Discrete Fréchet Distance under Translation: Conditional Hardness and an Improved Algorithm

KARL BRINGMANN, Saarland University and Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
MARVIN KÜNNEMANN, Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
ANDRÉ NUSSER, Max Planck Institute for Informatics and Graduate School of Computer Science, Saarland Informatics Campus, Germany

The discrete Fréchet distance is a popular measure for comparing polygonal curves. An important variant is the discrete Fréchet distance under translation, which enables detection of similar movement patterns in different spatial domains. For polygonal curves of length n in the plane, the fastest known algorithm runs in time $\tilde{O}(n^5)$ [12]. This is achieved by constructing an arrangement of disks of size $O(n^4)$, and then traversing its faces while updating reachability in a directed grid graph of size $N := O(n^2)$, which can be done in time $\tilde{O}(\sqrt{N})$ per update [27]. The contribution of this article is two-fold.

First, although it is an open problem to solve dynamic reachability in directed grid graphs faster than $\tilde{O}(\sqrt{N})$, we improve this part of the algorithm: We observe that an offline variant of dynamic s-t-reachability in directed grid graphs suffices, and we solve this variant in amortized time $\tilde{O}(N^{1/3})$ per update, resulting in an improved running time of $\tilde{O}(n^{4.66\cdots})$ for the discrete Fréchet distance under translation. Second, we provide evidence that constructing the arrangement of size $O(n^4)$ is necessary in the worst case by proving a conditional lower bound of $n^{4-o(1)}$ on the running time for the discrete Fréchet distance under translation, assuming the Strong Exponential Time Hypothesis.

CCS Concepts: • **Mathematics of computing → Paths and connectivity problems**; • **Theory of computation → Problems, reductions and completeness**; **Dynamic graph algorithms**; **Computational geometry;**

Additional Key Words and Phrases: Fréchet distance, conditional lower bounds

ACM Reference format:
Karl Bringmann, Marvin Künnemann, and André Nusser. 2021. Discrete Fréchet Distance under Translation: Conditional Hardness and an Improved Algorithm. *ACM Trans. Algorithms* 17, 3, Article 25 (July 2021), 42 pages.
https://doi.org/10.1145/3460656

Extended abstracts of this article appeared at SODA'19 [15] and https://arxiv.org/abs/1810.10982.
This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979).
Authors' addresses: K. Bringmann, Saarland University and Max Planck Institute for Informatics, Saarland Informatics Campus, Campus E1 3, Saarbrücken, 66123, Germany; email: kbringma@mpi-inf.mpg.de; M. Künnemann, Max Planck Institute for Informatics, Saarland Informatics Campus, Campus E1 4, Saarbrücken, 66123, Germany; email: marvin@mpiinf.mpg.de; A. Nusser, Max Planck Institute for Informatics and Graduate School of Computer Science, Saarland Informatics Campus, Campus E1 4, Saarbrücken, 66123, Germany; email: anusser@mpi-inf.mpg.de.

1549-6325/2021/07-ART25
https://doi.org/10.1145/3460656

1 INTRODUCTION

Fréchet distance. Modern tracking devices yield an abundance of movement data, e.g., in the form of GPS trajectories. This data is usually given as a sequence of points in $\mathbb{R}^d$ for some small dimension d like 2 or 3. By interpolating linearly between consecutive points, we obtain a corresponding polygonal curve. One of the most fundamental tasks on such objects is to measure similarity between two curves π, σ. A popular approach is to measure their distance using the Fréchet distance, which has two important variants: The classic *continuous Fréchet distance* is the minimal length of a leash connecting a dog and its owner as they continuously walk along the interpolated curves π and σ, respectively, from the startpoints to the endpoints without backtracking. In the *discrete Fréchet distance*, at any time step the dog and its owner must be at vertices of their curves and may jump to the next vertex. This discrete version is well motivated when we think of the inputs as sequences of points rather than polygonal curves, i.e., if the interpolated line segments between input points have no meaning in the underlying application. In comparison to other similarity measures such as the Hausdorff distance, the Fréchet distance considers the ordering of the vertices along the curves, thus reflecting an intuitive property of curve similarity.

The time complexity of the Fréchet distance is well understood. For the continuous Fréchet distance, Alt and Godau designed an $O(n^2 \log n)$-time algorithm for polygonal curves π, σ consisting of n vertices [6]. Buchin et al. [18] improved on this result by giving an algorithm that runs in time $O(n^2 \sqrt{\log n} (\log \log n)^{3/2})$ on the Real RAM and $O(n^2 (\log \log n)^2)$ on the Word RAM. The first algorithm for the discrete Fréchet distance ran in time $O(n^2)$ [31], which was later improved to $O(n^2 \frac{\log \log n}{\log n})$ [4]. On the hardness side, conditional on the Strong Exponential Time Hypothesis, Bringmann [14] ruled out $O(n^{2-\varepsilon})$-time algorithms for any $\varepsilon > 0$, for both variants of the Fréchet distance. Recently, Abboud and Bringmann [1] showed that any $O(n^2/\log^{17+\varepsilon} n)$-time algorithm for the discrete Fréchet distance would prove novel circuit lower bounds. On the practical side, several fast implementations for computing the continuous Fréchet distance resulted from the SIGSPATIAL GIS Cup 2017 [10, 20, 29] with a follow-up work significantly improving on these results [16].

Many extensions and variants of the Fréchet distance have been studied, e.g., generalizing from curves to other types of objects, replacing the ground space $\mathbb{R}^d$ by more complex spaces, and many more (see, e.g., References [5, 11, 19, 21, 26, 28, 38, 43]). Applications of the Fréchet distance range from moving objects analysis (see, e.g., Reference [17]) through map-matching tracking data (see, e.g., Reference [13]) to signature verification (see, e.g., Reference [48]).

