How to Stay Socially Distant: A Geometric Approach

Omrit Filtser  
Stony Brook University, New York, USA  
omrit.filtser@gmail.com

Mayank Goswami  
Queens College CUNY, Flushing, New York, USA  
mayank.goswami@qc.cuny.edu

Joseph S. B. Mitchell  
Stony Brook University, New York, USA  
joseph.mitchell@stonybrook.edu

Valentin Polishchuk  
Linköping University, Sweden  
valentin.polishchuk@liu.se

Abstract

We introduce the notion of social distance width (SDW) in geometric domains, to model and quantify the ability for two or more agents to maintain social distancing while moving within their respective, possibly shared, domains. Depending on whether the agents’ motion is continuous or discrete, we first study the social distance width of two polygonal curves in one and two dimensions, providing conditional lower bounds and matching algorithms.

We then define the social distance width of a polygon, which measures the minimum distance that two agents can maintain while restricted to travel in or on the boundary of the same polygon, and give efficient algorithms to compute it. We also consider other interesting variants where the agents move on a graph, and provide hardness results and algorithms for both general and special (e.g., trees) types of graphs. We draw connections between our proposed social distancing measure and existing related work in computational geometry, hoping that our new measure may spawn investigations into further interesting problems that arise.

2012 ACM Subject Classification  Theory of computation → Lower bounds and information complexity; Theory of computation → Computational geometry; Theory of computation → Problems, reductions and completeness

Keywords and phrases Algorithms, Curve similarity, Dispersion, Trajectories, Conditional lower bounds, Fréchet distance, Polygons

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding Omrit Filtser: Supported by the Eric and Wendy Schmidt Fund for Strategic Innovation, by the Council for Higher Education of Israel, and by Ben-Gurion University of the Negev. Mayank Goswami: [Supported by National Science Foundation grants CRII-1755791 and CCF-1910873.] Joseph S. B. Mitchell: This work is partially supported by the National Science Foundation (CCF-2007275), the US-Israel Binational Science Foundation (BSF project 2016116), Sandia National Labs, and DARPA (Lagrange). Valentin Polishchuk: [funding]

Acknowledgements 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 solve a system of non-linear equations.
1 Introduction

The current ongoing pandemic has raised many new challenges in various fields of research. In this paper we present a new set of problems, inspired by the challenge of maintaining social distance among groups and individuals. We introduce the notion of social distance width (SDW) in geometric domains, as a way to measure the extent to which two (or more) agents can stay socially distant within some given domain.

In general, we are given \( k \) agents and \( k \) associated domains, with each agent restricted to move only within its respective domain, and at least one of the agents has some mission: it can be either moving from a given starting point to a given end point, or traversing 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.

In this paper, we consider the case of \( k = 2 \), i.e., two agents, Red and Blue, are moving inside their given domains. Further, in this paper, 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 Blue can move at sufficient speed. We begin by considering the scenario in which the two domains are polygonal curves \( R \) and \( B \) of complexity \( m \) and \( n \), respectively.

The agents’ missions are to traverse their respective curves, from the first point to the last point of the curve\(^2\), and the goal is to maximize the minimum distance between the agents. This problem is closely related to the notion of Fréchet distance for curves (see related work below), in which the restrictions on the movement are similar, but the goal is “flipped”: the agents want to minimize the maximum distance between them.

Next, we examine the setting in which both the agents are moving within some simple polygon \( P \). First, we consider the scenario in which only the Red agent has a mission: walk along a given (shortest) path inside the polygon. The Blue agent must stay as far as possible from the Red, and is restricted to move only inside \( P \). This is related to the problem of motion planning, in which we are usually given a set of disks, rectangles, or other fat geometric shapes that represent agents, and the goal is to find a valid sequence of movements that allows them to get from their initial configuration to some given configuration.

Another scenario that we consider for the setting of a simple polygon, is when the Red agent has a mission to traverse the boundary of \( P \), and the Blue agent is restricted to move only on the boundary of \( P \) (but he may choose the starting point). The social distance width of the polygon \( P \) is then the minimum Red-Blue distance throughout the movement, maximized over all possible movement strategies. The notion of SDW of a polygon is 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.

Finally, we consider the setting in which the two agents are moving on the edges of a graph \( G \). Specifically, we investigate the scenario in which the Red agent has to traverse all the edges of the graph, and the Blue agent can move anywhere along the edges of the graph.

Our results. We consider three main settings for social distance width problems. Specifically, our main results include the following:

---

1 For simplicity of presentation, in the following sections we consider only the case of \( m = n \). However, our algorithms and proofs can be easily adapted to the general case of \( m \neq n \).

---
For SDW between two curves, we consider both the continuous case (where the agents are moving continuously along the edges of their curves), and the discrete case (where the agents are only “jumping” between two consecutive vertices of their curves). As mentioned above, these notions of SDW for curves are basically a “flipped” version of the continuous and discrete Fréchet distance, respectively. Indeed, we show that similar upper bounds, conditional lower bounds, and approximation algorithms apply for this problem as well. However, surprisingly, there is one exception: we show that the SDW between two curves in the continuous 1D case can be computed in near linear time.

For SDW in polygons, a scenario for which we present a near linear time algorithm is when Red walks along a shortest path inside a simple polygon $P$, and Blue can move anywhere inside $P$. For the scenario where both agents are walking on the boundary of $P$, and only Red has to traverse the entire boundary, we present a quadratic-time algorithm.

In the graph setting, we give quadratic algorithms for most versions, and show that when the given graph is a tree, then we can compute the SDW in linear time.

Related Work. The Fréchet distance is a well-known and well investigated distance measure for curves, starting with the early work of Alt and Godau [2]. There exists a quadratic time algorithm for computing it [13, 2], however, it was recently shown [5] that under the Strong Exponential Time Hypothesis (SETH), no subquadratic algorithm exists, and not even in one dimension [6]. Moreover, under SETH, no subquadratic algorithm exists for approximating the Fréchet distance up to a factor of 3 [7]. The best known approximation algorithm for Fréchet distance is an $\alpha$-approximation, running in $O((n^3/\alpha) \log n)$ time [10], and in $O(n \log n + n^2/\alpha)$ time for the discrete version [8].

There is a very extensive literature on the related problem of motion planning in robotics. Perhaps most closely related to our work is that of coordinated motion planning of 2 or more disks. See the recent paper of Demaine et al. [11] on nearly optimal (in terms of lengths of motions) rearrangements of multiple unit disks; see also the related work of [12, 16].

The problem of computing safe paths for multiple speed-bounded mobile agents that must maintain separation standards arises in air traffic management (ATM) applications as well. Motivated by ATM applications, Arkin et al. [3] studied the problem of computing a large number of “thick paths” for multiple speed-bounded agents, from a source region to a sink region; here, 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 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 approximation algorithms are known [4, 14]. (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. Here, the problem is to move a swarm of robots, through “doorways”, into a geometric domain, in order to achieve a well dispersed set of agents dispersed throughout the domain. Such movements can be accomplished using simple local strategies that are competitive (see, e.g., [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 [9].


2 Preliminaries

A polygonal curve $P$ in $\mathbb{R}^d$ is a continuous function $P : [1, n] \rightarrow \mathbb{R}^d$, such that for any integer $1 \leq i \leq n - 1$ 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 $\alpha, \beta \in [1, n]$, $\alpha \leq \beta$, we denote by $P(\alpha, \beta)$ the restriction of $P$ to the interval $[\alpha, \beta]$.

Then, for any integer $1 \leq i \leq n - 1$, $P[i, i+1]$ is an edge of $P$.

Let $P : [1, n] \rightarrow \mathbb{R}^d$ and $Q : [1, n] \rightarrow \mathbb{R}^d$ be two polygonal curves. The standard well-studied Fréchet distance between two curves $P$ and $Q$ is defined to be $\inf_{f,g} \max_{t \in [0,1]} \|P(f(t)) - Q(g(t))\|$, where $f : [0, 1] \rightarrow [1, n]$ and $g : [0, 1] \rightarrow [1, n]$ are continuous, non-decreasing, surjections. We call the functions $f,g$ traversals of $P,Q$ respectively.

**Definition 1 (Social Distance Width).** We define the (continuous) social distance width (SDW) of $P$ and $Q$ to be

$$
SDW(P,Q) = \sup_{f,g} \min_{t \in [0,1]} \|P(f(t)) - Q(g(t))\|,
$$

where $f$ and $g$ are traversals of $P$ and $Q$, respectively.

**Free Space diagram:** The $\delta$-free space diagram of two curves $P$ and $Q$ was defined in [2] as a way to represent all possible traversals of $P$ and $Q$ with Fréchet distance at most $\delta$. We adapt this notion to our new setting.

For every two integers $1 \leq i, j \leq n - 1$, let $C_{ij} = [i, i+1] \times [j, j+1]$ be a unit square in the plane. Denote by $B = [1, n] \times [1, n]$ the square in the plane that is the union $\cup_{i,j \in \{1, \ldots, n-1\}} C_{ij}$. Given $\delta > 0$, the $\delta$-free space is $F_\delta = \{(p,q) \in B \mid \|P(p) - Q(q)\| \geq \delta \}$. In other words, it is the set of all red-blue positions for which the distance between the agents is at least $\delta$. A point $(p,q) \in F_\delta$ is a free point, and the set of non-free points (or forbidden points) is then $B \setminus F_\delta$. Note that for Fréchet distance, these definitions are reversed (“flipped”). We call the squares $C_{ij}$ 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 $C_{ij}$ (i.e., $C_{ij} \cap F_\delta$) is convex.

![Figure 1](image)

- **Figure 1** Right: a free space cell $C_{ij}$. For SDW, the free space is white, while for Fréchet distance, the free-space is pink. The pink region is convex. Left: the free space diagram of two curves. The black points and dashed lines indicates a critical value of type (iii), which is an opening in the free space diagram defined by two red edges and one blue edge.

**The discrete case:** When considering discrete polygonal curves, we simply define a polygonal curve $P$ as a sequence of $n$ points in $\mathbb{R}^d$. We denote by $P[1], \ldots, P[n]$ the vertices of $P$, and for any $1 \leq i \leq j \leq n$ let $P[i,j] = (P[i], P[i+1], \ldots, P[j])$ be a subcurve of $P$.

