic distance.*

Proof We use $|ab|$ to denote the geodesic distance between points $a, b \in P$. For a point $t \in \mathbb{R}$ let $D_t = \{p \in P : |tp| \leq 1\}$ be the unit geodesic disk centered on t; let $M = \cup_{t \in R} D_t$ be the set of points within geodesic distance 1 from R (Fig. 7, left). Without loss of generality, assume $b \notin D_r$ (otherwise separation fails from the start). The disk D_r splits P into connected components (a component is a maximal connected subset of $P \setminus D_r$): Blue can freely move inside a component without intersecting D_r; in particular, if the component $P' \ni b$ of b is not equal to $M \setminus D_r$ (i.e., if $P' \setminus M \neq \emptyset$), then Blue can move to a point in $P' \setminus M$ (a safe point) and maintain the social distance of 1 from Red (existence of a safe point can be determined by tracing the boundary of M). Next, we show that the existence of such a safe point is also *necessary* for Blue to maintain the distance of 1 from the geodesic shortest path R.

Indeed, as Red follows R, D_t sweeps M; let $S_t \subseteq M$ be the points swept (at least once) by the time Red is at $t \in R$ and let $U_t = M \setminus S_t$ be the unswept points. Since

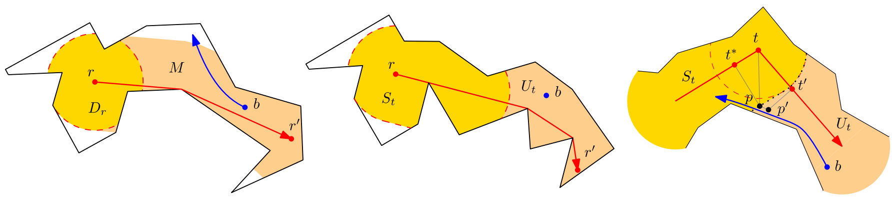


Fig. 7 Left: D_r splits P into connected components. Blue escaping Red as there exists a safe point. Middle: the boundary between S_t and U_t is dashed. Blue cannot escape because its connected component equals $M \setminus D_r$. Right: Blue escapes when Red is at t

$b \notin D_r = S_r$, initially Blue is in the unswept region. Assume that there is no safe point ($P' = M \setminus D_r$) and yet Blue can escape. Suppose Blue can escape from the unswept to swept when Red is at $t \in R$ (Fig. 7, right). Then there exists a point p on the boundary between U_t and S_t that is further than 1 from t, say $|pt| = 1 + \varepsilon$ for some $\varepsilon > 0$. Since p is on the boundary of S_t, at some position $t^* \in \pi$ before t, we had $|t^* p| = 1$. Since p is on the boundary of U_t, there exists an unswept point $p' \in U_t$ within distance less than ε from p: $|p'p| < \varepsilon$. Finally, since $U_t = M \setminus S_t$ is part of M, p' becomes swept when Red is at some point $t' \in \pi$ after t: $|t'p'| \le 1$. We obtain that there are three points t^*, t, t' along a geodesic path π and a point p such that $|t^* p| \le 1 < 1+\varepsilon = |tp|$ and $|t'p| \le |t'p'| + |p'p| < 1+\varepsilon = |tp|$, contradicting the fact that the geodesic distance from a point to a geodesic path is a convex function of the point on the path [19, Lemma 1] (this is the place where we use that R is a geodesic path: if R is not geodesic, it is not necessary for the Blue to escape from M while Red is at r).

We now show how to implement our solution to the decision problem in $O(n)$ time. To build the geodesic unit disk D_r we compute the *shortest path map* (SPM) from r (the decomposition of P into cells such that for any point p inside a cell the shortest r-p path has the same vertex v of P as the last vertex before p) – the SPM can be built in linear time [15]; then in every cell of the SPM we determine the points of D_r: any cell is either fully inside D_r, or fully outside, or the boundary of the disk in the cell is an arc of the radius-$(1 - |rv|)$ circle centered on the vertex v of P. The set M can be constructed similarly, using SPM from R. To build the SPM, we decompose P by drawing perpendiculars to the edges of R at every vertex of the path (see Fig. 8): in any cell of the decomposition, the map can be built separately because the same *feature* (a feature is a vertex or a side of an edge) of R will be closest to points in the cell (the decomposition is essentially the Voronoi diagram of the features). In every cell, the SPM from the feature can be built in time proportional to the complexity of the cell (the linear-time funnel algorithm for SPM [15] works to build the SPM from a segment too: the algorithm actually propagates shortest path information from segments in the polygon). Since the total complexity of all cells is linear, the SPM is built in overall linear time. After D_r and M are built, we test whether $b \in D_r$ (if yes, the answer is No) and trace the boundary of M to determine the existence of a safe point (the answer is Yes iff such a point exists). □