Fréchet distance under translation. For some applications, it is useful to change the definition of the Fréchet distance slightly. In particular, several applications on curves evolve around the theme of *detecting movement patterns*. Consider the task of signature verification. Whether two signatures are similar is a translation-invariant property—intuitively, by translating a signature in space, we cannot make it more or less similar to another signature. Consider another example: Given GPS trajectories of an animal, we might want to detect different movement patterns (just considering their shape) by chopping the trajectories into smaller pieces and clustering these pieces according to some distance measure. For the two applications mentioned above, it is inconvenient that the Fréchet distance is not invariant under translation.[1] To overcome this issue, the *Fréchet distance under translation* between curves π, σ is defined as the minimal Fréchet distance between π and

[1]In this context one could even ask for a version of the Fréchet distance that is translation- and rotation-invariant, but we focus on the former in this article.

any translation of σ, i.e., we minimize over all possible translations of σ. Clearly, this yields a translation-invariant distance measure, and thus enables the above application.

The *continuous* Fréchet distance under translation was independently introduced by Efrat et al. [30] and Alt et al. [7], who designed algorithms in the plane with running time $\tilde{O}(n^{10})$ and $\tilde{O}(n^8)$, respectively.[2] Both groups of researchers also presented approximation algorithms, e.g., a $(1+\varepsilon)$-approximation running in time $O(n^2/\varepsilon^2)$ in the plane [7]. This line of work was extended to three dimensions with a running time of $\tilde{O}(n^{11})$ [51].

The *discrete* Fréchet distance under translation was first studied by Jiang et al. [41] who designed an $\tilde{O}(n^6)$-time algorithm in the plane. Mosig et al. [47] presented an approximation algorithm that computes the discrete Fréchet distance under translation, rotation, and scaling in the plane, up to a factor close to 2, and runs in time $O(n^4)$. The best known exact algorithm for the discrete Fréchet distance under translation in the plane is due to Ben Avraham et al. [12]. It is an improvement of the algorithm by Jiang et al. [41] and runs in time $\tilde{O}(n^5)$.

Our contribution. In this article, we further study the time complexity of the discrete Fréchet distance under translation in the plane. First, we improve the running time from $\tilde{O}(n^5)$ to $\tilde{O}(n^{4.66\cdots})$. This is achieved by designing an improved algorithm for a subroutine of the previously best algorithm, namely, offline dynamic s-t-reachability in directed grid graphs; see Section 1.1 below for a more detailed overview.

Theorem 1.1. *The discrete Fréchet distance under translation on curves of length n in the plane can be computed in time $\tilde{O}(n^{14/3}) = \tilde{O}(n^{4.66\cdots})$.*

Our second main result is a lower bound of $n^{4-o(1)}$, conditional on the standard Strong Exponential Time Hypothesis. The Strong Exponential Time Hypothesis essentially asserts that Satisfiability requires time $2^{n-o(n)}$; see Section 2 for a definition. This (conditionally) separates the discrete Fréchet distance under translation from the classic Fréchet distance, which can be computed in time $\tilde{O}(n^2)$. Moreover, the first step of all known algorithms for the discrete Fréchet distance under translation is to construct an arrangement of disks of size $O(n^4)$. Our conditional lower bound shows that this is essentially unavoidable.

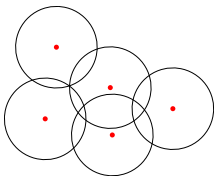
Theorem 1.2. *The discrete Fréchet distance under translation of curves of length n in the plane requires time $n^{4-o(1)}$, unless the Strong Exponential Time Hypothesis fails.*

We leave closing the gap between $\tilde{O}(n^{4.66\cdots})$ and $n^{4-o(1)}$ as an open problem.

1.1 Technical Overview

Previous algorithms for the discrete Fréchet distance under translation. Let us sketch the algorithms by Jiang et al. [41] and Ben Avraham et al. [12]. Given sequences $\pi = (\pi_1, \ldots, \pi_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_n)$ in $\mathbb{R}^2$ and a number $\delta \geq 0$, we want to decide whether the discrete Fréchet distance under translation of π and σ is at most δ. From this decision procedure one can obtain an algorithm to compute the actual distance via standard techniques (i.e., parametric search).

The translations τ for which the distance of π_i and $\sigma_j + \tau$ is at most δ form a disk in $\mathbb{R}^2$. Over all pairs (π_i, σ_j) this yields $O(n^2)$ disks, all of them having radius δ. Construct their arrangement $\mathcal{A}$ (see an illustration to the right), which is guaranteed to have $O(n^4)$ faces. Within each face of $\mathcal{A}$, any two translations are equivalent, in the sense that they leave the same pairs (π_i, σ_j) in distance at most δ. Thus, whether the discrete Fréchet distance is at most δ is constant in each face. Hence, it suffices to compute the discrete



[2] By $\tilde{O}(\cdot)$, we hide polylogarithmic factors in n.