Consider two sequences of points $P, Q \in \mathbb{R}^{d \times n}$. A traversal $\tau$ of $P$ and $Q$ is a sequence of pairs of indexes $(i_1, j_1), \ldots, (i_t, j_t)$ such that $i_1 = j_1 = 1$, $i_t = j_t = n$, and for any pair $(i,j)$ it
holds that the following pair is either \((i, j+1), (i+1, j),\) or \((i+1, j+1)\). The well studied \textit{discrete Fréchet distance} (DFD) between \(P\) and \(Q\) is defined to be \(\min_{\tau} \max_{(i,j) \in \tau} \|P[i] - Q[j]\|\).

- **Definition 2** (Discrete Social Distance Width). \textit{We define the discrete social distance width (dSDW) of \(P\) and \(Q\) to be}

\[
dSDW(P, Q) = \max_{\tau} \min_{(i,j) \in \tau} \|P[i] - Q[j]\|.
\]

Thus, unlike in the continuous case, the distances between the agents are only calculated at the vertices (corners) of the polygonal paths.

### 3 Social distancing on polygonal paths

In this section the domains of Red and Blue are two polygonal curves \(R\) and \(B\) respectively. For simplicity we assume both \(R\) and \(B\) have length \(n\). We first briefly show how to compute (both continuous and discrete) \(SDW(R, B)\) in quadratic time, using the free-space diagram defined in Section 2. We then move to the more interesting question of which versions allow subquadratic time algorithms.

#### 3.1 Computation of SDW

Notice that a monotone path through the free space \(F_\delta\) between two free points \((p, q)\) and \((p', q')\) corresponds to a traversal of \(P[p, p']\) and \(Q[q, q']\). Thus, \(SDW(P, Q) \geq \delta\) if and only if there exists a monotone path through the free space \(F_\delta\) between \((0, 0)\) and \((n, n)\). As mentioned in the introduction, the Fréchet distance between \(P\) and \(Q\) can be computed in \(O(n^2 \log n)\) time [2], as follows; First, solve the decision version of the problem, for a given value of \(\delta\). This can be done in \(O(n^2)\) time, as the set of free points in each cell is convex, which means that determining the possible ways for a path to pass through a single cell can be done in constant time (see Figure 1). Therefore, a reachability graph of size \(O(n^2)\) can be constructed to determine is there exists a path from \((0, 0)\) to \((n, n)\) in the \(\delta\)-free space. Next, they showed that there are \(O(n^3)\) critical values of \(\delta\), which are defined by (i) the distances between starting points and endpoints of the curves, (ii) the distance between vertices of one curve and edges of the other, (iii) and the common distance of two vertices of one curve to the intersection point of the bisector with some edge of the other. Then, they showed that a parametric search based on sorting can be preformed in \(O((n^2 + T_{dec}) \log n)\) where \(T_{dec}\) is the running time for the decision algorithm.

In the case of \(SDW\), we can again compute a reachability graph of size \(O(n^2)\), as in each cell the set of forbidden points is convex, and thus there are only 8 ranges which determine the reachability in a single cell. The set of critical values is almost similar, except that the third type can occur between three edges (see Figure 1, left). Thus, by arguments similar to [2], we have a \(O(n^2 \log n)\) time algorithm for computing \(SDW(P, Q)\).

- **Theorem 3.** There exists an exact algorithm for computing the continuous \(SDW\) for polygonal curves in 2D, that runs in time \(O(n^2 \log n)\).

For the discrete version of \(SDW\), a simple dynamic programming algorithm (similar to the one known for discrete Fréchet distance) gives a \(O(n^2)\) solution. Moreover, the \(f\)-approximation shown in [8] can be adapted as well to the discrete \(SDW\). Details will be given in a full version of this paper.

- **Theorem 4.** There exist exact and \(f\)-approximation algorithms, for the discrete \(SDW\) in 2D, that run in time \(O(n^2)\) and \(O(n \log n + n^2/f^2)\), respectively.
3.2 Can SDW be computed in subquadratic time?

For the sake of brevity, we shall refer to polygonal curves simply as curves. We summarize our answers to the above question: We first give quadratic lower bounds on approximating $dSDW$ and continuous $SDW$ for 2D curves, conditioned on a version of the Strong Exponential Time Hypothesis (SETH). The hardness of approximation factors for the discrete and continuous versions are 0.86 and 0.92, respectively. We then give the same quadratic conditional lower bound for exact $dSDW$ even for 1D curves. Finally, we give a positive result: deciding if $SDW(R, B) \geq 1$ for 1D curves can be done in near-linear time (this is in contrast to the usual Fréchet distance, for which the lower bound holds even for 1d paths, both for continuous and discrete versions [6, 7]).

3.3 2D Lower bounds

We will consider SAT formulas with $N$ variables. SETH states that there is no $\delta > 0$ such that $k$-SAT admits an $O((2 - \delta)^N)$ algorithm for all $k$. Let $\varphi$ be a CNF-SAT formula with clauses $c_1, \ldots, c_M$. We will condition on the following weaker variant of SETH, also used by Bringmann [5] in the lower bound for the usual Fréchet distance:

**SETH’**: There exists no $O^*((2 - \delta)^N)$-time algorithm to decide if $\varphi$ is satisfiable for any $\delta > 0$, where $O^*$ hides polynomial factors in $N$ and $M$.

We will assume that $M$ is even, which loses no generality because if SETH’ fails for even $M$, it fails for odd $M$ as well: we can set true, one-by-one, the (at most $N$) variables in any fixed clause and solve the (at most $N$) instances of CNF-SAT with the remaining (even number of) clauses. We also assume that $N$ is even (otherwise, add a dummy variable).

In the following, by an algorithm with approximation factor $\alpha < 1$ we mean an algorithm that outputs a traversal whose SDW is at least $\alpha$ times the SDW of an optimal traversal. Our first main result is the following conditional quadratic lower bound on an approximation algorithm for the discrete SDW.

**Theorem 5.** [Discrete 2D Lower Bound] There does not exist an algorithm that computes the discrete SDW between two polygonal curves of length $n$ in the plane, up to an approximation factor $\alpha \geq 0.86$, and runs in time $O(n^{2-\delta})$ time for any $\delta > 0$, unless SETH’ (and hence SETH) fails.

**Proof.** We construct paths $R, B$ such that the discrete $SDW(R, B) \geq 1$ iff $\varphi$ is satisfiable. The number of vertices in both paths will be $n \in O(M2^{N/2})$, so an $O(n^{2-\delta})$ algorithm for deciding whether $SDW(R, B) \geq 1$ would imply an $O(M2^{N/2}2^{N(1-\delta/2)})$ algorithm for CNF-SAT, violating SETH’. As in [5], we split the variables into two sets $V_1, V_2$ each of size $N/2$, and denote by $A_1$ (resp. $A_2$) the set of all assignments of $T$ or $F$ to the variables in $V_1$ (resp. $V_2$); $|A_1| = |A_2| = 2^{N/2}$. We first construct assignment gadgets, and then connect them into the paths.

Let $P[i]$ denote the $i$th vertex of the path $P$. Let $P \circ Q$ denote the concatenation of polygonal paths $P$ and $Q$, which consists from $P$, the segment from the last vertex of $P$ to $Q[1]$, and then $Q$; let $\bigcirc$ denote the concatenation of a sequence of indexed paths. For an assignment $\sigma \in A_1 \cup A_2$ and a clause $C$ of $\varphi$ let sat$(\sigma, C)$ be $T$ or $F$ depending on whether $\sigma$ satisfies $C$ or not.

---

3 We remark that in the short version, we stated that the hardness factor for continuous was 0.95; we have improved on this result slightly. Also, in the proof for the quadratic lower bound for discrete SDW, we had to tweak certain values to obtain the new, slightly worse factor of 0.86.
Choose any $0 < \varepsilon < 1/6$ and set the following points: $c_0^0 = (0, \frac{1}{2} + \varepsilon), c_0^1 = (0, \frac{1}{2} - \varepsilon), c_1^0 = (0, -\frac{1}{2} - \varepsilon), c_1^1 = (0, -\frac{1}{2} + \varepsilon), b = (1/2, 0), r = (-1/2, 0)$ (Fig. 2). Observe that with the bounds on $\varepsilon$, $d(c_0^0, b) = d(c_1^1, b) < 1 - \varepsilon$ (i.e., both are within distance $1 - \varepsilon$ of $b$). This is because the $1/6$ upper bound on $\varepsilon$ is the solution to the equation $d(c_0^0, b) = 1 - \varepsilon$, that by plugging in the extremal value of $c_0^0 = (0, 1/2 + \varepsilon)$ gives the quadratic $1/4 + (1/2 + \varepsilon)^2 = (1 - \varepsilon)^2$, whose unique positive solution equals $1/6$. We remark that the fact that we can allow any $\varepsilon < 1/6$ only plays a role in tightening the approximation factor, but for the point of quadratic hardness, the reader can just consider $\varepsilon$ to be small enough.

For any assignment $\sigma_1 \in A_1$, construct the gadget:

$$B_{\sigma_1} = b \circ \bigcirc_{i=1}^{M} c_{\text{sat}(\sigma_1, C_i)}^{i \mod 2}$$

and for any assignment $\sigma_2 \in A_2$ construct the gadget

$$R_{\sigma_2} = r \circ \bigcirc_{i=1}^{M} c_{\text{sat}(\sigma_2, C_i)}^{(i+1) \mod 2}$$

See Figure 2 for the gadgets. One can then prove:

**Lemma 6.** For any $\sigma_1 \in A_1, \sigma_2 \in A_2$:

1. $SDW(B_{\sigma_1}, R_{\sigma_2}) \notin (1, 1 + \varepsilon)$.
2. If $(\sigma_1, \sigma_2)$ satisfies $\varphi$, then $SDW(B_{\sigma_1}, R_{\sigma_2}) \geq 1$.
3. If $(\sigma_1, \sigma_2)$ does not satisfy $\varphi$, then $SDW(B_{\sigma_1}, R_{\sigma_2}) \leq 1 - \varepsilon$, and for any two curves $P, Q$ it holds that $SDW(B_{\sigma_1} \circ P, R_{\sigma_2} \circ Q) \leq 1 - \varepsilon$.

