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 Replace ccsdesc macro with valid one

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

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: [funding]
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.

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*

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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. We begin by considering the scenario in which the two domains are polygonal curves R and B of complexity m and n, respectively.[1] The agents' missions are to traverse their respective curves, from the first point to the last point of the curve, without backtracking (can we say anything about case with backtracking?), 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 where 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, where we are usually given a set of disks, rectangles, or other fat geometric shapes that represent agents or robots, and the goal is to find a valid sequence of movements that allows them to get from their initial configuration to some given final 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 in a DFS-like ordering, 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:

- 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

[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 $m \neq n$.

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 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 [14, 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 α-approximation, running in $O((n^3/\alpha^2)\log n)$ time [11], and in $O(n \log n + n^2/\alpha^2)$ time for the discrete version [9].

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. [12] on nearly optimal (in terms of lengths of motions) rearrangements of multiple unit disks; see also the related work of [13, 17].

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 (the number of thick paths).

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, 15]. (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. This movement into a dispersed state can be accomplished using simple local movement strategies that are competitive (see, e.g., [18]).

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 [10].

2 Preliminaries

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 $\alpha, \beta \in [1, n]$, $\alpha \leq \beta$, 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 $P : [1, n] \to \mathbb{R}^d$ and $Q : [1, n] \to \mathbb{R}^d$ be two polygonal curves. 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 $f : [0, 1] \to [1, n]$ and $g : [0, 1] \to [1, n]$ are continuous, non-decreasing, surjections.

The standard *(continuous) Fréchet distance* between P and Q is defined to be

$$d_{dF}(P, Q) = \inf_{f,g} \max_{t \in [0,1]} \|P(f(t)) - Q(g(t))\|.$$

The *δ-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 δ. We adapt this notion to our new setting. For every two integers $1 \leq i, j \leq n-1$, let $\mathcal{C}_{ij} = [i, i+1] \times [j, j+1]$ be a unit square in the plane. Denote by $\mathcal{B} = [1, n] \times [1, n]$ the square in the plane that is the union $\cup_{i,j \in \{1,\ldots,n-1\}} \mathcal{C}_{ij}$. Given $\delta > 0$, the *δ-free space* is $\mathcal{F}_\delta = \{(p, q) \in \mathcal{B} \mid \|P(p) - Q(q)\| \geq \delta\}$. In other words, it is the set of all red-blue positions for which the distance between the agents is at least 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 $\mathcal{B} \setminus \mathcal{F}_\delta$. Notice that for Fréchet distance, the definitions are reversed ("flipped"). We call the rectangles $\mathcal{C}_{ij}$ 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.

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 $P[1], P[2], \ldots, P[n]$ the vertices (points) 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* τ 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 *discrete Fréchet distance* (DFD) between P and Q is defined to be

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

We define the *discrete social distance width* (dSDW) of P and Q to be

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

3 Social distancing on polygonal paths

In this section the domains of Red and Blue are two polygonal paths R and B resp., and the question is whether they can traverse their paths while maintaining the distance at least 1.

We give quadratic lower bounds on approximating $dSDW$ and SDW for 2D paths (the factors are different in both cases), conditioned on a version of the Strong Exponential Time Hypothesis (SETH). We then give the same quadratic conditional lower bound for exact $dSDW$ even for 1D paths. Finally, we give a positive result: deciding if $SDW(R, B) \geq 1$ for 1d paths can be done in near-linear time (this is in contrast to the usual Fréchet distance, for which the bound holds even for 1d paths, both for continuous and discrete versions [6, 7]). We also outline how to use free-space diagrams to obtain exact algorithms with quadratic running time for computing $dSDW$ and SDW in any dimension.

Mayank: where? In appendix? If so, cite.

3.1 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 φ 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 exits no $O^*((2-\delta)^N)$-time algorithm to decide if φ 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 α 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 1.** *[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 at least* $(3-\sqrt{3})/2 \approx 0.6339$*, and runs in time* $O(n^{2-\delta})$ *time for any* $\delta > 0$*, unless SETH' (and hence SETH) fails.*

We sketch the proof of Theorem 1, with full proof in full version in appendix. We construct paths R, B such that the discrete $SDW(R,B) \geq 1$ iff φ 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 $O(M^{2-\delta}2^{N(1-\delta/2)})$ 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 ith 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 φ let $\mathrm{sat}(\sigma, C)$ be T or F depending on whether σ satisfies C or not. Choose any $0 < \varepsilon < (\sqrt{3}-1)/2$ and set the following points: $c_T^0 = (0, \frac{1}{2}+\varepsilon), c_F^0 = (0, \frac{1}{2}-\varepsilon), c_T^1 = (0, -\frac{1}{2}-\varepsilon), c_F^1 = (0, -\frac{1}{2}+\varepsilon), b = (1/2, 0), r = (-1/2, 0)$ (Fig. 1). Observe that $d(c_T^0, b)) = d(c_T^1, b)) < 1$ (i.e., both are within distance one of b). By symmetry, the same is true for b replaced by r above.