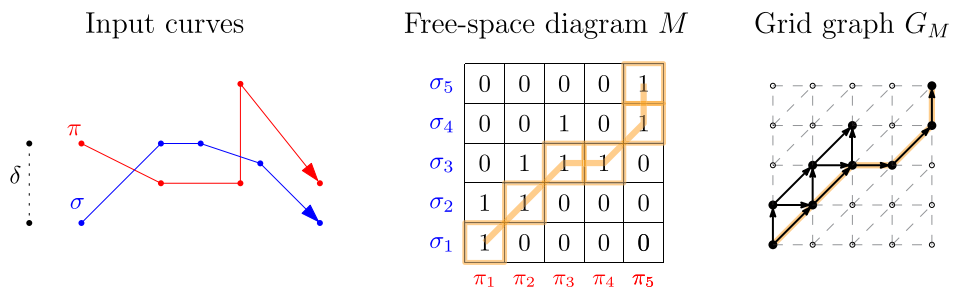


Fig. 1. Two input curves π, σ and a distance δ, the corresponding free-space diagram M, and the grid graph G_M corresponding to M. A monotone traversal of M and G_M is marked in orange.

Fréchet distance between π and σ translated by τ over $O(n^4)$ choices for τ, one for each face of $\mathcal{A}$. Since the discrete Fréchet distance can be computed in time $O(n^2)$, this yields an $O(n^6)$-time algorithm, which is essentially the algorithm by Jiang et al. [41].

Ben Avraham et al. [12] improve this algorithm as follows: Denote by M the $n \times n$ matrix with $M_{i,j} = 1$ if the points π_i, σ_j are in distance at most δ, and $M_{i,j} = 0$ otherwise (M is called the "free-space diagram"). It is well-known that the discrete Fréchet distance of π, σ is at most δ if and only if there exists a monotone path from the lower left to the upper right corner of M using only 1-entries. Equivalently, consider a *directed grid graph* G_M on $n \times n$ vertices, where each node (i, j) has directed edges to $(i + 1, j)$, $(i, j + 1)$, and $(i + 1, j + 1)$, and the nodes (i, j) of G_M with $M_{i,j} = 0$ are "deactivated" (i.e., removed). Then the discrete Fréchet distance of π, σ is at most δ if and only if node (n, n) is reachable from node $(1, 1)$ in G_M. See Figure 1 for an example of a pair of curves, its corresponding free-space diagram M, and directed grid graph G_M.

Ben Avraham et al. observe that it is easy to construct a sequence of $O(n^4)$ faces $f_1, \ldots, f_L$ of the arrangement $\mathcal{A}$ such that (1) each face of $\mathcal{A}$ is visited at least once and (2) f_ℓ and $f_{\ell+1}$ are neighboring in $\mathcal{A}$ for all ℓ. Since consecutive faces in this sequence are neighbors, only one pair (π_i, σ_j) changes its distance, i.e., either π_i, σ_j are in distance at most δ in f_ℓ and in distance larger than δ in $f_{\ell+1}$, or vice versa. This corresponds to one activation or deactivation of a node in G_M. After this update, we want to again check whether node (n, n) is reachable from node $(1, 1)$ in G_M. That is, using a dynamic algorithm for s-t-reachability in directed grid graphs, we can maintain whether the Fréchet distance is at most δ. The best known solution to dynamic reachability in directed $n \times n$ grids runs in time $\tilde{O}(n)$ [27].[3] Over all $O(n^4)$ faces, this yields time $\tilde{O}(n^5)$ for the discrete Fréchet distance under translation in the plane [12].

Intuition. There are two parts to the above algorithm: (1) Constructing the arrangement $\mathcal{A}$ and iterating over its faces and (2) maintaining reachability in the grid graph G_M. Both parts could potentially be improved.

The natural first attempt is to attack the arrangement enumeration, i.e., part (1). The size of the arrangement is $O(n^4)$, and for no other computational problem it is known—to the best of our knowledge—that any optimal algorithm must construct such a large arrangement, so this part seems intuitively wasteful. Surprisingly, our conditional lower bound of Theorem 1.2 shows that constructing the arrangement is essentially unavoidable.

The remaining part (2) at first sight seems much less likely to be improvable, since it is a well-known open problem to find a faster dynamic algorithm for reachability in directed grid graphs. Nevertheless, we show how to improve the running time of this part of the algorithm.

[3]This algorithm even works more generally for dynamic reachability in directed planar graphs.

Our algorithm. We observe that we do not need the full power of dynamic reachability, since we can precompute all $O(n^4)$ updates. This leaves us with the following problem.

Offline Dynamic Grid Reachability: We start from the directed $n \times n$-grid graph G in which all nodes are deactivated.[4] We are given a sequence of updates $u_1, \ldots, u_U$, where each u_ℓ is of the form "activate node (i, j)" or "deactivate node (i, j)." The goal is to compute for each $1 \leq \ell \leq U$ whether node $(1, 1)$ can reach node (n, n) in G after performing the updates $u_1, \ldots, u_\ell$.

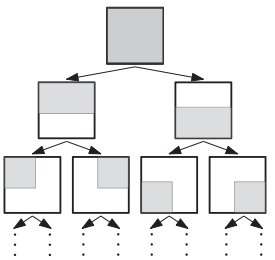
Our main algorithmic contribution is an algorithm for Offline Dynamic Grid Reachability in amortized time $\tilde{O}(n^{2/3})$ per update. This is faster than the update time $\tilde{O}(n)$ obtained by using a dynamic algorithm for reachability in directed planar graphs [27].

Theorem 1.3. *Offline Dynamic Grid Reachability can be solved in time* $\tilde{O}(n^2 + U \cdot n^{2/3})$.