**Proof.** By definition of $b$ and $r$, $\|B_{\sigma_1}[1] - R_{\sigma_2}[1]\| = 1$. If $(\sigma_1, \sigma_2)$ satisfies a clause $C_i$ then $\|B_{\sigma_1}[i+1] - R_{\sigma_2}[i+1]\| \geq 1$ (with equality if exactly one partial assignment satisfies $C_i$; if both of them satisfy the clause then the distance is $1 + \varepsilon$). If $(\sigma_1, \sigma_2)$ satisfies $\varphi$, then the “parallel” traversal $\tau = \langle (1, 1), (2, 2), \ldots, (M, M) \rangle$ has SDW at least 1. If $(\sigma_1, \sigma_2)$ does not satisfy $\varphi$, then there is a clause $C_i$ not satisfied by any of $\sigma_1$ and $\sigma_2$. For any traversal other than the parallel traversal $\tau$, the SDW is at most $2\varepsilon$. For $\tau$, the SDW is at most $1 - \varepsilon$, proving the assertion. Concatenating any paths $P, Q$ to $B_{\sigma_1}, R_{\sigma_2}$ still keeps $SDW(B_{\sigma_1} \circ P, R_{\sigma_2} \circ Q) \leq 1 - \varepsilon$.  

---

**Figure 2** Assignment gadgets: blue edges represent $B_{\sigma_1}$ and red edges represent $R_{\sigma_2}$
We now add few more points and connect the assignment gadgets to form the paths $R$ and $B$. Let $\beta = -\frac{2}{15}(4/\sqrt{3} - 5)$ be the unique negative solution to the quadratic equation: $(x - (2/3))^2 = (5/6)^2 - (0.3)^2$. Set $x = (-1, 0)$, $s_1 = (1, 0.6053)$, $t_1 = (1, -0.6053)$, $s_2 = (0.3, \beta)$, $t_2 = (0.3, -\beta)$, and define $R = x \circ s_2 \circ \bigcirc_{\sigma_2 \in A_2}(R_{\sigma_2}) \circ t_2 \circ x$, $B = \bigcirc_{\sigma_1 \in A_1}(s_1 \circ B_{\sigma_1} \circ t_1)$ (Fig. 3).

\begin{lemma}
The following distance relations hold
1. $s_1$ (resp. $t_1$) is within distance 1 of $t_2$ (resp. $s_2$).
2. $s_1$ (resp. $t_1$) is farther than 1 from $s_2$ (resp. $t_2$).
3. $s_2$ and $t_2$ are within distance 1 to all points in any of the assignment gadgets for Blue.
4. $s_1$ and $t_1$ are at least a distance 1 away from all points in any of the assignment gadgets for Red.
5. $b$ is within distance one of $c^0_T$ and $c^1_T$.
\end{lemma}

**Proof.** The following distances can be computed:

- $d(s_1, t_2) = d(t_1, s_2) \approx 0.8557$.
- $d(s_1, s_2) = d(t_1, t_2) \approx 1.00139$.
- $d(s_2, c^1_T) < d(s_2, c^0_T) < d(s_2, (0, 2/3)) = 5/6$, since the $y$-coordinate of $c^0_T$ is $1/2 + \varepsilon$ which for $\varepsilon < 1/6$, is at most 2/3. Also, $d(s_2, b) = d(t_2, b) < 0.4$.
- This is simply because all points in the assignment gadget are on the $y$-axis, whereas $s_1$ and $t_1$ are on the line $x = 1$.
- $d(b, c^0_T) = d(b, c^1_T) < d(b, (0, 2/3)) = 5/6$.

\begin{lemma}
$\varphi$ is satisfiable if, and only if, the discrete $\text{SDW}(R, B) \geq 1$. $\varphi$ is not satisfiable if, and only if, the discrete $\text{SDW}(R, B) \leq 0.8557$.
\end{lemma}

**Proof:** Assume that the assignment $\sigma = (\sigma_i, \sigma_j)$, $\sigma_i \in A_1, \sigma_j \in A_2$ for some $1 \leq i \leq |A_1|, 1 \leq j \leq |A_2|$ satisfies $\varphi$. Consider a traversal that matches $\bigcirc_{1 \leq k \leq i-1}(s_1 \circ B_{\sigma_k} \circ t_1)$ to
x, and then matches \( s_1 \) to \( s_2 \circ \bigcirc_{1 \leq k \leq j-1}(R_{s_k}) \). Since the number of clauses is even, this (partial) traversal ends at either \((s_1, c_{i_1}^j)\) or \((s_1, c_{i_1}^j)\). The next step matches \( b \) to \( r \). The traversal then matches \( B_{s_1} \) and \( R_{s_j} \) in a “parallel” fashion (Lemma 6), then matches \( t_1 \) to the last vertex of \( R_{s_1} \), stays at \( t_1 \) and matches \( t_1 \) to \( \bigcirc_{j+1 \leq k \leq |A_j|}(R_{s_k}) \circ t_2 \circ x \). and finally matches \( \bigcirc_{i+1 \leq k \mid A_i}(s_1 \circ B_{s_k} \circ t_1) \) to \( x \). One can check that this traversal keeps Red and Blue separated by at least 1.

Conversely, assume that there is a traversal \( \tau = ((\tau_1 := s_1, j_1 := x), \ldots, (\tau_T := s_1, j_T := x)) \) that has SDW at least 1, and let \( \alpha = \max\{t : j_i = s_2\}, \beta = \min\{t : j_i = t_2\} \). Note that \( \alpha < \beta \) (Red hits \( s_2 \) before \( t_2 \)), and that \( i_\alpha = s_1, i_\beta = t_1 \) by (Lemma 7,3). It follows that \( i_\gamma = b \) for some \( \alpha < \gamma < \beta \), which, in turn, implies that \( j_\gamma = r \), as \( r \) is the only point (except \( x \)) on \( R \) that is farther than 1 from \( b \) (Lemma 7, and \( x \neq i_\gamma \) for any \( \alpha < \gamma < \beta \). In words, we have a time \( \gamma \) when Red is at \( r \) and Blue is at \( b \). (Also we know that Red is yet to visit \( t_2 \), and Blue will visit \( t_1 \) at a time \( \beta > \gamma \).) But the only time when Red and Blue can be at \( b \) and \( r \) resp. is before a pair of assignment gadgets, say \( \sigma_1 \) for Blue and \( \sigma_2 \) for Red. (The possibility that Red has finished all assignment gadgets and is now waiting at \( r \) to go to \( t_2 \) next cannot arise, as then Red would be within 1 of Blue.)

The traversal cannot be complete at time \( \gamma \); Red and Blue must move forward. The only way to do it while maintaining a distance at least 1 is to traverse in parallel (“opposite” each other) for the next \( M \) (number of clauses) steps. By Lemma 6, this is possible only if \( (\sigma_1, \sigma_2) \) satisfies \( \varphi \).

To prove the second assertion, observe that:
- \( d(c_{i_1}^j, b) < d((0, 2/3), b) = 5/6 < d(c_{i_1}^j, b') \) for any other point \( b' \in B \), the curve for Blue.
- when Red is at \( s_2 \), Blue can be at either \( s_1 \), or \( c_{i_2}^j \), or \( t_1 \) (all other options/locations for Blue achieve a worse social distancing). Since \( d(s_2, s_1) > 1 \) by 3) in proof of Lemma 7, unless Blue is at \( s_1 \), this maintains a social distancing at most \( \max(d(s_2, c_{i_2}^j), d(s_2, t_1)) = \max(5/6, 0.8557) = 0.8557 \). This is also the distance obtained when Red is at \( t_2 \), unless Blue is at \( t_1 \) then.
- Now consider the case when Blue is at \( s_1 \) while Red is at \( s_2 \), and Blue is at \( t_1 \) while Red is at \( t_2 \). By the same arguments, in the time interval between when Red is at \( s_2 \) and \( t_2 \), Blue must be at \( b \). When Blue is at \( b \), the farthest Red can be (given it cannot be at \( x \) since it hasn’t hit \( t_2 \) yet) is either \( c_{i_1}^j \) or \( c_{i_2}^j \), both of which are at most a distance \( 5/6 < 0.8557 \) away.

This proves that any successful traversal can guarantee a distance at most 0.8557.

**Approximation hardness**: According to Lemma 11, any subquadratic algorithm with approximation factor strictly better than 0.86 will be able to distinguish between the cases \( SDW \geq 1 \) and \( SDW \leq 0.8557 \), thereby resolving satisfiability of \( \varphi \) in subquadratic time, contradicting SETH’. This gives the claimed factor of 0.86.

We now prove a similar lower bound for the continuous SDW in 2D. The proof gets considerably more technical, as the figure for the curves indicates.

**Theorem 9.** [Continuous 2D Lower Bound] There does not exist an algorithm that computes the continuous SDW between two polygonal curves of length \( n \) in the plane, up to an approximation factor at least 0.92, and runs in time \( O(n^{2-\delta}) \) time for any \( \delta > 0 \), unless SETH’ (and hence SETH) fails.
Proof. We first remark that the proof below can be simplified if one is only looking for a quadratic lower bound on an exact algorithm. Most of the work is spent in figuring out the approximation hardness factor.

The points in the construction Let $O$ denote the regular octagon with side length $a = \sqrt{2} - 1$ centered at the origin. Label its eight vertices in counter-clockwise order, starting from $p_1 = (1/2, a/2)$, finishing at $p_8 = (1/2, -a/2)$. Label as $E_i$ the edge starting at $p_i$ and ending at $p_{i+1}$ for $1 \leq i \leq 7$, with $E_8$ being the edge between $p_8$ and $p_1$. The incircle of $O$ is the unit circle centered at the origin.

In what follows $C(p, r)$ denotes the circles of radius $r$ centered at $p$. Let $b = (1/2, 0)$, and let $r_u$ and $r_\ell$ be the two points of intersection of $O$ with $C(b, 1)$. If $\alpha = \frac{1}{4}(1 - \sqrt{2} - \sqrt{5} - 2\sqrt{2})$ and $\beta = \frac{1}{4}(1 + \sqrt{2} - \sqrt{5} - 2\sqrt{2})$, then $r_u = (\alpha, \beta)$ and $r_\ell = (\alpha, -\beta)$. Let $\delta > 0$ (will be chosen later) and define $r_u = r_u + \delta(1, 1) = (\alpha + \delta, \beta + \delta)$ and similarly $r_\ell = r_\ell + \delta(1, -1) = (\alpha + \delta, -\beta - \delta)$. The line segment $\overrightarrow{r_u r_\ell}$ intersects the $x$-axis at the point $r'$, as shown in Figure 4.