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

$B_{\sigma_1} = b \circ \bigcirc_{i=1\ldots M} c^{i \bmod 2}_{\mathrm{sat}(\sigma_1, C_i)}$, and for any assignment $\sigma_2 \in A_2$ construct the gadget

$R_{\sigma_2} = r \circ \bigcirc_{i=1\ldots M} c^{(i+1) \bmod 2}_{\mathrm{sat}(\sigma_2, C_i)}$. See Figure 1 for the gadgets. One can then prove:

▶ **Lemma 2.** *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* (σ_1, σ_2) *satisfies* φ*, then* $SDW(B_{\sigma_1}, R_{\sigma_2}) \geq 1$.
3. *If* (σ_1, σ_2) *does not satisfy* φ*, then* $SDW(B_{\sigma_1}, R_{\sigma_2}) \leq 1-\varepsilon$*, and for any two curves* P, Q *it holds that* $SDW(B_{\sigma_1} \circ P, R_{\sigma_2} \circ Q) \leq 1-\varepsilon$.

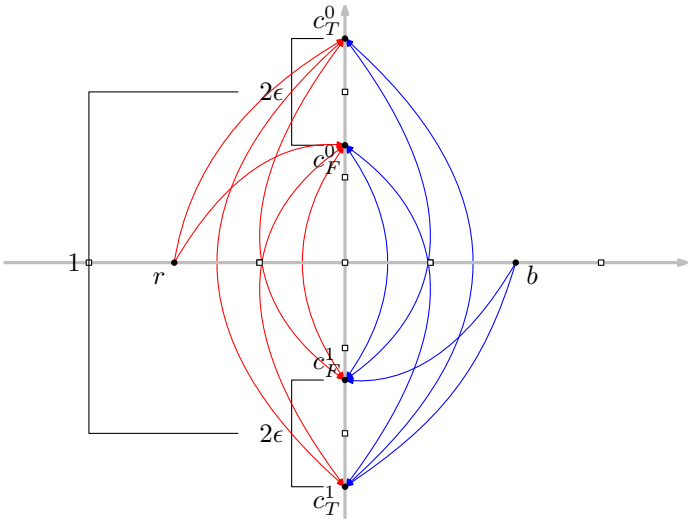

Figure 1 The assignment gadgets. The blue edges represent B_{σ_1} and the red edges represent R_{σ_2} for some $\sigma_1 \in A_1$, $\sigma_2 \in A_2$

We now add few more points and connect the assignment gadgets to form the paths R and B. Let $\beta \approx -0.0504897$ be the negative solution[2] to the quadratic equation $x^2 - \sqrt{3}x - 0.09 = 0$. set $x = (-1, 0)$, $s_1 = (0.994, 0.755)$, $t_1 = (0.994, -0.755)$, $s_2 = (0.4, \beta)$, $t_2 = (0.4, -\beta)$, 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. 2). The following lemma can be verified by computing distances.

[2] β is chosen so that $c_T^0 = (0, 1/2 + \varepsilon)$ is exactly distance 1 to the point $(0.4, \beta)$ when $\varepsilon = (\sqrt{3} - 1)/2$, and they are less than 1 apart if ε is smaller.

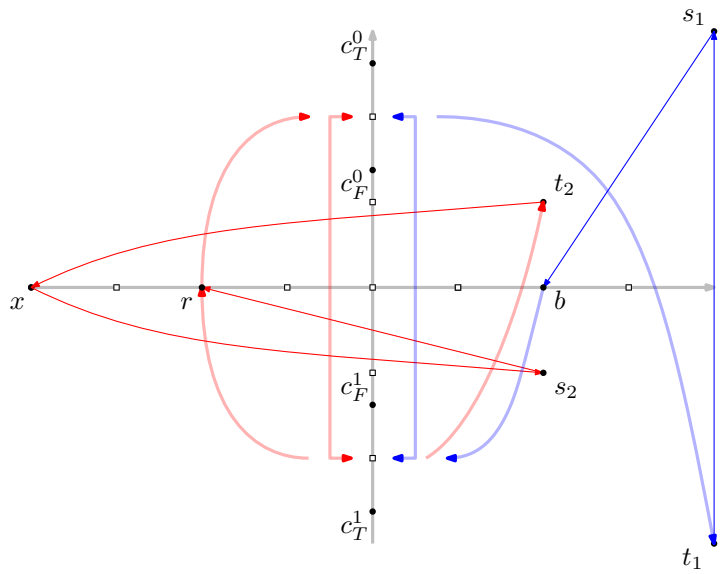

Figure 2 The curves R and B.