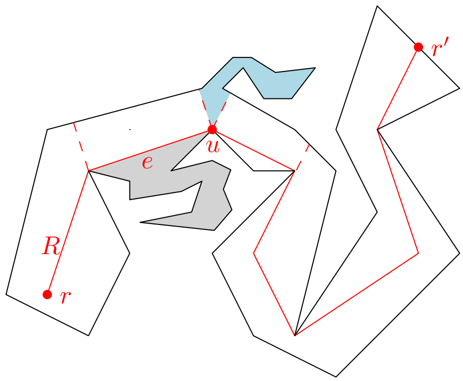


Fig. 8 SPMs from edges and vertices of π are computed separately in portions of P defined by perpendiculars to path edges (some shown dashed). Gray and light blue parts are charged to (the right side of) the edge e and to vertex u of R, resp

3.3 SDW of Closed Curves and Polygons

Consider a scenario in which the polygonal curves R and B are closed curves. Here, the starting points of Red and Blue are not given as an input, and the goal is to decide whether they can traverse their respective curves while maintaining distance at least δ. The analogous Fréchet problem has been investigated by Alt and Godau [1], who presented an $O(n^2 \log^2 n)$ time algorithm, and later in [22], where the running time was improved to $O(n^2 \log n)$. Those algorithms include the construction of dynamic data structures for the free space diagram, which is again based on the fact that the free space within a cell is convex. Since, in our "flipped" case, the forbidden space is convex, similar data structures can be used in order to compute the SDW of two closed curves in near quadratic time.

We can then define the Social Distance Width of two polygons P_1, P_2 as a special case in which R is the boundary of P_1 and B is the boundary of P_2; i.e., $SDW(P_1, P_2) = SDW(\partial P_1, \partial P_2)$. Similarly, the Social Distance Width of a (single) polygon P is $SDW(P) = SDW(\partial P, \partial P)$. We have

Theorem 18 *The social distance width of a polygon P of n vertices can be decided in $O(n^2 \log n)$ time.*

The notion of SDW of a polygon is possibly related to other characteristics of polygons, such as fatness. Intuitively, if the polygon P is fat under standard definitions, then the SDW of P will be large. However, the exact connection is yet unclear (see Fig. 9), and we leave open the question of exact relation between the two notions.
We now show that in special cases, the SDW can be computed in linear-time.

3.4 Social Distancing in a Tree (a "Skinny" Polygon)

In this section we consider the case in which the shared domain of Red and Blue is a tree T, and the distance is the shortest-path distance in the tree (the distance between vertices u and v denoted $|uv|$). Red moves around T in a depth-first fashion: there is no start and end point, it keeps moving ad infinitum. In particular, if T is embedded in the plane, the motion is the limiting case of moving around the boundary of an

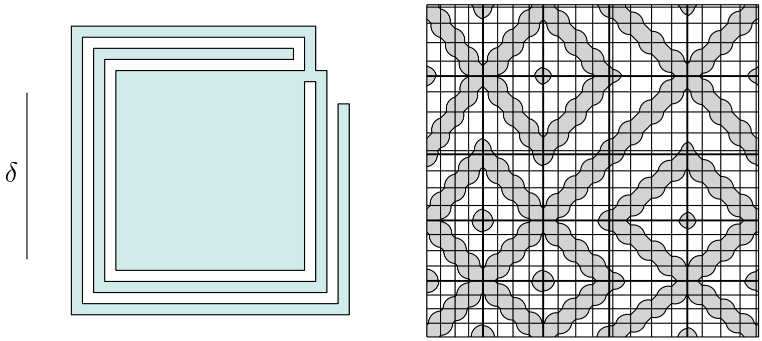


Fig. 9 A δ-fat polygon P, and the free space diagram showing that $SDW(P) \ll \delta$