Let $b_t$ be the intersection of $E_8$ with $C(r'_u, 1)$. Similarly, let $b_u$ be the intersection of $E_8$ with $C(r'_\ell, 1)$. Let $b' = (1 + \delta - \alpha, 0)$.

Let $\varepsilon > 0$ (will be chosen later), and define $c^0_{2.1} = (0, 1/2 + \varepsilon)$, $c^0_{2.2} = (0, 1/2 - \varepsilon)$ and similarly $c^0_{1.1} = (0, -1/2 + \varepsilon)$ and $c^0_{1.2} = (0, -1/2 - \varepsilon)$. Let $u = (0, 1/2)$ and $d = (0, -1/2)$.

Choosing $\varepsilon$ and $\delta$: Our $\delta$ and $\varepsilon$ will be chosen to satisfy the system:

$$d(b', c^0_{2.1}) = d(b, r'_u) = d(c^0_{2.2}, p_1).$$

Using a numerical solver for a system of non-linear equations, we get that $\varepsilon \approx 0.113$ and $\delta \approx 0.144$ does the job, for which all the three distances are roughly 0.9104. This distance is larger than $d(s_2, c^0_{2.1})$ and $d(s_2, t_1)$ which are unchanged from before.

Define the four points $s_1, s_2$ and $t_2$ as before, that is, $s_1 = (1, 0.6053)$, $t_1 = (1, -0.6053)$, $s_2 = (0.3, -(2/15)(\sqrt{34} - 5))$, $t_2 = (0.3, (2/15)(\sqrt{34} - 5))$. 

**Figure 4** The curves $R$ and $B$ for the continuous SDW lower bound.
The curves in the construction: All our curves are piecewise linear, and we represent them as a sequence of vertices. For \( j \in \{ T, F \} \), define \( \Gamma_r(c_j^0, r) = (c_j^0, d, p_0, p_5, r) \), and define \( \Gamma_r(c_j^0, r) = (c_j^0, d, p_0, p_5, r) \).

For \( j, j' \in \{ T, F \} \) define \( \Gamma_r(c_j^0, c_j^1) = (c_j^0, u, p_3, r', r', p_0, d, c_j^1) \). Similarly, \( \Gamma_r(c_j^0, c_j^2) = (c_j^0, d, p_0, r', r', p_3, u, c_j^2) \).

For \( j \in \{ T, F \} \), define \( \Gamma_d(b, c_j) = (b, p_8, p_7, d, c_j) \). For \( j, j' \in \{ T, F \} \) define \( \Gamma_d(c_j^0, c_j^1) = (c_j^0, u, p_2, p_1, b_u, b', b_t, p_8, p_7, d, c_j^1) \). Similarly, \( \Gamma_d(c_j^0, c_j^2) = (c_j^0, d, p_7, p_8, b_t, b_u, p_2, p_1, p_2, u, c_j^2) \).

Lastly, for \( j \in \{ T, F \} \) define \( \Gamma_d(b, c_j) = (b, u, (1/2, 1/2), (1, 0), t_j) \).

Assignment gadgets: For \( \sigma_1 \in A_1 \) define
\[
B_{\sigma_1} = \Gamma(b, c_{\text{sat}(\sigma_1, C_1)}) \circ \bigcirc_{i=1,\ldots,M-1} \Gamma_d(c_{\text{sat}(\sigma_1, C_1)}, c_{\text{sat}(\sigma_1, C_{i+1})})
\]

For \( \sigma_2 \in A_2 \) define
\[
R_{\sigma_2} = \Gamma(r, c_{\text{sat}(\sigma_1, C_1)}) \circ \bigcirc_{i=1,\ldots,M-1} \Gamma_r(c_{\text{sat}(\sigma_1, C_i)}, c_{\text{sat}(\sigma_2, C_{i+1})})
\]

Recall that since \( M \) is even, the assignment gadget for Blue finishes on top and the assignment gadget for Red finishes on bottom.

**Lemma 10.** For any \( \sigma_1 \in A_1, \sigma_2 \in A_2 \):

1. If \( (\sigma_1, \sigma_2) \) satisfies \( \varphi \), then \( SDW(B_{\sigma_1}, R_{\sigma_2}) \geq 1 \).
2. If \( (\sigma_1, \sigma_2) \) does not satisfy \( \varphi \), then \( SDW(B_{\sigma_1}, R_{\sigma_2}) \geq \max(d(b', c_j^0), d(c_j^0, p_7)) \), and for any two curves \( P, Q \) it holds that \( SDW(B_{\sigma_1} \circ P, R_{\sigma_2} \circ Q) \leq \max(d(b', c_j'^0), d(b, r_u), d(c_j^0, p_7)) \).

**Proof.** Part 1: Observe that the unit circle is the incircle of the octagon \( O \). Starting from \( b \) and \( r \), Blue and Red move opposite each other, in an “antipodal” manner, where antipodes are taken on the octagon. Since the diameter of the unit circle is always contained in the line segment joining the positions of Blue and Red, they maintain distance 1, until Blue hits \( d \) and Red hits \( u \). Since \( C_1 \) must be satisfied by at least one of \( \sigma_1 \) or \( \sigma_2 \), the agent whose assignment satisfies \( C_1 \) moves outside the octagon to \( c_{\tau^j} \) at the same rate as the other agent moves inside \( c_{\tau^j} \). If \( C_1 \) is satisfied by both \( \sigma_1 \) and \( \sigma_2 \), both move outside. At the end of this step they have each traversed the first clause, Red is at \( u \), and Blue is at \( d \). They again move opposite each other, until Red reaches \( r_u \). At this point, Red waits until Blue goes to \( b' \). We observe that all of the segment \( (d, p_7, p_8, b_t) \) is a distance at least 1 away from \( r_u \), so Blue can do this while maintaining distance 1. When Blue is at \( b' \), it waits until Red moves to \( r' \). By construction, \( d(r', b') = 1 \), so Red can do this safely too. Red waits at \( r' \) while Blue moves to \( b_u \). They then move again, in opposite fashion, Red to \( d \) and Blue to \( u \), and repeat this for the next \( m \) steps.

Part 2: We prove this by a case analysis. By Blue “taking clause \( i \)”, we mean the event when Blue is at \( c_{\text{sat}(\sigma_1, C_1)} \), and similarly when Red is at \( c_{\text{sat}(\sigma_1, C_1)} \).

1. In any time interval \( I \) where an agent takes no clauses, the other agent takes at most two clauses. Let Blue correspond to agent 0 and Red to agent 1. If agent \( i \) took no clauses in \( i \) and agent \( 1 - i \) took at most two, then the last clause taken before \( I \) must necessarily be by agent \( i \). Therefore, an agent cannot be more than one clause ahead of another. Since \( (\sigma_1, \sigma_2) \) does not satisfy \( \varphi \), there must exists a clause \( C^* \) that they both do not satisfy. Assume agent \( i \) takes clause \( C^* \) before agent \( 1 - i \). Then agent \( i \) must pass through either the point \( c_{\text{sat}(\sigma_1, C_1)} = (0, 1/2 - c) \) or \( c_{\text{sat}(\sigma_2, C_1)} = (0, -1/2 + c) \). At this instance, the farthest the other agent can be, given its assignment also does not satisfy the clause \( C^* \) which it has to take next, is at \( p_6/p_7 \) or \( p_2/p_3 \). All of these distances equal \( d(c_{\text{sat}(\sigma_1, C_1)}, p_7) \).
2. There is a time interval when an agent takes no clauses, and the other agent takes three or more clauses. If Red is the agent taking no clauses, then Blue must hit $u$ and $d$, and the farthest Red can be at these instants is at $r$. If Blue is the agent taking no clauses, then Red must hit $u$ and $d$, and the best Blue can do it to stay at $b'$; if Blue passes $b'$ to either towards $b_u$ or $b_d$, then when Red reverses and goes to take another clause they will maintain a distance less than what they would have had Blue stayed at $b'$. Therefore, the maximum distance obtained in this case is $d(b', c_F^u)$.

**The curves:** Define $R = (x, s_2) \circ \bigcap_{s_2 \in A_2}(R_{s_2}) \circ (t_2, x)$, $B = \bigcap_{s_1 \in A_1}(s_1 \circ B_{s_1} \circ \Gamma_b^{\text{sat}(\sigma_1, C_M)})$ (Fig. 4).

**Lemma 11.** $\varphi$ is satisfiable if, and only if, the continuous $SDW(R, B) \geq 1$. $\varphi$ is not-satisfiable if, and only if, the continuous $SDW(R, B) < \max(d(b', c_F^u), d(c_F^p, p_T), d(b, r'_u))$.

**Proof.** Assume that the assignment $\sigma = (\sigma_1, \sigma_2)$, $\sigma_1 \in A_1, \sigma_2 \in A_2$ for some $1 \leq i \leq |A_1|, 1 \leq j \leq |A_2|$ satisfies $\varphi$. Consider a traversal that matches $\bigcap_{1 \leq k \leq i-1}(s_1 \circ B_{s_1} \circ \Gamma_b^{\text{sat}(\sigma_1, C_M)})$ to $x$, and then matches $s_1 \rightarrow s_2 \rightarrow \bigcap_{1 \leq k \leq j-1}(R_{s_k})$. Since the number of clauses is even, this (partial) traversal ends at either $(s_1, c_F^u)$ or $(s_1, c_F^p)$, both of which are at least 1 ($s_1$ has $x$-coordinate 1). In the next step Red moves to $r$, and then Blue moves to $b$. The traversal then matches $B_{s_1}$ and $R_{s_2}$ in a “parallel” fashion (Lemma 10), Red finishes at $d$, and Blue then traverses $\Gamma_b^{\text{sat}(\sigma_1, C_M)}$ (we observe that the distance between $d$ and $\Gamma_b^{\text{sat}(\sigma_1, C_M)}$ is at least $3/2\sqrt{2} > 1$). Blue then stays at $t_1$ and Red traverses $\bigcap_{j+1 \leq k \leq |A_2|}(R_{s_k}) \circ (t_2, x)$. Finally, Blue traverse the remaining $\bigcap_{i+1 \leq k \leq |A_1|}(s_1 \circ B_{s_1} \circ \Gamma_b^{\text{sat}(\sigma_1, C_M)})$ while Red stays at $x$. One can check that this traversal keeps Red and Blue separated by at least 1.