▶ **Lemma 3.** *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_T^0 *and* c_T^1*.*

▶ **Lemma 4.** φ *is satisfiable if, and only if, the discrete* $SDW(R, B) \geq 1$*.* ϕ *is not-satisfiable if, and only if, the discrete* $SDW(R, B) \leq 1 - \epsilon$

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 φ, 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 $s_2 \circ \bigcirc_{1 \leq k \leq j-1}(R_{\sigma_k})$. Since the number of clauses is even, this (partial) traversal ends at either (s_1, c_F^1) or (s_1, c_T^1). The next step matches b to r. The traversal then matches B_{σ_i} and R_{σ_j} in a "parallel" fashion (Lemma 2), then matches t_1 to the last vertex of R_{σ_j}, stays at t_1 and matches t_1 to $\bigcirc_{j+1 \leq k \leq |A_2|}(R_{\sigma_k}) \circ t_2 \circ x$, and finally matches $\bigcirc_{i+1 \leq k | A_1|}(s_1 \circ B_{\sigma_k} \circ t_1)$ to x. One can check that this traversal keeps Red and Blue separated by at least 1.

Conversely, assume that there is a traversal $\tau = ((i_1 := s_1, j_1 := x), \ldots, (i_T := s_1, j_T := x))$ that has SDW at least 1, and let $\alpha = \max\{t : j_t = s_2\}, \beta = \min\{t : j_t = t_2\}$. Note that $\alpha < \beta$ (Red hits s_2 before t_2), and that $i_\alpha = s_1, i_\beta = t_1$ by (Lemma 3,3). 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 3), and $x \neq i_\gamma$ for any $\alpha < \gamma < \beta$. In words, we have a time γ 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 σ_1 for Blue and σ_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 γ: 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 2, this is possible only if (σ_1, σ_2) satisfies φ. The assertion about ϕ being not-satisfiable iff $SDW(P, B) \leq 1 - \varepsilon$ now simply follows from Property 3 of Lemma 2.

Approximation hardness: According to Lemma 4, any algorithm with approximation factor strictly better than $1 - \varepsilon$ will be able to distinguish between the cases $SDW \geq 1$ and $SDW \leq 1 - \varepsilon$. Since ε is an arbitrary number less than $(\sqrt{3} - 1)/2$, this gives the claimed factor of $(3 - \sqrt{3})/2 \approx 0.6339$.

We now prove a similar lower bound for the *continuous* SDW in 2D.

▶ **Theorem 5.** *[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 , and runs in time $O(n^{2-\delta})$ time for any $\delta > 0$, unless SETH' (and hence SETH) fails.*

Mayank: Enter correct factor here

We sketch the proof of Theorem 5, with full proof in full version in appendix. For the discrete SDW it did not matter how exactly the vertices of R and of B were connected into the paths (the distancing had to be ensured only at the vertices). To show the lower bound for continuous, we modify the construction and connect the vertices so that Red and Blue stay separated also while moving between the vertices (Fig. **??**). Specifically, imagine the circle of diameter 1 centered on the origin, and let it be part of all assignment gadgets (both Red and Blue): now Red and Blue can move between odd and even clauses along the circle, opposite each other. However, a circle is technically not allowed, and to keep R and B polygonal, the circle is replaced by regular octagon circumscribed around the circle – this creates additional problems as red goes through r both in between clauses and in between assignments, which we remedy by placing two additional curves by passing parts of the octagon. See Figure 3, and the full proof in appendix for details. One then needs to show that Red and Blue can still move between the clauses, and the only way to do it is to move (almost) opposite each other. The bound on approximation hardness involves two cases but is derived similarly by finding the SDW of a traversal corresponding to non-satisfying assignments for both Red and Blue. See Figure 3 for the gadgets.

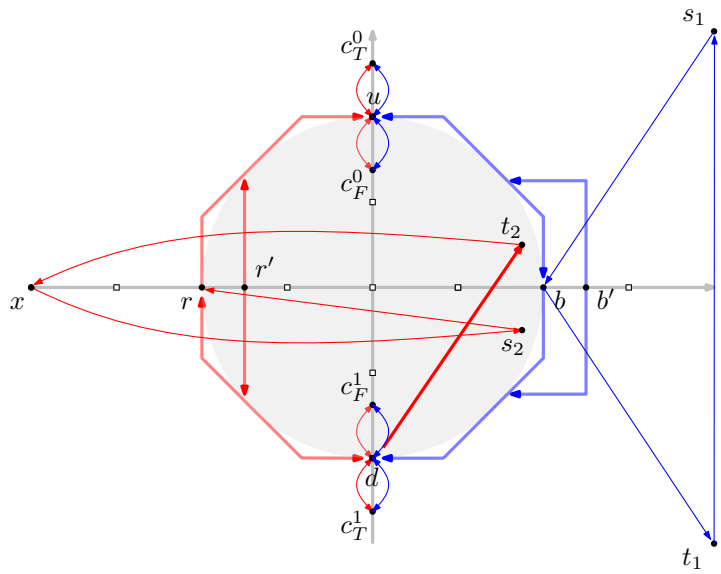

Figure 3 The curves R and B.