The high-level approach of this algorithm is to consider all U updates in batches of size at most k, which we call *chunks*. Roughly speaking, we design a grid reachability data structure that given a chunk of k updates $u_1, \ldots, u_k$, enables us to (1) for any $1 \leq j \leq k$, answer a grid reachability query in the matrix updated by $u_1, \ldots, u_j$ in time $\tilde{O}(k)$ and (2) obtain the data structure for the matrix updated by the complete chunk $u_1, \ldots, u_k$ in time $\tilde{O}(n\sqrt{k}+k)$. This way, for each of the $O(U/k)$ chunks, we only need time $\tilde{O}(k^2)$ to answer all k reachability queries for this chunk and time $\tilde{O}(n\sqrt{k} + k)$ to update the data structure for the next chunk, leading to a total time of $\tilde{O}((U/k)(k^2 + n\sqrt{k})) = \tilde{O}(U(k + n/\sqrt{k}))$. By setting $k \approx n^{2/3}$, we obtain the desired algorithm running in time $\tilde{O}(Un^{2/3})$ after $\tilde{O}(n^2)$ preprocessing.

To obtain our data structure, we build on the reachability data structure of Ben Avraham et al. [12], augmented by two crucial insights: How to incorporate a chunk of k updates faster than k single updates, and how to succinctly store reachability information for k distinguished nodes in the grid (coined *terminals*, which correspond to the updates of the next chunk) in the data structure. The latter is given by a surprisingly succinct characterization of reachability of terminals in a grid graph (see Corollary 4.6), which is the key technical contribution for the algorithm.

Let us give a more detailed overview of our algorithm and its main ingredients. Start with a *block* $[n] \times [n]$ corresponding to the matrix M. Repeatedly split every block horizontally in the middle, and then split every block vertically in the middle, until we end up with constant-size blocks. We call all the blocks considered during this process (not just the constant-size blocks!) the "canonical" blocks; see the figure to the right. Ben Avraham et al. [12] showed that one can store for each canonical block of sidelength s reachability information for each pair of boundary nodes, succinctly represented using only $\tilde{O}(s)$ bits of space, and efficiently computable in time $\tilde{O}(s)$ from the information of the two canonical child-blocks. In particular, over all blocks this information can be maintained in time $\tilde{O}(n)$ per update u_i.



Ingredient 1: Batched updates. The first insight is that we can compute the reachability after a given chunk of k updates $u_1, \ldots, u_k$ faster than $\tilde{O}(nk)$: Intuitively, each update "touches" roughly $2 \log n$ blocks—all those that contain the node that is activated or deactivated. Our approach now uses that among the canonical blocks containing an update, the large blocks must be shared by many updates. Specifically, instead of recomputing the reachability information of the large blocks

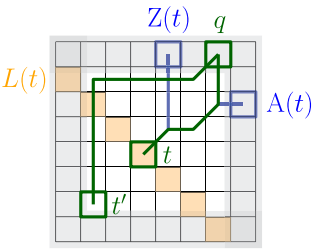
[4] In fact, our algorithm also works in the general case in which nodes can be arbitrarily activated/deactivated in the beginning.

at the top of the hierarchy k times, we perform those updates jointly and thus avoid the runtime of k explicit updates of large blocks. A careful tradeoff yields an update time of $\tilde{O}(n\sqrt{k}+k)$.

Ingredient 2: Reachability among terminals. Now fix a chunk $C = u_{\ell+1}, \ldots, u_{\ell+k}$ and let M denote the matrix at the beginning of C, i.e., after incorporating all updates prior to $u_{\ell+1}$. Denote by $\mathcal{T}$ ("terminals") the entries that get activated or deactivated during this chunk C and also add $(1,1)$ and (n,n) to the set of terminals. We first deactivate all terminals, obtaining a matrix $M^{\mathbf{0}}$ and a corresponding grid graph $G_{M^{\mathbf{0}}}$. The basic idea now is to determine for each pair of terminals $t, t' \in \mathcal{T}$ whether t' is reachable from t in $G_{M^{\mathbf{0}}}$.

Let us sketch a simplified algorithm that assumes we have built a graph H with vertex set $\mathcal{T}$, containing a directed edge (t,t') if and only if t' is reachable from t in $G_{M^{\mathbf{0}}}$. To answer the reachability query whether (n,n) is reachable from $(1,1)$ after updating M by $u_{\ell+1}, \ldots, u_{\ell+j}$, we proceed as follows: For each terminal t, activate t in H if and only if t is activated in M updated by $u_{\ell+1}, \ldots, u_{\ell+j}$. Check whether (n,n) is reachable from $(1,1)$ in H. Since H has $O(k)$ nodes and $O(k^2)$ edges, this reachability check can be performed in time $O(k^2)$. (By choosing a chunk size of $k \approx n^{2/5}$ this would result in an $\tilde{O}(Un^{4/5})$ algorithm for Offline Dynamic Grid Reachability, ignoring the preprocessing time.) We will later show how to improve the reachability query time from $O(k^2)$ to $\tilde{O}(k)$ by working directly on the graph $G_{M^{\mathbf{0}}}$ instead of constructing the graph H. These details are given in the subsequent sections.

It remains to describe how to determine reachability information among terminals. To this end, we design a surprisingly succinct representation of reachability from terminals to block boundaries. Consider a canonical block B and let $\mathcal{T}_B$ be the terminals in B. For each terminal $t \in \mathcal{T}_B$ let $A(t)$ be the lowest/rightmost point on the right/upper boundary of B that is reachable from t, and similarly let $Z(t)$ be the highest/leftmost reachable point see the illustration to the right. We label any terminal $t = (x,y)$ by $L(t) := x + y$, i.e., the anti-diagonal that t is contained in. For any right/upper boundary point q of B, let $\ell(q)$ be the minimal label of any terminal in $\mathcal{T}_B$ from which q is reachable. We prove the following succinct representation of reachability (see Corollary 4.6) that significantly generalizes a previous characterization for reachability among the *boundaries* of blocks [6, 12].