Conversely, assume that there is a traversal showing that $SDW(R, B) \geq 1$. Consider the last time $q_{\text{min}}$ when Red is at $s_2$; Blue must be at $s_1$ or possibly on a short connected segment of $B$ containing $s_1$ (all other points in $B$ are within 1 of $s_2$). Let $q_{\text{max}}$ be the first time when Red is at $t_2$; Blue must be at $t_1$ or possibly on a short connected segment of $B$ containing $t_1$ (all other points in $B$ are within 1 of $t_2$). This implies that Blue moved from near $s_1$ to near $t_1$ in the time interval $[q_{\text{min}}, q_{\text{max}}]$, and therefore must have hit $b$ at a first time $q$. At this instant, Red must be on the segment $(r_T, r, r_u)$ as by construction, $r_T$ and $r_u$ are exactly one away from $b$. However, Red is on this segment only between assignments, and must be starting an assignment $\sigma_2$ at time $q$. Similarly, Blue must be starting an assignment $\sigma_3$ at time $q$.

Now it is easy to see that the only way for them to keep moving forward is for them to move opposite each other, in the fashion described in the proof of Lemma 10. This implies that all clauses are satisfied by either $\sigma_1$ or $\sigma_2$, because if any agent moves inside the octagon the other must move outside, since $d(c_F^p, p_T) < 1$.

To prove the second assertion, observe that apart from maintaining a distance at most $\max(d(b', c_F^u), d(c_F^p, p_T)$ once Red and Blue are inside an assignment gadget, a traversal can also guarantee a distance:

$d(b, r'_u)$: When we argued that when Blue is at $b$, Red must be on the segment $(r_T, r, r_u)$, we assumed they were maintaining a distance of 1. However, Red could finish part of an assignment, wait at $r'_u$ when Blue moves to $b$, and “pretend” to start a new assignment, going in parallel opposite to Blue for the next $M$ clauses. This combined with Lemma 10 shows that a traversal could maintain a distance equal to $\max(d(b', c_F^u), d(c_F^p, p_T), d(b, r'_u))$ and still not correspond to a satisfying assignment. This is why we solve the non-linear system $d(b', c_F^u) = d(c_F^p, p_T) = d(b, r'_u)$.
\(d(s_2, c^0_F):\) Blue finishes all its assignments and stays at \(c^0_F\), and only then Red moves to \(s_2\). Their distance now exactly \(d(s_2, c^0_F)\), Blue and Red now move to \(t_1\) and \(r\) respectively. Red traverse the entirety of \(R\), returns to \(x\), and Blue moves to \(s_1\). Fortunately, when we solve \(d(b, c^0_F) = d(c^0_F, p_T) = d(b, r')\), we obtain each of these distances to be roughly 0.9104. This distance is larger than \(d(s_2, c^0_F)\) and \(d(s_2, t_1)\) which were at most 0.8557.

\[\text{Approximation hardness:}\] By the previous lemma, we know that an approximation algorithm with an approximation factor at least 0.92 will be able to distinguish between the case \(SDW \geq 1\) versus \(SDW \leq 0.9104\). This completes the proof of the Theorem.

**Weak Versions:** One may also be interested in the \textit{weak} SDW, where, analogous to the weak Frechet distance, the traversal is allowed to backtrack. We prove:

\[\text{Theorem 12.} \] \textbf{[Weak SDW Lower Bound]} There does not exist an algorithm that computes the:

- weak SDW of two curves up to an approximation factor \(\alpha > 0.86\), or
- continuous SDW of two curves exactly,

and runs in time \(O(n^{2-\varepsilon})\) for any \(\varepsilon > 0\), unless SETH’ (and hence SETH) fails.

\[\text{Proof.}\] We prove the discrete version, and the continuous version can be proven similarly. The assignment gadgets are the same as in the proof of Theorem 5. However, the construction of the curves is slightly different, in that Blue has to go to an extra point \(e = (1/4, 0)\) between moving from \(t_1\) to \(s_1\). Please see Figure 5.

As before, if there exists a pair of satisfying assignments \((\sigma_1, \sigma_2)\), for Blue and Red, respectively, then they can maintain a social distance width of 1 by traversing exactly the same way as in the proof of Theorem 5. To summarize, Blue traverses \(B\) until it is about to start \(\sigma_1\), and waits at \(s_1\). Red then traverse \(R\) until it is about to start \(\sigma_2\), and is at \(r\). Blue moves to \(b\) in the next time step. Now Blue and Red run the assignment gadget for \((\sigma_1, \sigma_2)\) “in parallel”, with Red finishing at one of the bottom clause points and Blue at one of the top clause points. Blue moves to \(t_1\), Red finishes traversing its curve, and go back to \(r\). Blue then finishes traversing its curve.

We now prove that if there is a traversal of length 1, then there must exist a pair of satisfying assignments \((\sigma_1, \sigma_2)\). As before, this will be shown by proving that Red and Blue must traverse at least one assignment gadget in parallel fashion.

Let \(i'\) be the last time Red is at \(s_1\), and \(j\) be the first time Red is at \(t_2\). Because \(t_1\) and \(s_1\) are the only points on \(B\) farther than \(s_1\) and \(t_1\), respectively, Blue must be at \(s_1\) on time \(i'\) and at \(t_1\) on time \(j\). Therefore, it must have moved from \(s_1\) to \(t_1\) in the time interval \([i', j]\). This is where the extra point comes in. Before, we could just say that Blue must have moved from \(s_1\) to \(t_1\) via \(b\). However, in the weak version Blue is also allowed to travel “backward” from \(s_1\) to \(t_1\). Nevertheless, we observe that the extra point \(e\) is within 1 of any point Red could possibly be at in the interval \([i', j]\). Thus this route of going to \(t_1\) from \(s_1\) via \(e\) is closed for Blue.

This means that Blue passed through \(b\) en-route from \(s_1\) to \(t_1\) at some time, the last of which we denote as \(i\). \(r\) is the only point Red can be at at this time, and Red is at \(r\) only just before it is about to start an assignment \(\sigma_2\). By definition of \(i\), Blue moves forward at time \(i + 1\), and it must do so when starting a new assignment \(\sigma_1\); Red must also move forward in a parallel fashion, for the next \(m\) steps, as otherwise they would end up in the same clause gadget. This gives us the pair of satisfying assignments.
All our points have the same coordinates as in the proof of Theorem 5, except the new point \( e \). However, \( e \) is at most distance \( \sqrt{73}/12 \approx 0.712 \) from either \( c_0^T \) or \( c_1^T \), making \( r \) the farthest point from it, with distance 0.75. Since this is less than the distances we considered while considering the approximation hardness in Theorem 5, the same hardness of approximation holds.

**Figure 5** The curves \( R \) and \( B \) for the weak discrete SDW lower bound.

### 3.4 1D Lower and Upper Bounds

Having obtained evidence that computing either the discrete or continuous SDW in 2D requires quadratic time, with matching algorithms available, we now turn our focus to 1D. For the 1D case we show that while the discrete version still (likely) has no subquadratic algorithms, the continuous version is solvable in near-linear time! We first state our lower bound for the discrete case, and then our \( O(n) \) algorithm for the continuous case.

Using the problem of Orthogonal Vectors (OV), Bringmann and Mulzer [6] gave a lower bound (conditioned on SETH) for computing the usual discrete Fréchet distance between two 1D paths. We prove a similar result below.

► **Theorem 13.** *(Discrete 1D Lower Bound)* There does not exist an \( O(n^{2-\varepsilon}) \) time algorithm that computes the discrete SDW of two paths in 1D, unless SETH fails.

**Proof Sketch:** A path in 1D is simply a sequence of points. Our paths \( R \) and \( B \) have vertices at 9 points of the real line: \( w_1 = -w_2 = 23/14, x_2 = -x_1 = 1, a_0^1 = b_0^1 = -a_0^0 = -b_0^0 = 10/14, a_0^2 = b_0^2 = -a_0^1 = -b_0^1 = 6/14, s = 0 \) (Fig. 6). The crucial properties are:

- \( w_1 \) far from: \( a_0^0, a_0^1, s, x_1 \)
- \( w_2 \) far from: \( a_0^0, a_0^1, s, x_2 \)
- \( x_1 \) close to: \( w_2, b_0^0, b_0^1 \)
- \( x_2 \) close to: \( w_1, b_0^0, b_0^1 \)
As in [6], we assume that $D$ is even (otherwise duplicate a dimension). Given an instance of $OV$, we list construct vector gadgets from $U$ and $V$. For each $u_i \in U$, we create a sequence $A_i$ of $D$ vertices of $B$: for odd (resp. even) $k$ the $k$th point in $A_i$ is $a_{u_i}^k$ (resp. $a_{u_i}^k$). Similarly, for $v_j \in V$, we create a subsequence $B_j$ of $R$, using $bs$ instead of $as$. It is easy to see that Red and Blue can traverse $B_j$ and $A_i$ while maintaining distance at least 1 iff $u_i, v_j$ are orthogonal (they jump between odd and even points in sync 'opposite' each other, and at least one of them is at 'far' point, indexed with 0).

The gadgets $B_j$ are connected into $R$ as follows: $R = w_1 \circ B_1 \circ w_2 \circ w_1 \circ B_2 \circ w_2 \circ \ldots \circ w_1 \circ B_N \circ w_2$. Let $W$ be the sequence of $D(N - 1)$ points that alternate between $a_0^0$ and $a_0^1$ starting with $a_0^0$ (Blue can traverse each of $N - 1$ length-$D$ subpaths of $W$ in sync with Red on any $B_j$); define $B = W \circ s_1 \circ s_0 \circ a_i \circ s_0 \circ a_j \circ \ldots \circ s_0 \circ w_N \circ s_0 \circ w_2 \circ W$.

The proof that $OV$ is a Yes instance iff the discrete $SDW(B, R) \geq 1$ is similar to that in [6]. If $u_i \in U, v_j \in V$ are orthogonal, then Blue traverses $D(N - j)$ points on $W$ while Red stays at $w_1$, then Red traverses $B_1 \ldots B_{j-1}$ in sync with Blue traversing the rest of $W$, then Blue goes to $x_1$ and traverses $A_1 \ldots A_{i-1}$ and goes to $s$ before $A_i$ while Red stays at $w_1$ before $B_j$, then $A_i, B_j$ are traversed in sync, etc. For the converse, when Blue is on $x_1$, Red must be left of $s$, but if Red is not on $w_1$, then they cannot take the next step – so Red must be on $w_1$; immediately after leaving $w_1$, Red gets to $b_0^0$ or $b_0^1$, implying that Blue must be at $a_0^0$ or $a_1^0$, i.e., either in a vector gadget $A_i$ or on the second $W$ (since it already passed $x_1$). But if it is on $W$, it must have gone through $x_2$ which is close to $w_1$, so Blue is at $A_i$ (and if it is not in the first point of the vector gadget, it will finish $A_i$ and appear at $s$ before Red has finished $B_j$). See [6] for details.