3.2 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 sticks to being hard, i.e., no subquadratic algorithms, the continuous version gives way and is solvable in near-linear time! We first state our lower bound for the discrete case, and then our linear time 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, the full proof of which is in the appendix.

▶ **Theorem 6.** *[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.*

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

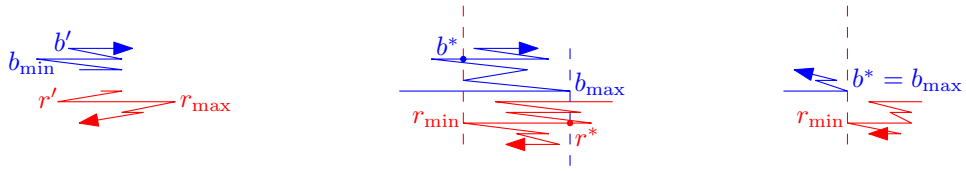

Figure 4 Left: Blue and Red may go from the start to $(b_{\min}, r')$ to $(b_{\min}.r_{\max})$ to $(b', r_{\max})$ to the end. Middle: r^* must be visited before $b_{\max}$, and b^* before $r_{\min}$. Right: $x = b_{\max} = b^* = r_{\min}$ separates B and R

▶ **Lemma 7.** *Suppose $SDW(B, R) > 0$. While Blue is at $b_{\min}$, any subpath of R can be traversed by Red. While Red is at $r_{\max}$, any subpath of B can be traversed by Blue.*

▶ **Lemma 8.** *If $SDW(B, R) > 0$, then there exists a non-crossing traversal such that at some point Blue is at $b_{\min}$ and Red is at $r_{\max}$.*

Proof. Consider a non-crossing traversal τ, and assume that Blue reaches $b_{\min}$ before Red reached $r_{\max}$. Let r' be the location of Red when Blue is at $b_{\min}$; r' does not have to a be a vertex of R, but in any case r' precedes $r_{\max}$ along R (Fig. 4). Let $b' \in B$ be Blue's location at the time Read reaches $r_{\max}$ (b' is after $b_{\min}$). By Lemma 7, while Blue is at $b_{\min}$, Red can go from r' to $r_{\max}$ – so $(b_{\min}, r_{\max})$ becomes part of the traversal. Again by Lemma 7, while Red is at $r_{\max}$, Blue can go from $b_{\min}$ to b'. From $(b', r_{\max})$, Blue and Red can follow τ to complete the traversal. ◀

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

Let $b_{\max}$ 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 $r_{\min}$ be the leftmost point of R and let b^* be the last point of B intersecting the vertical line $x = r_{\min}$ (Fig. 4, middle). We consider different cases of how $b^*, b_{\max}$ and $r^*, r_{\min}$ are located along B and R resp.

If $b_{\max} = b^*$ (Fig. 4, right), then since b^* coincides with $r_{\min}$, we have that B is to the left of the common abscissa of $b_{\max}, b^*$ and $r_{\min}$, while R is to the right. Hence, any traversal will be non-crossing (mod the trivial case when $b_{\max}$ and $r_{\min}$ are the endpoints of their paths). Similarly, we are done if $r_{\min} = r^*$. In what follows we treat the cases when $b_{\max} \neq b^*, r_{\min} \neq r^*$.

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

▶ **Lemma 9.** *If $b^* < b_{\max}, r^* < r_{\min}$, then there is no non-crossing traversal.*

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

We are thus left with 3 cases: (1) $b^* > b_{\max}, r^* > r_{\min}$, (2) $b^* > b_{\max}, r^* < r_{\min}$, (3) $b^* < b_{\max}, r^* > r_{\min}$. Cases (2) and (3) are symmetric, so assume w.l.o.g. that $b^* > b_{\max}$. By our assumption that B and R do not have coinciding vertices, b^* is not a vertex (since $r_{\min}$ 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 8) $B[1]$ is the leftmost point of B, by Lemma 7, 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 9) or done.