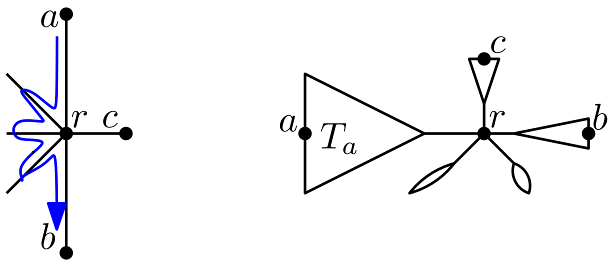


Fig. 10 Red is at c while Blue moves between a and b. Left: A star. Right: A schematic representation of a tree

infinitesimally thin simple polygon, and the distance is the geodesic distance inside the polygon.

Theorem 19 *Let T be a tree with n vertices, embedded in the plane. Suppose Red goes around T (circularly, ad infinitum), and suppose Blue has to do the same. There exists an $O(n)$-time algorithm to schedule the motion of Blue so as to maximize the minimum distance (measured along the tree) between Red and Blue.*

Proof We start with the case when T is a star (Fig. 10, left). Let r be the root of the star and let $|ra| \geq |rb| \geq |rc|$ be the 3 largest distances from r to the leaves (i.e., the distance to the root from all other leaves is at most $|rc|$). Assume that the leaves a, b, c are encountered in this order as Red moves around T (this assumption is w.l.o.g., since the other orders are handled similarly); we call r and $|rc|$ the *2-outlier center* and *radius* of T because allowing 2 outliers, $|rc|$ is the smallest radius to cover T with a disk centered at a vertex of the tree. Now, on the one hand, Blue can maintain distance $|rc|$ from Red: when Red is at a, Blue is at c; when Red is at b, Blue moves to a; when Red is at c, Blue moves to b; the minimum distance of $|rc|$ is achieved when Blue is at c. On the other hand, the distance must be at least $|rc|$ at some point, since Blue cannot sit at a or at b all the time, and, while Blue moves from a to b through r, Red must be somewhere else (other than a or b).

We now consider an arbitrary tree T. Assume w.l.o.g. that T has at least two vertices each of degree at least 3: if all degrees are at most 2, then T is a path and the solution

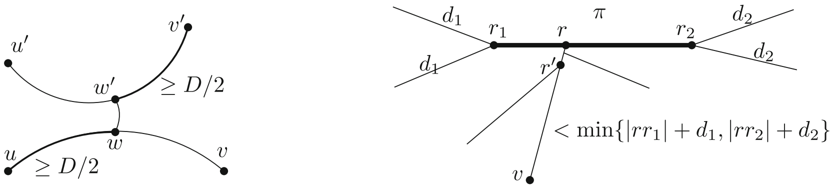


Fig. 11 Left: If $|uv| = |u'v'| = D$, then $|uww'v'| > D$. Right: The distance from any diameter endpoint to the closest endpoint of π (thick) is the d, for otherwise one of the diameters is longer than another. The 2-outlier radius of r is d, while the 2-outlier radius of r' cannot exceed d

is trivial; if there is only one vertex of degree at least 3, the solution is the same as for the star (treating the vertex as the root).

Let $r \in T$ be a vertex of degree at least 3. Removal of r disconnects T into several trees; for a vertex $v \neq r$ of T let $T_v \ni v$ be the subtree of v. Let a be the vertex of T furthest from r, let b be the vertex of $T \setminus T_a$ furthest from r, and let c be the vertex of $T \setminus (T_a \cup T_b)$ furthest from r (Fig. 10, right). Call $|rc|$ the *2-outlier radius* of r, and assume r^* is the vertex whose 2-outlier radius is the largest. As in a star, Blue can maintain the distance of $|r^*c|$ from Red by cycling among a, b, c "one step behind" Red. Also as in a star, a larger distance cannot be maintained because, again, Blue has to pass through r^* on its way from a to b, and the best moment to do so is when Red is at c. If instead Blue does it when Red is, say, at a vertex $a' \in T_a$, then let $l \in T_a$ be the least common ancestor of a and a': if $|la'| < |r^*c|$, then Blue comes within $|la'| < |r^*c|$ from Red; otherwise the 2-outlier radius of l is larger than that of r^*.