**A near linear time algorithm for continuous $SDW$ in $1D$**

We now turn to the continuous $SDW$. Let $R'$ and $B$ be two paths in $1D$; assume that $B$ starts to the left of $R'$ ($B[1] < R'[1]$). To have $SDW(B, R') > 1$ it must hold that $R'[1] - B[1] > 1$. Let $R$ be $R'$ shifted by 1 to the left. Clearly, $SDW(B, R') > 1$ iff $SDW(B, R) > 0$, i.e., if Red and Blue can traverse $R$ and $B$ so that Blue always stays...
to the left of Red - call such a traversal non-crossing. We will show how to find a non-crossing traversal time nearly-linear in the complexity of $R$ and $B$; we will use the continuity extensively, which is not surprising in view of the lower bound for the 1D discrete SDW from the previous section. We will assume that $B$ and $R$ do not have coinciding vertices. We first prove some lemmas, and use them to motivate the steps of our algorithm.

Let $b_{\text{min}}$ be the leftmost point of $B$ and $r_{\text{max}}$ be the rightmost point of $R$. If $SDW(B, R) > 0$, then all of $R$ must be to the right of $b_{\text{min}}$: by continuity, if $R$ has a point left of $b_{\text{min}}$, Red must cross Blue before getting to that point. This implies

► **Lemma 14.** Suppose $SDW(B, R) > 0$. While Blue is at $b_{\text{min}}$, any subpath of $R$ can be traversed by Red. While Red is at $r_{\text{max}}$, any subpath of $B$ can be traversed by Blue.

► **Lemma 15.** If $SDW(B, R) > 0$, then there exists a non-crossing traversal such that at some point Blue is at $b_{\text{min}}$ and Red is at $r_{\text{max}}$.

The above lemma allows us to assume w.l.o.g. that $b_{\text{min}}$ and $r_{\text{max}}$ are the first points of $B$ and $R$ resp. ($B[1] = b_{\text{min}}, R[1] = r_{\text{max}}$): for arbitrary $B, R$ we can first solve the problem for $B[b_{\text{min}}, end]$ vs $R[r_{\text{max}}, end]$, and then for the reversed paths $B[1, b_{\text{min}}], R[1, r_{\text{max}}]$.

Let $b_{\text{max}}$ be the rightmost point of $B$ (if there are ties, take the point closest to the end of the path). Let $r^* \in R$ be the last point on $R$ at $b$ (i.e., at the x-coordinate of $b$; in other words $r^*$ is the last point where $R$ intersects the vertical line $x = b$ and thus goes over $b$). Similarly, let $r_{\text{min}}$ be the leftmost point of $R$ and let $b^*$ be the last point of $B$ intersecting the vertical line $x = r_{\text{min}}$ (Fig. 7, middle). We consider different cases of how $b^*, b_{\text{max}}$ and $r^*, r_{\text{min}}$ are located along $B$ and $R$ resp.

If $b_{\text{max}} = b^*$ (Fig. 7, right), then since $b^*$ coincides with $r_{\text{min}}$, we have that $B$ is to the left of the common abscissa of $b^*, b_{\text{max}}$ and $r_{\text{min}}$, while $R$ is to the right. Hence, any traversal will be non-crossing (mod the trivial case when $b_{\text{max}}$ and $r_{\text{min}}$ are the endpoints of their paths). Similarly, we are done if $r_{\text{min}} = r^*$. In what follows we treat the cases when $b_{\text{max}} \neq b^*, r_{\text{min}} \neq r^*$.

**Notation:** For two points $p, q$ on the same path such that $p$ precedes $q$, write $p < q$.

► **Lemma 16.** If $b^* < b_{\text{max}}, r^* < r_{\text{min}}$, then there is no non-crossing traversal.

**Proof.** By continuity, Blue must visit $b_{\text{max}}$ before Red visits $r^*$, while Red must visit $r_{\text{min}}$ before Blue visits $b^*$.

We are thus left with 3 cases: (1) $b^* > b_{\text{max}}, r^* > r_{\text{min}}$, (2) $b^* > b_{\text{max}}, r^* < r_{\text{min}}$, (3) $b^* < b_{\text{max}}, r^* > r_{\text{min}}$. Cases (2) and (3) are symmetric, so assume w.l.o.g. that $b^* > b_{\text{max}}$.

By our assumption that $B$ and $R$ do not have coinciding vertices, $b^*$ is not a vertex (since $r_{\text{min}}$ is): the edge of $B$ that contains $b^*$ has a point $b^-$ to the left of $b^*$; similarly, the edge of $R$ containing $r^*$ has a point $r^+$ to the right of $r^*$. Since by our assumption (made w.l.o.g.,
thanks to Lemma 15) $B[1]$ is the leftmost point of $B$, by Lemma 14, Red can go to $r^+$ while Blue sits at the start. By definition of $r^*$, Blue can then go to $b^-$ while Red sits at $r^+$.

**The Algorithm:** We have thus reduced our problem to the one in which the paths have fewer vertices. Moreover, by definitions of $b^*$ and $r^*$, the new, shorter paths still have the property that they start from leftmost (for Blue) and rightmost (for Red) points. We can thus recurse until we are stuck (Lemma 16) or done.

For the runtime, our recursive solution involves answering semi-dynamic maximum/minimum queries to obtain $b_{\text{max}}/r_{\text{min}}$ (the queries are semi-dynamic because the paths only shorten), and semi-dynamic queries for the first time of reaching $b_{\text{max}}/r_{\text{min}}$ (to obtain $r^*, b^*$). A query of the first type can be answered in constant time after storing running maximums of the subpaths (linear time). A query of a second time can be answered in $O(\text{polylog } n)$ time where $n$ is the maximum complexity of $B, R$, e.g., as follows: Turn each of $B, R$ into a monotone (and hence simple) 2d path by lifting its vertices (cf. Fig. 7). Build the hierarchy of $O(\log n)$ convex hulls of vertices of the path: the convex hulls of 2 consecutive vertices (i.e., the edges) on the first level of the hierarchy, of 4 vertices on the second, and so on, ending with the convex hull of the whole path at the last level. Any convex hull can be maintained in logarithmic time per vertex deletion. Treat the query as a vertical ray coming from infinity. During the query, determine which edge of the last-level convex hull the query ray intersects first (logarithmic time), and recurse down the hierarchy (polylogarithmic time overall per query). We have proven:

> **Theorem 17.** [Continuous 1D Algorithm] There exists an algorithm that computes the continuous SDW of two curves in 1D in $O(n \text{polylog } n)$ time.

### 4 Social distancing in a simple polygon

In this section we consider distancing problems in which the given domain is a simple polygon.

#### 4.1 Blue distancing from Red-on-a-mission

We first switch to the asymmetric case in which Blue is not restricted to stay on a given path. Of course, if Blue had no restrictions at all, it would trivially go to infinity to stay far from Red on any path. We therefore restrict the domain to a simple polygon $P$ and use the geodesic (shortest) paths within $P$ to measure distances (the motivation to consider geodesic social distancing is that the infection spread is also confined to $P$).

> **Theorem 18.** Assume that Red moves along the geodesic path $\pi$ between two given points $r$ and $r'$ 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 $b$. There exists an $O(n)$-time algorithm to decide whether Blue can maintain the (geodesic) social distance 1 from Red, where $n$ is the complexity of $P$.
The disk $D_t$ splits $P$ into connected components (a component is a maximal connected subset of $P \setminus D_t$). Blue can freely move inside a component without intersecting $D_t$; in particular, if the component $P' \supset b$ of $b$ is not equal to $M \setminus D_t$ (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 $M$). The next lemma asserts that the existence of such a safe point is also necessary for Blue to maintain the distance of 1.

**Lemma 19.** If $P' = M \setminus D_t$, then there is no traversal for Blue that maintains distance at least one from Red.

**Proof.** Indeed, as Red follows $\pi$, $D_t$ sweeps $M$; let $S_t \subseteq M$ be the points swept by the time Red is at $t \in \pi$ and let $U_t = M \setminus S_t$ be the unswept points. Since $b \notin D_r = S_r$, initially Blue is in the unswept region. Assume that there is no safe point ($P' = M \setminus D_t$) and yet Blue can escape. Suppose Blue can escape from the unswept to swept when Red is at $t \in \pi$ (Fig. 8, middle). 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 $|t^*p| \leq 1$. Since $p$ is on the boundary of $U_t$, there exists an unswept point $p' \in U_t$ within distance less than $\varepsilon$ 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 $t$: $|t'p| = 1$. We obtain that there are three points $t^*, t, t'$ along a geodesic path $\pi$ and a point $p$ such that $|t^*p| \leq 1 < 1 + \varepsilon = |tp|$ and $|t'p| \leq |t'p'| + |p'p| < 1 + \varepsilon = |tp|$, contradicting the fact that the geodesic distance from a point to a geodesic path is a convex function of the point on the path [18, Lemma 1] (this is the place where we use that $\pi$ is a geodesic path: if $\pi$ is not geodesic, it is not necessary for the Blue to escape from $M$ while Red is at $r$ – see Fig. 8, right; for an arbitrary path $\pi$ the problem can be solved in nearly-quadratic time using the free-space diagram for Red on $\pi$ and Blue on the boundary of $P$, since w.l.o.g. Blue is most separated from Red when Blue is on the boundary – this may be close to best possible because for a self-intersecting $\pi$ the complexity of the set of safe points can be quadratic).
We now show how to implement our solution to the decision problem \(^4\) 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 \(\pi\). To build the SPM, we decompose \(P\) by drawing perpendiculars to the edges of \(\pi\) at every vertex of the path (Fig. 9): 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 \(\pi\) 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 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.

Remark: Our linear-time algorithm above required the path for Red to be a geodesic. If this is not the case, the algorithms in the next section can be used to give a quadratic time solution to the problem. We leave open the question of the existence of a sub-quadratic algorithm for this version of the problem.