For the runtime, our recursive solution involves answering semi-dynamic maximum/minimum queries to obtain $b_{\max}/r_{\min}$ (the queries are semi-dynamic because the paths only shorten), and semi-dynamic queries for the first time of reaching $b_{\max}/r_{\min}$ (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. 4). 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 10.** *[Continuous 1D Algorithm] There exists an alorithm that computes the continuous SDW of two curves in 1D in $O(n\text{polylog}\, n)$ time.*

4 Social distancing on a 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 11.** *Assume that Red moves along the geodesic path π 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.*

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.

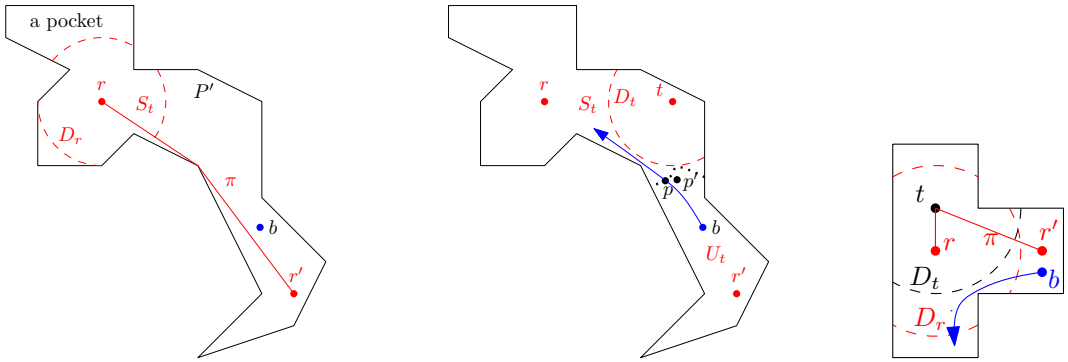

Figure 5 Left: D_r splits P into pockets. Middle: Blue escaping Red; the boundary between S_t and U_t is dotted. Right: Blue escapes when Red is at t

We use $|ab|$ to denote the geodesic distance between points $a, b \in P$. For a point $t \in \pi$ let $D_t = \{p \in P : |tp| \leq 1\}$ be the unit geodesic disc centered on t; let $M = \cup_{t \in \pi} D_t$ be the set of points within geodesic distance 1 from π (Fig. 5, left).

Without loss of generality, assume $b \notin D_r$ (for otherwise Blue is doomed 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). The next lemma asserts that the existence of such a safe point is also *necessary* for Blue to maintain the distance of 1. The full proof in the appendix relies on 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].

▶ **Lemma 12.** *If $P' = M \setminus D_r$, i.e, if Blue does not have the possibility to move to a safe point while Red is at r, then there is no traversal for Blue that maintains a social distance of 1 from Red.*

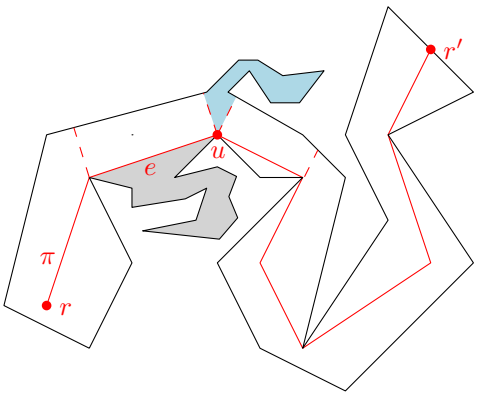

Figure 6 SPMs from edges and vertices of π is computed separately in parts of P defined by perpendiculars to path edges (some shown dashed). Gray and lightblue parts are charged to (the right side of) the edge e and to vertex u of π resp.

Finally we show how to implement our solution 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 [16]; 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 π. To build the SPM, we decompose P by drawing perpendiculars to the edges of π at every vertex of the path (Fig. 6): 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 π 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 [16] 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.

4.2 The Social Distance Width of a polygon

Imagine that Red is a guard patrolling the boundary of P with some constant speed, i.e., Red walks along ∂P, without any restriction on the number of rounds, which may be infinite. Blue has unlimited speed, and has to stay as far away from Red as possible. Notice that the movement strategy for Blue is not necessarily repetitive; in other words, when Red completes one round of ∂P and returns to s_r, it might be that Blue is on a point $x \neq s_b$, and cannot reach s_b, while Red is on s_r. In the next round of Red, Blue uses a different strategy, as the starting point changes. Therefore, the movement strategy of blue might not have a finite description. In the following lemma, we actually show that for a given δ, we can always compute a "strategy manual" for Blue of size $O(n^2)$, independent of the number of rounds that Red does!

We define the Social Distance Width (SDW) of P, denoted SDW(P), to be the maximum δ that allows Blue to avoid Red. Notice that $SDW(P) \leq width(P)$ and

TODO:Mayank help out here...