For any right/upper boundary point q of B and any terminal $t \in \mathcal{T}_B$,
q is reachable from t if and only if $q \in [A(t), Z(t)]$ and $\ell(q) \leq L(t)$.

Here, $q \in [A(t), Z(t)]$ is to be understood as "q lies between $A(t)$ and $Z(t)$ along the boundary of B," which can be expressed using a constant number of inequalities. The "only if" part is immediate, since t can only reach boundary vertices in $[A(t), Z(t)]$, and $\ell(q)$ is the minimal label of any terminal reaching q; the "if" part is surprising.

Assume we can maintain the information $A(t), Z(t), \ell(q)$. Then, using this characterization, we can determine all terminals reaching a boundary point q by a single call to *orthogonal range searching*, since we can express the characterization using a constant number of inequalities. A complex extension of this trick allows us to determine reachability among terminals (indeed, this technical overview is missing many details of Section 4). This yields our algorithm; see Sections 3 and 4 for details.

Conditional lower bound. Our reduction starts from the k-OV problem, which asks for k vectors from k given sets such that in no dimension all vectors are 1. More formally:

k-Orthogonal Vectors (k-OV). : Given sets $V_1, \ldots, V_k$ of N vectors in $\{0,1\}^D$, are there $v_1 \in V_1, \ldots, v_k \in V_k$ such that for any $j \in [D]$ there exists an $i \in [k]$ with $v_i[j] = 0$?

A naive algorithm solves k-OV in time $O(N^k D)$. It is well-known that the Strong Exponential Time Hypothesis implies that k-OV has no $O(N^{k-\varepsilon}\text{poly}(D))$-time algorithm for all $\varepsilon > 0$ and $k \geq 2$ [52].

In our reduction, we set $k = 4$. An overview of our construction can be found in Figure 10 on page 32. We consider *canonical translations* of the form $\tau = (\varepsilon \cdot h_1, \varepsilon \cdot h_2) \in \mathbb{R}^2$ with $h_1, h_2 \in \{0, \ldots, N^2 - 1\}$. By a simple gadget, we ensure that any translation resulting in a Fréchet distance of at most 1 must be close to a canonical translation. For simplicity, here we restrict our attention to exactly the canonical translations. Note that there are N^4 canonical translations, and thus they are in one-to-one correspondence to choices of vectors $(v_1, \ldots, v_4) \in V_1 \times \ldots \times V_4$. In other words, the outermost existential quantifier in the definition of 4-OV corresponds to the existential quantifier over the translation τ in the Fréchet distance under translation.

The next part in the definition of 4-OV is the universal quantifier over all dimensions $j \in [D]$. For this, our constructed curves π, σ are split into $\pi = \pi^{(1)} \ldots \pi^{(D)}, \sigma = \sigma^{(1)} \ldots \sigma^{(D)}$ such that $\pi^{(i)}, \sigma^{(j)}$ are very far for $i \neq j$. This ensures that the Fréchet distance of π, σ is the maximum over all Fréchet distances of $\pi^{(i)}, \sigma^{(i)}$, and thus simulates a universal quantifier.

The next part is an existential quantifier over $i \in [k]$. Here, we need an OR-gadget for the Fréchet distance. Such a construction in principle exists in previous work [1, 14], however, no previous construction would work with translations, in the sense that a translation in y-direction could only decrease the Fréchet distance. By constructing a more complex OR-gadget, we avoid this monotonicity.

Finally, we need to implement a check whether the translation τ corresponds to a particular choice of vectors. We exemplify this with the first dimension of the translation, which we call τ_1, explaining how it corresponds to choosing (v_1, v_2). Let $\text{ind}(v_1), \text{ind}(v_2) \in \{0, \ldots, N-1\}$ be the indices of these vectors in their sets V_1, V_2, respectively. We want to test whether $\tau_1 = \varepsilon \cdot (\text{ind}(v_1) + \text{ind}(v_2) \cdot N)$. We split this equality into two inequalities. For the inequality $\tau_1 \geq \varepsilon \cdot (\text{ind}(v_1) + \text{ind}(v_2) \cdot N)$, in one curve we place a point at $\pi_1 = (1 + \varepsilon \cdot \text{ind}(v_1), -1 - \eta)$, and in the other we place a point at $\sigma_1 = (-1 - \varepsilon \cdot \text{ind}(v_2) \cdot N, -1 - \eta)$, for some $\eta > 0$, which we specify later in this work. Then the distance of π_1 to the translated σ_1 is essentially their difference in x-coordinates, which is $(1 + \varepsilon \cdot \text{ind}(v_1)) - (-1 - \varepsilon \cdot \text{ind}(v_2) \cdot N + \tau_1) = 2 + \varepsilon \cdot (\text{ind}(v_1) + \text{ind}(v_2) \cdot N) - \tau_1$. This is at most 2 if and only if the inequality for τ_1 holds. We handle the opposite inequality similarly, and we concatenate the constructed points for both inequalities to test equality.

In total, our construction yields curves π, σ such that their discrete Fréchet distance under translation is at most 1 if and only if $V_1, \ldots, V_4$ contain orthogonal vectors. The curves π, σ consist of $n = O(D \cdot N)$ vertices. Hence, an algorithm for the discrete Fréchet distance under translation in time $O(n^{4-\varepsilon})$ would yield an algorithm for 4-OV in time $O(N^{4-\varepsilon}\text{poly}(D))$, and thus violate the Strong Exponential Time Hypothesis. See Section 5 for details.