4.2 The Social Distance Width of a polygon

Consider two agents, Red and Blue, walking on the boundary of a polygon \(P\) with \(n\) vertices. The starting point of Red is some point \(s_r\) on \(\partial P\), and it moves at constant speed, clockwise around \(\partial P\). Given a distance threshold \(\delta\), our goal is to find a starting point \(s_b\) and a movement strategy for Blue, moving with unbounded speed, such that Blue is always at (geodesic or Euclidean) distance at least \(\delta\) from Red. The minimum \(\delta\) that allows Blue to avoid Red, over all possible starting points \(s_r\), is the Social Distance Width (SDW) of \(P\), \(^4\)

\(^4\) We remark that while we solve the decision version, standard techniques for converting decision algorithms for Fréchet-type problems into optimization ones \([2]\) can potentially be used to compute the largest geodesic social distance that Blue can maintain
denoted SDW($P$). We refer to the two versions corresponding to the Euclidean and geodesic distance functions as ESDW($P$) and GSDW($P$), respectively.

This problem can be solved in a similar manner to the problem of computing the Fréchet distance between two closed curves. Alt and Godau [2] presented an $O(n^2 \log^2 n)$ time algorithm for the Fréchet problem, which was improved later by Schlesinger et. al. [19] to $O(n^2 \log n)$ time. Those algorithms include the construction of dynamic data structures for the free space diagram. However, in our problem, Red and Blue are walking on the same closed curve, $\partial P$, a case that does not make sense for the classic Fréchet distance. Therefore, by making some interesting observations on the free-space diagram, we suggest a much simpler solution in $O(n^2 \log n)$ time.

\begin{theorem}
Given a value $\delta$, if there exists a strategy for Blue to maintain distance at least $\delta$ from Red, when Red is patrolling $\partial P$ for $m \geq n$ rounds, then such a strategy can be computed in $O(n^2)$ time.
\end{theorem}

In order to prove Theorem 20, we first need a few observations. Denote the vertices of $P$ by $p_1, p_2, \ldots, p_n$, and consider the two closed polygonal curves $R : [0, n] \to \mathbb{R}^d$ and $B : [0, 1] \to \mathbb{R}^d$ such that $R(i) = B(i) = p_{i+1}$ for $0 \leq i \leq n-1$, and $R(n) = B(n) = p_1$. Clearly, $B = R = \partial P$. Let $\mathcal{F}_\delta$ be the free space diagram of $R$ and $B$, for some $\delta > 0$.

Notice that in the case of closed curves, the diagram is “cyclic” in the sense that it can be “folded” into a tube either on its vertical or horizontal boundary. Thus, a path in the diagram can exit at the top (resp. right) boundary and enter again at the respective point of the bottom (resp. left) boundary (see Figure 10). We say that a path $\Pi$ in the diagram is $y$-monotone if for any two points $(r, b), (r', b') \in \Pi$ such that $(r, b)$ appears before $(r', b')$ in $\Pi$, we have $r' \geq r$.

A $y$-monotone path in $\mathcal{F}_\delta$ from a point $(s, s')$ to a point $(t, t')$ corresponds to a traversal in which Red walks from $R(s)$ to $R(t)$ without backtracking while Blue walks from $B(s')$ to $B(t')$ (possibly with backtracking), and the distance between them is at least $\delta$ at any point in time. Therefore, there exists a strategy for Blue to maintain distance at least $\delta$ from Red, if and only if there exists a sequence of $y$-monotone paths $\Pi_1, \Pi_2, \ldots, \Pi_k$ in $\mathcal{F}_\delta$ such that $^5$

$\Pi_1$ starts at a point $(s_0, 0)$ on the lower boundary of $\mathcal{F}_\delta$ and ends at a point $(n, s_1)$ on the right boundary of $\mathcal{F}_\delta$ ($\Pi_1$ might move through the bottom boundary to the top boundary and back), then $\Pi_2$ starts at a point $(0, s_1)$ on the left boundary of $\mathcal{F}_\delta$ and ends at a point $(s_2, n)$ on the top boundary of $\mathcal{F}_\delta$. Now $\Pi_3$ start at a point $(s_2, 0)$ on the lower boundary of $\mathcal{F}_\delta$, and we continue this way for the rest of the paths in the sequence (see Figure 10).

Observe that if one of the paths $\Pi_i$ for $i > 1$ reach $(s_0, 0)$, then we can repeat the strategy from this step onward and get a finite sequence of paths. In addition, if two paths $\Pi_i$ and $\Pi_j$ for $i < j$ are crossing, then we could find a shorter sequence of paths by going directly from the entry point of $\Pi_i$ to the exist point of $\Pi_j$, and removing $\Pi_{i+1}, \ldots, \Pi_{j-1}$ from the sequence. Therefore, it must hold that either $s_0, s_2, \ldots, s_{2i}, \ldots$ are monotonically increasing and $s_1, s_3, \ldots, s_{2i+1}, \ldots$ are monotonically decreasing, or vice versa.

\footnote{Note that $\Pi_1$ is optional, and the sequence might start with $\Pi_2$. The proof applies for this case as well.}
Consider a cell $C_{ij}$ of $F_{\delta}$, and recall that the set $C_{ij} \setminus F_{\delta}$ of forbidden points is a convex shape. In the following observation we claim that a path through the free-space of $C_{ij}$ can always avoid entering the bounding box of the forbidden points $C_{ij} \setminus F_{\delta}$ (see Fig. 11).

**Observation 1.** Let $R$ denote the bounding rectangle of $C_{ij} \setminus F_{\delta}$. For any $y$-monotone path $\Pi$ between two points $s$ and $t$ on the boundary of $C_{ij}$ there a $y$-monotone path $\Pi'$ between $s$ and $t$ that do not contain any point on the interior of $R$.

For each cell $C_{ij}$ we construct the four orthogonal tangents to $C_{ij} \setminus F_{\delta}$, and extend them horizontally and vertically until hitting a forbidden point. Now consider the arrangement of segments that constitute the set of all tangents and cell boundaries in $F_{\delta}$. Let $G$ be the embedded graph implied by this arrangement, i.e., the graph whose vertices are the set of all intersection points, and whose edges connect pairs of consecutive vertices along the same segment. By Observation 1, if there exists a $y$-monotone path $\Pi$ between two points $s$ and $t$ on the boundary of $F_{\delta}$ then there exists a $y$-monotone path $\Pi'$ between $s$ and $t$ whose edges lying on the edges of $G$. In other words, there exists a path $\Pi'$ in $G$ between two vertices $s'$ and $t'$, such that $s$ is on an edge of $G$ adjacent to $s'$, and $t$ is on an edge of $G$ adjacent to $t'$, and the path $s \circ \Pi' \circ t$ is $y$-monotone.

We direct the edges of $G$ such that the vertical edges are bi-directional, and the horizontal edges are directed from left to right.

Now we are ready to prove Theorem 20. Assume that there exists a strategy for Blue to maintain distance at least $\delta$ from Red. Then, there exists a sequence of $y$-monotone paths $P_1, \ldots, P_k, \ldots$ in $F_{\delta}$ as described above, where either $p_0, p_2, \ldots, p_{2i}, \ldots$ are monotonically...
increasing and \( p_1, p_3, \ldots, p_{2i+1}, \ldots \) are monotonically decreasing, or vice versa. In both cases, after at most \( n \) steps, one of the sequences will have two points on the same edge of \( G \). Then, by the above observation we get that there exist corresponding paths in \( G \). This means that there always exists a finite and repetitive strategy (even if Red continues to patrol \( P \) in an infinite number of rounds). Moreover, when we add the “cyclic” edges to \( G \), i.e. connecting vertices from the top to their respective edges on the bottom, and vertices from the right with directed edges to their corresponding vertices on the left, we get that a successful strategy corresponds to a cycle in the directed graph. Since the size of \( G \) is \( O(n^2) \), we can compute a strategy for Blue in \( O(n^2) \) time as required.

**Remark 21.** To find \( ESDW(P) \) (where Red does a single traversal of \( \partial P \), instead of adding the “cyclic” right-left edges to the graph \( G \), we add a source and sink vertices to all the vertices the first and last column, respectively, and check if they are connected.

### 5 Graphs

In this section we consider social distancing measures for graphs. An undirected graph \( G = (V, E) \) will be assumed to have weights \( w_e = 1 \) for all \( e \in E \). A geometric graph is a connected graph with edges as straight line segments embedded in the plane. We consider geometric graphs to be automatically weighted, with Euclidean edge weights. Since we are interested in formulating meaningful distance measures, we assume that an all-pairs shortest path preprocessing has been done on \( G \), and pairwise distances are available in \( O(1) \) time. Let \( n = |V| \) and \( m = |E| \). By a path of length \( k \) in \( G = (V, E) \) we mean a sequence \( P \) of \( k \) vertices: \( P = \{p_1, p_2, \ldots, p_k\} \subseteq V \), with edges \( p_ip_{i+1} \in E \) for all \( 1 \leq i \leq k-1 \), and we write \(|P| = k \). A curve on an unweighted graph will be assumed be discrete, and therefore corresponds to a path. A curve on a geometric graph will be assumed to be a continuous map from the unit interval into the graph.

**Blue distancing from Red on a mission:** We first state our result for Blue staying away from Red on an undirected graph.

**Theorem 22 (Blue distancing from Red, non-geometric).** Assume Red travels on a known path of length \( k \) in an unweighted graph, and Blue can travel anywhere with a speed \( s \) times that of Red. There exists a decision algorithm for the Blue distancing from Red-on-a-mission problem that,

- for \( s \in \{1, n-1\} \), runs in time \( O(km) \),
- for arbitrary \( 1 < s < n-1 \), runs in time \( O(nk \min(d^n, n)) \), where \( d \) is the maximum degree of any vertex in \( G \).

Now we consider geometric graphs. Let \( R : [0, \ell] \to \mathbb{R}^2 \) be a polygonal curve in \( \mathbb{R}^2 \), consisting of \( \ell \) line segments, with the \( i \)th line segment \( R_i = R_{[i-1,i]} \) where \( 1 \leq i \leq \ell \). We also assume that each segment \( R_i \) is parameterized naturally by \( R(i + \lambda) = (1 - \lambda)R(i) + \lambda R(i + 1) \). The curve \( R \) is assumed to be known: this is how Red is traveling. On the other hand, we are also given a geometric graph \( G \) in which Blue is restricted to travel, and the problem is to determine if there is a path \( P \in G \) (a path in \( G \) is the polygonal curve formed by the edges between the start and end points of \( P \)), such that \( SDW(R, P) \geq \delta \), where the SDW between \( R \) and \( P \) is defined as in the continuous SDW between polygonal curves in Section 2. Note that by adopting this definition we are considering the Euclidean distance between Red and

---

6 We focus on undirected graphs, but our results readily extend to directed graphs.
Blue: if one instead wants to consider the geodesic distance, Red will need to be restricted to also travel in $G$. We remark on this setting later.