▶ **Theorem 13.** *Given a value δ, if there exists a strategy for Blue to maintain distance at least δ from Red, when Red is patrolling ∂P for m rounds, then such a strategy can be computed in $O(n^2)$ time.*

In order to prove the above lemma, 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 $\delta > 0$. 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 7). 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 δ at any point in time.

Notice that there exists a strategy for Blue to maintain distance at least δ from Red, if and only if there exists a sequence of y-monotone paths $\Pi_1, \ldots, \Pi_k, \ldots$ in $\mathcal{F}_\delta$ such that Π_1 start 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 $\mathcal{F}_\delta$ (Π_1 might move through the bottom boundary to the top boundary and back), then Π_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 Π_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 7).

Observe that if one of the paths Π_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 Π_i and Π_j for $i < j$ are crossing, then we could find a shorter sequence of paths by going directly from the entry point of Π_i to the exist point of Π_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.

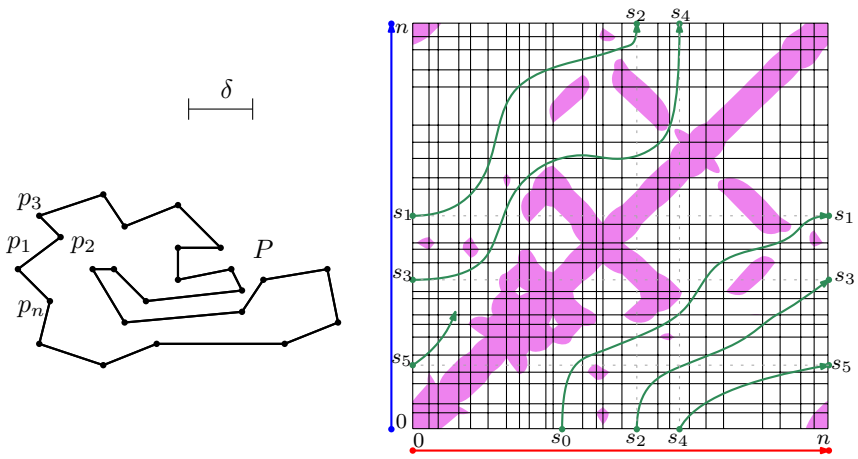

Figure 7 The polygon P and the free space diagram.

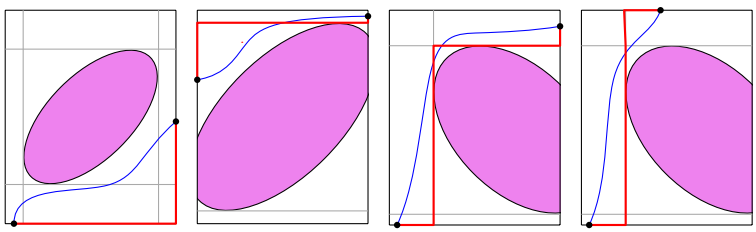

Figure 8 Examples of a free space cell $C_{i,j}$, and the set of forbidden points in pink. The blue path can be replaced by the red.

Consider a cell $\mathcal{C}_{ij}$ of $\mathcal{F}_\delta$, and recall that the set $\mathcal{C}_{ij} \setminus \mathcal{F}_\delta$ of forbidden points is a convex shape. For each cell C_{ij} we construct the four orthogonal tangents to $\mathcal{C}_{ij} \setminus \mathcal{F}_\delta$, and extend them horizontally and vertically until hitting a forbidden point. Roughly speaking, notice that a path through the free-space of $\mathcal{C}_{ij}$ can always avoid entering the bounding box of the free space $\mathcal{C}_{ij} \cap \mathcal{F}_\delta$ (see Fig. 8). Now consider the arrangement of segments that constitute the set of all tangents and cell boundaries in $\mathcal{F}_\delta$. 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. The vertical edges are bi-directional, and the horizontal edges are directed from left to right. The details for the following lemma can be found in the appendix.

▶ **Lemma 14.** *For any path Π through $\mathcal{F}_\delta$ that starts at a point $(s, 0)$ and ends at a point $(t, 0)$, both on the boundary of $\mathcal{F}_\delta$, there exists a path Π' in G between two vertices $(s', 0)$ and*

$(t', 0)$, *such that* $s < s'$, $t < t'$, $(s, 0)$ *is on an edge of* G *adjacent to* $(s', 0)$, *and* $(t, 0)$ *is on an edge of* G *adjacent to* $(t', 0)$. *Moreover, if* Π *is* y*-monotone then* Π' *is* y*-monotone.*

As we get a "discrete" description of the diagram, which has size $O(n^2)$, Theorem 13 now follows from Lemma 14 and the observations above. More details are in the full version.