1.2 Further Related Work

On directed planar/grid graphs. In this article, we improve offline dynamic s-t-reachability in directed grid graphs. The previously best algorithm for this problem came from a more general solution to dynamic reachability in directed planar graphs. For this problem, a solution with $\tilde{O}(N^{2/3})$ update time was given by Subramanian [50], which was later improved to update time $\tilde{O}(\sqrt{N})$ by Diks and Sankowski [27]. In particular, our work yields additional motivation to study offline variants of classic dynamic graph problems.

Related work on dynamic directed planar or grid graphs includes, e.g., shortest path computation [3, 40, 42], reachability in the decremental setting [39], or computing the transitive closure [27]. Recently, the first conditional lower bounds for dynamic problems on planar graphs were shown by Abboud and Dahlgaard [2], however, they did not cover dynamic reachability in directed planar graphs.

Other work on directed planar and grid graphs studies, e.g., the minimum amount of space necessary to determine reachability between two nodes in polynomial time, can be found in References [8, 9]. For grid graphs this was recently improved from $\tilde{O}(\sqrt{N})$ to $\tilde{O}(N^{1/3})$ [9], but with very different techniques compared to ours.

On related distance measures. A related distance measure to the Fréchet distance, which does not take traversals into account, is the Hausdorff distance. The Hausdorff distance can be computed in almost linear time in the plane. As for the Fréchet distance, there are also related variants of the Hausdorff distance that allow rigid motions of one of the input curves. For the Hausdorff distance under translation there exists a cubic-time algorithm [36] for all L_p norms in the plane. For the L_1 and L_∞ norm, there even is a quadratic-time algorithm for the Hausdorff distance under translation in the plane [24]. Surprisingly, despite this quadratic-time algorithm, the is a cubic lower bound for the size of the arrangement of the Hausdorff distance under translation that holds for all L_p norms [49].

The cubic-time algorithm for the Hausdorff distance under translation constructs the upper envelope of Voronoi surfaces and then tests for minima on the vertices and edges of this construction. Thus, this algorithm is also an arrangement-based approach. Similarly, the first algorithms [7, 30] for the continuous Fréchet distance under translation as well as the weak continuous Fréchet distance under translation relied on arrangement constructions.

A recently introduced variant of the Fréchet distance is the *Fréchet gap* [32, 33]. Some researchers have argued that this measure is similar to the Fréchet distance under translation in certain aspects, in particular, since the Fréchet gap between a curve π and a translation $\pi + \tau$ of the same curve is 0 [32]. Moreover, the Fréchet gap can be computed significantly faster, with the currently fastest known algorithm running in time $O(n^3)$ [33]. In some sense, our conditional lower bound in Theorem 1.2 explains why replacing the Fréchet distance under translation by such a surrogate measure is necessary to obtain more efficient algorithms. Additionally, the discrete Fréchet distance *with shortcuts* was also recently considered in a translation-invariant setting [34].

On related reachability data structures. In Reference [6], a reachability data structure on the free-space diagram is given to compute the Fréchet distance between closed curves and also to compute the best matching to any subcurve under the Fréchet distance. In Reference [44], a data structure called the free-space map is presented, which improves the reachability data structure of Reference [6] and as a consequence shaves off logarithmic factors in the running time for computing the above-mentioned applications as well as improving the running time for other applications.

1.3 Organization

We start off with introducing basic definitions, notational conventions, and algorithmic tools in Section 2. Afterwards, in Section 3, we give an overview of our algorithmic result and we reduce the problem to designing a certain data structure for Offline Dynamic Grid Reachability. This data structure, our main technical contribution, is developed in Section 4. Finally, we prove our conditional lower bound of $n^{4-o(1)}$ in Section 5.

2 PRELIMINARIES

We let $[n]$ denote the set $\{1, \ldots, n\}$. Furthermore, for convenience, we use as convention that $\min \emptyset = \infty$ and $\max \emptyset = -\infty$.

2.1 Curves, Traversals, Fréchet distances, and more

A polygonal curve π of length n over $\mathbb{R}^d$ is a sequence of points $\pi_1, \ldots, \pi_n \in \mathbb{R}^d$. Throughout the article, we only consider polygonal curves in the Euclidean plane, i.e., $d = 2$. Given any translation vector $\tau \in \mathbb{R}^2$, we denote by $\pi + \tau$ the polygonal curve $\pi' = (\pi'_1, \ldots, \pi'_n)$ given by $\pi'_i = \pi_i + \tau$.

We call any pair $(i, j) \in [n] \times [n]$ a *position*. A *traversal* T is a sequence $t_1, \ldots, t_\ell$ of positions, where $t_k = (i, j)$ implies that t_{k+1} is either $(i+1, j)$ (that is, we advance one step in π while staying in σ_j), $(i, j+1)$ (we advance in σ while staying in π_i) or $(i+1, j+1)$ (we advance in both curves simultaneously). We call $T = (t_1, \ldots, t_\ell)$ a traversal of π, σ, if $t_1 = (1, 1)$ and $t_\ell = (n, n)$.

We now define two types of concatenations: a concatenation of curves and a concatenation of traversals. Let $\pi = (\pi_1, \ldots, \pi_n), \sigma = (\sigma_1, \ldots, \sigma_n)$ be polygonal curves of lengths n. We define the concatenation of π and σ as $\pi \circ \sigma := (\pi_1, \ldots, \pi_n, \sigma_1, \ldots, \sigma_n)$. The resulting curve has length $2n$. We now define the concatenation of traversals. Given two traversals $T = (t_1, \ldots, t_\ell)$ and $T' = (t'_1, \ldots, t'_{\ell'})$ with $t_\ell = t'_1$, we define the concatenated traversal as $T \circ T' := (t_1, \ldots, t_\ell = t'_1, t'_2, \ldots, t'_{\ell'})$. Note that we obtain a traversals from t_1 to $t'_{\ell'}$.