To find r^* in linear time, note that ab is a diameter of T: it is a longest simple path in T. The diameter of a tree can be computed in linear time via dynamic programming: pick some arbitrary non-leaf node as the root, and store the depths of subtrees for each node; the diameter is realized by the summed depth of the two deepest subtrees of a node. All diameters of a tree intersect because if two diameters uv, $u'v'$ do not intersect, then there exist vertices $w \in uv$ and $w' \in u'v'$ that connect the two diameters and the distance from each of w, w' to one of the endpoints of its diameter is at least half the diameter, implying that the distance between these endpoints is strictly larger than the diameter (Fig. 11, left). Moreover, since the tree has no cycles, the intersection of all its diameters is a path π in T. Note that the distance from a diameter endpoint to the closest point on π can only have two values (depending on which endpoint of π is closer). We claim for any point r' not on π, the point r on π closest to it has a larger 2-outlier radius, and thus r^* may be found on π. Indeed, let T_r be the tree that contains r after removing r', and let v be the farthest point from r' not in T_r. Since $|r'v|$ is strictly smaller than the distance between r and the closest diameter endpoint (see Fig. 11, right), we get that the 2-outlier radius of r' is at most $|r'v|$. On the other hand, the 2-outlier radius of r is at least $|rv|$, and clearly $|rv| > |r'v|$.

We thus compute a diameter ab (linear time) and pick the vertex with the largest 2-outlier radius on the diameter as r^* by checking the vertices one by one. As we check consecutive vertices on ab, the distances $|ra|$ and $|rb|$ are updated trivially, and the subtrees $T \setminus (T_a \cup T_b)$ are pairwise-disjoint for different vertices r along the diameter; thus the longest paths in all the subtrees can be computed in total linear time. □

4 Conclusion

We considered problems of coordinating the motion of agents while maintaining a certain minimum (social) distance between them; this is a “flipped” variant of the well studied Fréchet distance. We obtained several upper and lower bounds on the complexity of the motion coordination.

Many open questions remain:

- How does social distancing work in polygons with holes?
- What changes if the agents are allowed to backtrack (the “weak” flipped Fréchet distance)?
- How do the solutions and lower bounds scale with the increase of the number of the agents?

Acknowledgements We thank the anonymous reviewers for their helpful comments. We thank the many participants of the Stony Brook CG Group, where discussions about geometric social distancing problems originated in Spring 2020, as the COVID-19 crisis expanded worldwide. We would also like to thank Gaurish Telang for helping to solve a system of non-linear equations.

Author Contributions All contributed equally to everything

Funding Mayank Goswami would like to acknowledge support from NSF grants CCF 1910873 and CRII 1755791. Valentin Polishchuk partially supported by the Swedish Transport Administration and the Swedish Research Council. Joseph Mitchell partially supported by the National Science Foundation (CCF-2007275). Omrit Filtser supported by the ISRAEL SCIENCE FOUNDATION (grant No. 2135/24).

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

1. Alt, Helmut, Godau, Michael: Computing the Fréchet distance between two polygonal curves. Int. J. Comput. Geom. Appl. **5**, 75–91 (1995)
2. Arkin, Esther M., Mitchell, Joseph S. B., Polishchuk, Valentin: Maximum thick paths in static and dynamic environments. Comput. Geom. **43**(3), 279–294 (2010)
3. Baur, Christoph, Fekete, Sándor. P.: Approximation of geometric dispersion problems. Algorithmica **30**(3), 451–470 (2001)
4. Bringmann, Karl: Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In Proc. of the 55th IEEE Annual Symposium on Foundations of Computer Science FOCS, pages 661–670, (2014)
5. Bringmann, Karl, Mulzer, Wolfgang: Approximability of the discrete Fréchet distance. JoCG **7**(2), 46–76 (2016)
6. Buchin, Kevin, Ophelders, Tim, Speckmann, Bettina: SETH says: Weak Fr\'echet distance is faster, but only if it is continuous and in one dimension. arXiv preprint arXiv:1807.08699, (2018)
7. Buchin, Kevin, Ophelders, Tim, Speckmann, Bettina: SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In Proc. of the 30th Annual Symposium on Discrete Algorithms SODA, pages 2887–2901. SIAM, (2019)
8. Chung, Timothy H., Hollinger, Geoffrey A., Isler, Volkan: Search and pursuit-evasion in mobile robotics - a survey. Auton. Robots **31**(4), 299–316 (2011)
9. Cook, Atlas, Wenk, Carola: Geodesic Fréchet distance inside a simple polygon. ACM Trans. Algorithms (TALG) **7**(1), 1–19 (2010)