5 Graphs

In this section we consider social distancing measures for graphs[3]. An undirected graph $G = (\mathcal{V}, \mathcal{E})$ 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 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 $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. By a path of length k in $G = (\mathcal{V}, \mathcal{E})$ we mean a sequence P of k vertices: $P = \{p_1, p_2, \cdots, p_k\} \subseteq \mathcal{V}$, with edges $p_i p_{i+1} \in E$ 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. A curve on a geometric graph corresponding to a surjective (non-surjective) map will be called a traversal (partial traversal) of the graph.

Blue distancing from Red on a mission

▶ **Theorem 15** (Unweighted graph). *In unweighted graphs, 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* $O(nkd^s)$, *where* d *is the maximum degree of any vertex in* G.

SDW between two curves in a graph

SDW between curves on graphs Given two paths P and Q in an unweighted graph, one can define the social distance width of these paths by $SDW(P, Q) = \max_\tau \min_{(i,j)\in\tau} d(P(i), Q(j))$, where τ is a non-backtracking traversal of P and Q (different from the usual traversal on a single graph), defined similarly as for the discrete SDW in Section 2. Similarly, define the $SDW(P, Q)$ for two curves in a geometric graph analogous to the definition for continuous SDW in Section 2, except that the distances are now measured in the graph.

SDW of a Graph Define, for two geometric graphs H and G,

$$SDW(H, G) = \sup_{h,g} \min_{t\in[0,1]} d(h(t), g(t))||.$$

We remark that we deviate from the Euclidean versions of the Frechet distance between graphs used in [1] and [8]. The reasons are twofold: given our setting we feel it is more realistic to consider geodesic distances, and most "nice" observations for Euclidean Frechet do not apply when considering the geodesic SDW: e.g.for SDW, it may make sense for Blue to go on a non-simple path to stay away from red, whereas this is not the case for Frechet distance defined for graphs in [8].

[3] We focus on undirected graphs, but our results readily extend to directed graphs

5.1 Unweighted Graphs

Assume two agents, Red and Blue, have their motion restricted to an arbitrary (not necessarily geometric) graph G. If Red does not announce its path, and Blue has no knowledge of where Red is, it can do very little. Even if Blue can "see red coming," but Red is allowed to adapt, Blue may not be able to maintain distance from Red (e.g., if the graph is a tree, then Red goes to the subtree containing Blue and all Blue can do is watch Red approach, bringing the infection). This adversarial setting is related to the family of "lion and man" games, which we do not address here. We will focus on the case in which Red announces its path, does not care about social distancing, and Blue is trying to stay away from Red. First, we consider motion in an unweighted ($w_e = 1$), undirected graph G.

Problem: Red announces its path P of length k. Blue starts at a point b (which could be given, or Blue may be free to choose), and can travel s edges in the time it takes for Red to travel one edge[4], where $1 \leq s \leq n-1$. Can Blue maintain a distance of $\delta > 0$ away from Red?

5.2 Geometric Graphs

While a continuous traversal does not make sense in an arbitrary graph, our motivating application suggests the study of the problem in geometric graphs. A geometric graph $G = (\mathcal{V}, \mathcal{E})$ is an undirected, connected, planar graph with straight line edges. With $n = |\mathcal{V}|$, we know that $|\mathcal{E}| = O(n)$. We consider two problems: (1) Red is on a mission and Blue seeks to travel within G in order to maintain social distance from Red, and (2) we seek to compute the social distance width of a geometric graph G. The following setting is motivated by [1].

Red on a mission: Let $r : [0, \ell] \to \mathbb{R}^2$ be a polygonal curve in $\mathbb{R}^2$, consisting of ℓ line segments, with the ith 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 where 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 . Note that by adopting this definition we are considering the Euclidean distance between Red and 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.

Mayank: refer to appropriate section

Mayank: remark later

We show (in the APpendix) that the decision problem above can be solved in $O(\ell m)$ time, where ℓ is the length of Red's path and m is the number of edges in the graph G where Blue is restricted to.

Mayank: Next two paragraphs probably going to the appendix

We first define the free space surface for our problem. To this end, we assume any edge $e = (i, j) \in \mathcal{E}$ is parameterized by the unit interval, i.e., $\pi_{i,j} : [0, 1] \to \mathbb{R}^2$ is continuous and injective with range equal to e. Given the path r of Red, we define the free space of e as $F_{i,j} := \{(x, y) \in [0, \ell] \times [0, 1] : d(r(x), \pi(y)) \geq \delta\}$. Analogous to [1], we define the free space diagram $FD_{i,j}$ as the division of $[0, \ell] \times [0, 1]$ into $F_{i,j}$ (the feasible region) and its complement (the infeasible region). We can break up $FD_{i,j}$ along the x-axis into ℓ cells, each corresponding to an edge of r, and define FD_i (FD_j) as $F_{ij} \cap ([0, \ell] \times \{0\})$ ($F_{ij} \cap ([0, \ell] \times \{1\})$). Unlike the usual free space diagrams, however, the *infeasible region is convex, whereas the*