The *discrete Fréchet distance* is formally defined as

$$\delta_F(\pi, \sigma) := \min_{T=((i_1,j_1),\ldots,(i_\ell,j_\ell))} \max_{1 \le k \le \ell} \|\pi_{i_k} - \sigma_{j_k}\|,$$

where T ranges over all traversals of π, σ and $\|\cdot\|$ denotes the Euclidean distance in $\mathbb{R}^2$.

We obtain a well-known equivalent definition as follows: Fix some distance $\delta \ge 0$. We call a position (i, j) *free* if $\|\pi_i - \sigma_j\| \le \delta$. We say that a traversal $T = (t_1, \ldots, t_\ell)$ of π, σ is a *valid traversal* for δ if $t_1, \ldots, t_\ell$ are all free positions. The discrete Fréchet distance of π, σ is then the smallest δ such that there is a valid traversal of π, σ for δ.

Analogously, consider the $n \times n$ matrix M with $M_{i,j} = 1$ if (i, j) is free, and $M_{i,j} = 0$ otherwise. We call any traversal $T = (t_1, \ldots, t_\ell)$ a *monotone path from t_1 to t_ℓ*. If all positions (i, j) visited by T satisfy $M_{i,j} = 1$, then we call T a *monotone 1-path* from t_1 to t_ℓ in M. As yet another formulation, consider the $n \times n$ grid graph G_M where vertex (i, j) has directed edges to all of $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$ (in case they exist). Deactivate (i.e., remove) all non-free vertices (i, j) from G_M. Then a monotone 1-path in M corresponds to a (directed) path in G_M. Hence, $\delta_F(\pi, \sigma) \le \delta$ is equivalent to the existence of a valid traversal of π, σ for δ, which in turn is equivalent to the existence of a monotone 1-path from $(1, 1)$ to (n, n) in the matrix M, and to vertex (n, n) being reachable from $(1, 1)$ in G_M.

Finally, we define the discrete Fréchet distance under translation as $\min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau)$, i.e., the smallest discrete Fréchet distance of π to any translation of σ.

2.2 Hardness Assumptions

The ***Strong Exponential Time Hypothesis (SETH)*** was introduced by Impagliazzo and Paturi [37] and essentially postulates that there is no exponential-time improvement over exhaustive search for the Satisfiability Problem.

Hypothesis 2.1 (Strong Exponential Time Hypothesis (SETH)). *For any $\epsilon > 0$ there exists $k \ge 3$ such that k-SAT has no $O((2-\epsilon)^n)$-time algorithm.*

In fact, our reductions even hold under a weaker assumption, specifically, the *k-OV Hypothesis*.[5] Recall the k-OV problem: Given sets $V_1, \ldots, V_k$ of N vectors in $\{0, 1\}^D$, the task is to determine

[5] In fact, we only need the corresponding hypothesis for 4-OV.

whether there are $v_1 \in V_1, \ldots, v_k \in V_k$ such that for all $j \in [D]$ there exists an $i \in [k]$ with $v_i[j] = 0$.

Hypothesis 2.2 (k-OV Hypothesis). *For any $k \geq 2$ and $\epsilon > 0$, there is no $O(N^{k-\epsilon}\text{poly}(D))$-time algorithm for k-OV.*

The well-known split-and-list technique due to Williams [52] shows that SETH implies the k-OV Hypothesis. Thus, any conditional lower bound that holds under the k-OV hypothesis also holds under SETH.

2.3 Orthogonal Range Data Structures

We will use a tool from geometric data structures, namely, *(dynamic) orthogonal range data structures*. Let S be a set of key-value pairs $s = (k_s, v_s) \in \mathbb{Z}^d \times \mathbb{Z}$; in our applications, we will have $d = 2$ or $d = 3$. An orthogonal range data structure on S enables us to query the maximal value of any pair in S whose key lies in a given orthogonal range. Formally, we say $\mathcal{OR}$ *stores v_s under the key k_s for $s \in S$ for minimization queries*, if $\mathcal{OR}$ supports, for any $\ell_1, u_1, \ell_2, u_2, \ldots, \ell_d, u_d \in \mathbb{Z} \cup \{-\infty, \infty\}$, queries of the form

$$\mathcal{OR}.\min([\ell_1, u_1] \times \cdots \times [\ell_d, u_d]) : \text{ return } \min\{v_s \mid s = (k_s, v_s) \in S, k_s \in [\ell_1, u_1] \times \cdots \times [\ell_d, u_d]\}.$$

We will also consider analogous maximization queries.

Classic results [23, 35] show that for any set S of size n and $d = 2$, we can construct such a data structure $\mathcal{OR}$ in time and space $O(n \log n)$, supporting minimization (or maximization) queries in time $O(\log n)$.

In Section 4.6.2, we will also use an orthogonal range searching data structure that allows (1) to *report* all values of pairs in S whose keys lie in a given orthogonal range and (2) to *remove* a key-value pair from S. Formally, we say that $\mathcal{OR}$ *stores v_s under the key k_s for $s \in S$ for decremental range reporting queries*, if $\mathcal{OR}$ supports, for any $\ell_1, u_1, \ell_2, u_2, \ldots, \ell_d, u_d \in \mathbb{Z} \cup \{-\infty, \infty\}$, queries of the form

$$\mathcal{OR}.\text{report}([\ell_1, u_1] \times \cdots \times [\ell_d, u_d]) : \text{ return } \{v_s \mid s = (k_s, v_s) \in S, k_s \in [\ell_1, u_1] \times \cdots \times [\ell_d, u_d]\},$$

as well as deletions from the set S.