10. Demaine, Erik D., Fekete, Sándor. P., Keldenich, Phillip, Meijer, Henk, Scheffer, Christian: Coordinated motion planning: reconfiguring a swarm of labeled robots with bounded stretch. SIAM J. Comput. **48**(6), 1727–1762 (2019)
11. Dumitrescu, Adrian, Rote, Günter: On the fréchet distance of a set of curves. In Proceedings of the 16th Canadian Conference on Computational Geometry, CCCG'04, Concordia University, Montréal, Québec, Canada, August 9-11, 2004, pages 162–165, (2004)
12. Eiter, Thomas, Mannila, Heikki: Computing discrete Fréchet distance. Technical report, Citeseer (1994)
13. Fekete, Sándor. P., Meijer, Henk: Maximum dispersion and geometric maximum weight cliques. Algorithmica **38**(3), 501–511 (2004)
14. Filtser, Omrit, Goswami, Mayank, Mitchell, Joseph S. B., Polishchuk, Valentin: On flipping the Fréchet distance. In Proceedings of the 14th Innovations in Theoretical Computer Science Conference, ITCS'23, January 10–13, 2023, MIT, Cambridge, Massachusetts, USA, pages 51:1–51:22 (2023)
15. Guibas, Leonidas J., Hershberger, John: Optimal shortest path queries in a simple polygon. J. Comput. Syst. Sci. **39**(2), 126–152 (1989)
16. Hirsch, Shai, Halperin, Dan: Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. In Jean-Daniel Boissonnat, Joel W. Burdick, Ken Goldberg, and Seth Hutchinson, editors, Algorithmic Foundations of Robotics V, Selected Contributions of the Fifth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2002, Nice, France, December 15-17, 2002, volume 7 of Springer Tracts in Advanced Robotics, pages 239–256. Springer, (2002)
17. Hsiang, Tien-Ruey, Arkin, Esther M., Bender, Michael A., Fekete, Sándor P., Mitchell, Joseph S. B.: Algorithms for rapidly dispersing robot swarms in unknown environments. In Jean-Daniel Boissonnat, Joel W. Burdick, Ken Goldberg, and Seth Hutchinson, editors, Algorithmic Foundations of Robotics V, Selected Contributions of the Fifth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2002, Nice, France, December 15-17, 2002, volume 7 of Springer Tracts in Advanced Robotics, pages 77–94. Springer, (2002)
18. O'Donnell, Patrick A., Lozano-Pérez, Tomás. Deadlock-free and collision-free coordination of two robot manipulators. In Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, May 14-19, 1989, pages 484–489. IEEE Computer Society, (1989)
19. Pollack, Richard, Sharir, Micha, Rote, Günter.: Computing the geodesic center of a simple polygon. Discret. Comput. Geom. **4**(6), 611–626 (1989)
20. Scheffer, Christian: Scheduling three trains is np-complete. In J. Mark Keil and Debajyoti Mondal, editors, Proceedings of the 32nd Canadian Conference on Computational Geometry, CCCG 2020, August 5-7, 2020, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, pages 87–93, (2020)
21. Scheffer, Christian: Train scheduling: Hardness and algorithms. In M. Sohel Rahman, Kunihiko Sadakane, and Wing-Kin Sung, editors, WALCOM: Algorithms and Computation - 14th International Conference, WALCOM 2020, Singapore, March 31 - April 2, 2020, Proceedings, volume 12049 of Lecture Notes in Computer Science, pages 342–347. Springer, (2020)
22. Schlesinger, Michail I., Vodolazskiy, E.V., Yakovenko, V.M.: Similarity of closed polygonal curves in Fréchet metric. arXiv preprint arXiv:1409.4613, (2014)