[4] One can assume that Blue travels on a simple path when Red is traversing an edge.

feasible region is the complement of the convex feasible set and is therefore non-convex. We also replace every undirected edge $(i, j) \in \mathcal{E}$ with two directed edges, and therefore obtain $FD_{i,j}$ and $FD_{j,i}$, which have FD_i as their bottom and top edges, respectively. We glue these diagrams along the corresponding edges; doing so for all edges $(i, j) \in \mathcal{E}$ yields the free space surface.

We now solve a reachability problem on this free surface.

Mayank: Omrit, please...

Social Distance Width between graphs

Let H and G be geometric graphs, with $m(H)$ and $m(G)$ edges, respectively. We now define a social distancing measure between H and G. Comparing geometric graphs has been done before; most relevant to us are the works of [1] and [8]. It turns out that for technical reasons, the right definition for us is more along the lines of [1].

A *traversal* of H is a map $h : [0, 1] \to H$ that is surjective and continuous. A continuous but not necessarily surjective map $g : [0, 1] \to G$ will be called a *partial traversal* of G.

We prove (in the Appendix) that $SDW(H, G)$ can be computed in time $O(m(H)m(G))$ and hence that $SDW(G)$ can be computed in time $O(m(G)^2)$.

Mayank: Next paragraph goes in appendix

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 $C_{v,f}$. 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. We now observe that the free space inside a cell is the complement of a convex set.

Mayank: Omrit, again please..

While we showed how to construct $SDW(G)$ in quadratic time, , we now show that if G is a tree, we can compute $GeoSDW(G)$ in linear time.

Mayank: and we gave an algorithm/reason on why $GeoSDW(G)$ also takes quadratic time

5.3 Computing SDW of a Tree

This section presents a linear-time algorithm for a non geometric (graph) version 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 $|uv|$). 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).

Mayank: It would be good if we can say that it does not help for red to backtrack. Or can it?

We start from the case when T is a star (Fig. 9, 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 us at most $|rc|$). Assume that the leaves a, b, c are encountered in this order as Red moves around T (the assumption is w.l.o.g., as the other order 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 $T_v \ni v$ 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

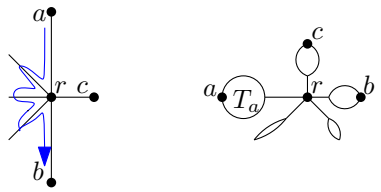

Figure 9 Red is at c while Blue moves between a and b. Left: A star. Right: An arbitrary tree.

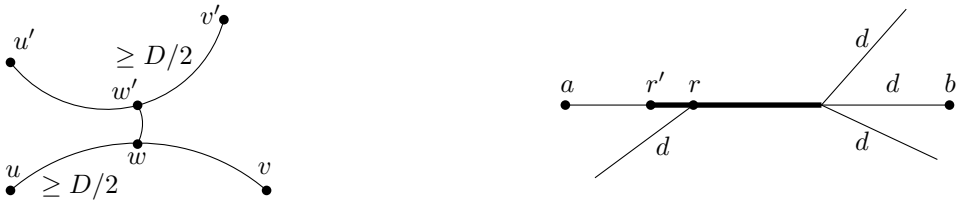

Figure 10 Left:: If $|uv| = |u'v'| = D$, then $|uww'v'| > D$. Right: The distance from an endpoint of π (thick) to any diameter endpoint is the same, for otherwise one of the diameters is longer than another. $ra = d$, and the 2-outlier radius of r is d, while the 2-outlier radius of r' is cannot exceeds d.

$T \setminus T_a \setminus T_b$ furthest from r (Fig. 9, 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 a 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. 10, left). Moreover, since the tree has no cycles, the intersection of all its diameters is a path π in T. We claim that r may be found on π. Indeed, the distance from π 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 π is d, while the 2-outlier radius of a vertex outside π is at most d (Fig. 10, 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).

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 subquadratic 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 Maike Buchin, Stef Sijben, and Carola Wenk. Distance measures for embedded graphs. In *Proc. 33rd European Workshop on Computational Geometry (EuroCG)*, pages 37–40, 2017.

9 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.

10 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.

11 Connor Colombe and Kyle Fox. Approximating the (continuous) fr\'echet distance. *arXiv preprint arXiv:2007.07994*, 2020.

12 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.

13 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.

14 Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. Technical report, Citeseer, 1994.

15 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.

16 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.

17 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.

18 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.

19 Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989.