**Theorem 23** (Blue distancing from Red, geometric). The decision problem, given $R$ (the path of Red) and $G$ (the graph of Blue) can be solved in time $O((\ell m)$, where $\ell = |R|$ and $m$ is the number of edges in $G$.

We also note that our conditional lower bound on weak SDW Theorem 12 also implies that the above result is essentially tight, as the above setting is more general.

**Proof.** We first define the free space surface for our problem. Consider an edge $e_{i,j} = (v_i, v_j) \in E$, and let $F^i_j$ denote the $\delta$-free space of $e_{i,j}$ and $r$ (note that $e_{i,j} : [0, 1] \rightarrow \mathbb{R}^2$ is a polygonal curve of length 1). Analogous to [1], we glue the corresponding free-space diagrams of any two edges in $E$ that share a common vertex, along the boundary of the diagram that corresponds to this vertex. Doing so for all edges $e_{i,j} \in E$ yields the free space surface $S$.

As in [1], and similarly to the decision algorithm described in the preliminaries section for SDW of two curves, we get that there exists a path $P$ in $G$ with $SDW(r, P) \geq \delta$ if and only if there exists a $\gamma$-monotone path through the $\delta$-free space in $S$ between a point $(0, s)$ in some $F^i_j$, to a point $(\ell, t)$ in some $F^{i'}_{j'}$. Such a path correspond to a continuous surjective traversal of $r$ from $r(0)$ to $r(\ell)$, and some continuous non-surjective traversal $P$ of $G$ that starts on a point $s$ on the edge $e_{i,j}$ and ends on a point $t$ on $e'_{i',j'}$. By similar arguments, such a path can be found in $O(\ell m)$ time by computing the reachability graph of the diagram.

**SDW of a Graph:** We now move to defining the social distance width of a graph, analogous to Section 4, where we defined it for a polygon. A curve on a geometric graph corresponding to a surjective (non-surjective) map will be called a traversal (partial traversal) of the graph. We note that traversals may need to backtrack, and we allow partial traversals to do so too.

Define, for two geometric graphs $H$ and $G$, $SDW(H, G) = \sup_{h, g} \min_{t \in [0, 1]} d(h(t), g(t))$, where $h$ is a traversal of $H$ and $g$ is a partial traversal of $G$. The setting is that Red is going about its business on $H$, and must traverse it completely in some order, while Blue is only trying to stay away, and is restricted to be on $G$. Finally, we define for a graph $G$, its social distance width, as $SDW(G) = SDW(G, G)$ where again, we are free to choose either the Euclidean or the geodesic version.

**Theorem 24.** For two geometric graphs $H$ and $G$, the Euclidean $SDW(H, G)$ can be computed in time $O(m(H)m(G))$, where $m(\cdot)$ denotes the number of edges in the graph.

**Proof.** It is enough to show how to compute $SDW(H, G)$; $SDW(G)$ is then obtained by putting $H = G$. To this end, we construct the free space surface in a manner similar to that of the Red-on-a-mission case described previously. For every edge $e = (u, v) \in H$ and $f = (x, y) \in G$, we construct a free space cell $C_{e,f}$ which can be thought of as a subset of $[0, 1]^2$. For edges $e$ and $e'$ sharing a vertex $v$, we glue $C_{e,f}$ and $C_{e',f}$ along their right and left edges, respectively, which correspond to $C_{e,f}$. In this way we obtain a cell complex in three dimensions, with faces corresponding to cells $C_{e,f}$, edges corresponding to $C_{u,f}$ (or $C_{v,f}$, $C_{e,x}$, or $C_{e,y}$) and vertices corresponding to $C_{u,x}$, etc. The rest of the proof follows the lines of [1].

The above result says that when $d$ denotes the Euclidean distance, $SDW(H, G)$ can be computed in time $O(m(H)m(G))$. The following observation shows that even the geodesic case behaves nicely, as the free space is convex.
How to Stay Socially Distant: A Geometric Approach

![Figure 12](image)

Red is at c while Blue moves between a and b. Left: A star. Right: An arbitrary tree.

Lemma 25. For any \( L > 0 \), the free space (defined to be the points for which the geodesic distance is at least \( L \)) inside a cell of the free-space diagram is a convex polygon.

Proof. Consider 2 edges, \( e \) and \( f \) in an embedded PSLG, with coordinates \( x \) on \( e \), \( y \) on \( f \).

Then, the geodesic distance between a point at position \( x \) on \( e \) and a point at position \( y \) on \( f \) is simply

\[
d(x, y) = \min(x + d_2 + y, x + d_1 + |f| - y, |e| - x + d_4 + y, |e| - x + d_3 + |f| - y),
\]

which is a min of four functions that are linear in \((x, y)\). Thus, \( d(x, y) \) is concave, piecewise-linear, and the locus of points \((x, y)\) for which \( d(x, y) \geq L \) is a convex polygon, for any \( L \).

As a result, we obtain an algorithm with the same running time for geodesic distance. However, although we do not have a (conditional) quadratic lower bound for the computation for the Euclidean or the geodesic \( SDW(G) \), our next result states that a quadratic lower bound is unlikely for all graphs, even in the geodesic setting (which is slightly harder than the Euclidean version, hence a quadratic lower bound for it is more likely). Specifically, we prove that if \( G \) is a tree, its geodesic \( SDW \) can be computed in linear time!

Computing Geodesic SDW of a Tree in \( O(n) \) time

This section presents a linear-time algorithm for the following version: 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 \(|uv|\)). Both Red and Blue move around \( T \) in the same direction in a depth-first fashion: there is no start and end point, they keep 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 infinitesimally thin simple polygon).

We start with the case when \( T \) is a star (Fig. 12, 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 in a Blue is in \( c \), when Red is in \( b \) Blue moves to \( a \), when Red is in \( c \) Blue moves to \( a \); 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 because Blue cannot sit at \( a \) or at \( b \) all the time, and while it moves from \( a \) to \( b \) through \( r \), Red must be somewhere else (not at \( a \) or \( b \)).
We now consider an arbitrary tree $T$. Let $r \in T$ be a vertex. Removal of $r$ disconnects $T$ into several trees; for a vertex $v \neq r$ of $T$ let $T_v$ be the tree 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 $T \setminus T_a \setminus T_b$ furthest from $r$ (Fig. 12, 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 $|rc|$ 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 $c$.

To find $r$ in linear time, note that $ab$ is a diameter of $T$ because it is the longest simple path in the tree. All diameters of a tree intersect because if two diameters $uv, u'v'$ do not intersect, then there exist vertices $w \in uv, w' \in u'v'$ 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. 13, left). Moreover, since the tree has no cycles, the intersection of all its diameters is a path $\pi$ in $T$. We claim that $r$ may be found on $\pi$. Indeed, the distance from $\pi$ to any diameter endpoint is the same (can it $d$), so if there is more than one diameter, the 2-outlier radius of an endpoint of $\pi$ is $d$, while the 2-outlier radius of a vertex outside $\pi$ is at most $d$ (Fig. 13, right).

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 $T \setminus T_a \setminus T_b$ 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. The solution extends directly to weighted trees and to the version in which the distance is measured between arbitrary points on edges of $T$ (in this version, the 2-outlier center may lie in the middle of an edge).

6 Conclusion and Open Problems:

We have defined a notion of social distancing width of geometric objects, and we study the complexity of computing it in various settings. We give exact quadratic algorithms, conditional quadratic hardness, and approximation hardness for the case when two agents are on curves, in a polygon, or in graphs.

There are several versions of the problem for two agents that still remain open, notably the existence of a sub-quadratic time approximation algorithm with a factor slightly worse than what our lower bounds state, or of better hardness of approximation factors. Finally, we hope that this article will motivate a study into computing social distancing width in the more general setting, with $k > 2$-agents.
References

1. Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.
2. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995. doi:10.1142/S0218195995000064.
3. Esther M. Arkin, Joseph S. B. Mitchell, and Valentin Polishchuk. Maximum thick paths in static and dynamic environments. *Comput. Geom.*, 43(3):279–294, 2010. doi:10.1016/j.comgeo.2009.02.007.
4. Christoph Baur and Sándor P. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001. doi:10.1007/s00453-001-0022-x.
5. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly sub-quadratic algorithms unless SETH fails. In *Proc. of the 55th IEEE Annual Symposium on Foundations of Computer Science FOCS*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.
6. Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *JoCG*, 7(2):46–76, 2016. doi:10.20382/jocg.v7i2a4.
7. Kevin Buchin, Tim Ophelders, and Bettina Speckmann. 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. doi:10.1137/1.9781611975482.179.
8. Timothy M. Chan and Zahed Rahmati. An improved approximation algorithm for the discrete Fréchet distance. *Inf. Process. Lett.*, 138:72–74, 2018. doi:10.1016/j.ipl.2018.06.011.
9. Timothy H. Chung, Geoffrey A. Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics - A survey. *Auton. Robots*, 31(4):299–316, 2011. doi:10.1007/s10514-011-9241-4.
10. Connor Colombe and Kyle Fox. Approximating the (continuous) Fréchet distance. arXiv preprint arXiv:2007.07994, 2020.
11. Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019. doi:10.1137/18M1194341.
12. Adrian Dumitrescu and Minghui Jiang. On reconfiguration of disks in the plane and related problems. *Comput. Geom.*, 46(3):191–202, 2013. doi:10.1016/j.comgeo.2012.06.001.
13. Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical report, Citeseer, 1994.
14. Sándor P. Fekete and Henk Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38(3):501–511, 2004. doi:10.1007/s00453-003-1074-x.
15. Leonidas J Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.
16. Shai Hirsch and Dan Halperin. 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. doi:10.1007/978-3-540-45058-0_15.
17. Tien-Ruey Hsiang, Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, and Joseph S. B. Mitchell. 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. doi:10.1007/978-3-540-45058-0_6.
18. Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989.
Michail I Schlesinger, EV Vodolazkiy, and VM Yakovenko. Similarity of closed polygonal curves in Frechet metric. \textit{arXiv preprint arXiv:1409.4615}, 2014.