Mortensen [46] and Chan and Tsakalidis [22] showed how to construct such a data structure $\mathcal{OR}$ for any set S of size n in time and space $O(n \log^{d-1} n)$, deletion time $O(\log^{d-1} n)$, and query time $O(\log^{d-1} n + k)$, where k denotes the output size of the query. (These works obtain even stronger results, however, we use simplified bounds for ease of presentation.)

3 ALGORITHM: REDUCTION TO GRID REACHABILITY

In this section, we prove our algorithmic result by showing how a certain grid reachability data structure (that we give in Section 4) yields an $\tilde{O}(n^{4+2/3})$-time algorithm for computing the discrete Fréchet distance under translation.

We start with a formal overview of the algorithm. First, we reduce the decision problem (i.e., is the discrete Fréchet distance under translation of π, σ at most δ?) to the problem of determining reachability in a dynamic grid graph, as shown by Ben Avraham et al. [12]. However, noting that all updates and queries are known in advance, we observe that the following *offline* version suffices.

Problem 3.1 (Offline Dynamic Grid Reachability). *Let M be an $n \times n$ matrix over $\{0, 1\}$. We call $u = (p, b)$ with $p \in [n] \times [n]$ and $b \in \{0, 1\}$ an* update *and define $M[\![u]\!]$ as the matrix obtained by setting the bit at position p to b, i.e.,*

$$M[\![(p, b)]\!]_{i,j} = \begin{cases} b & \text{if } p = (i, j), \\ M_{i,j} & \text{otherwise.} \end{cases}$$

For any sequence of updates $u_1, \ldots, u_k \in ([n] \times [n]) \times \{0,1\}$ *with* $k \geq 2$, *we define* $M[\![u_1, \ldots, u_k]\!] := (M[\![u_1]\!])[\![u_2, \ldots, u_k]\!]$.

The Offline Dynamic Grid Reachability *problem asks to determine, given* M *and any sequence of updates* $u_1, \ldots, u_U \in ([n] \times [n]) \times \{0,1\}$, *whether there is a monotone 1-path from* $(1,1)$ *to* (n,n) *in* $M[\![u_1, \ldots, u_k]\!]$ *for any* $1 \leq k \leq U$.

We show the following reduction in Section 3.1.

Lemma 3.2. *Assume there is an algorithm solving Offline Dynamic Grid Reachability in time* $T(n,U)$. *Then there is an algorithm that, given* $\delta > 0$ *and polygonal curves* π, σ *of length* n *over* $\mathbb{R}^2$, *determines whether* $\delta_F(\pi, \sigma + \tau) \leq \delta$ *for some* $\tau \in \mathbb{R}^2$ *in time* $O(T(n, n^4))$.

Our speedup is achieved by solving Offline Dynamic Grid Reachability in time $T(n,U) = \tilde{O}(n^2 + Un^{2/3})$ (Ben Avraham et al. [12] achieved $T(n,U) = O(n^2 + Un)$). To this end, we devise a *grid reachability data structure*, which is our central technical contribution.

Lemma 3.3 (Grid Reachability Data Structure). *Given an* $n \times n$ *matrix* M *over* $\{0,1\}$ *and a set of* terminals $\mathcal{T} \subseteq [n] \times [n]$ *of size* $k > 0$, *there is a data structure* $\mathcal{D}_{M,\mathcal{T}}$ *with the following properties.*

(i) (Construction:) We can construct $\mathcal{D}_{M,\mathcal{T}}$ *in time* $O(n^2 + k \log^2 n)$.
(ii) (Reachability Query:) Given $F \subseteq \mathcal{T}$, *we can determine in time* $O(k \log^3 n)$ *whether there is a monotone path from* $(1,1)$ *to* (n,n) *using only positions* (i,j) *with* $M_{i,j} = 1$ *or* $(i,j) \in F$.
(iii) (Update:) Given $\mathcal{T}' \subseteq [n] \times [n]$ *of size* k *and an* $n \times n$ *matrix* M' *over* $\{0,1\}$ *differing from* M *in at most* k *positions, we can update* $\mathcal{D}_{M,\mathcal{T}}$ *to* $\mathcal{D}_{M',\mathcal{T}'}$ *in time* $O(n\sqrt{k} \log n + k \log^2 n)$. *Here, we assume* M' *to be represented by the set* Δ *of positions in which* M *and* M' *differ.*

Section 4 is dedicated to devising this data structure, i.e., proving Lemma 3.3. Equipped with this data structure, we can efficiently *batch* updates and queries to the data structure. Specifically, we obtain the following theorem:

Theorem 3.1. *Offline Dynamic Grid Reachability can be solved in time* $O(n^2 + Un^{2/3} \log^2 n)$.

We prove this theorem in Section 3.2. Finally, it remains to use standard techniques of parametric search to transform the decision algorithm to an algorithm computing the discrete Fréchet distance under translation. This has already been shown by Ben Avraham et al. [12]; we sketch the details in Section 3.3.

Lemma 3.4. *Let* $T_{\text{dec}}(n)$ *be the running time to decide, given* $\delta > 0$ *and polygonal curves* π, σ *of length* n *over* $\mathbb{R}^2$, *whether* $\delta_F(\pi, \sigma + \tau) \leq \delta$ *for some* $\tau \in \mathbb{R}^2$. *Then there is an algorithm computing the discrete Fréchet distance under translation for any curves* π, σ *of length* n *over* $\mathbb{R}^2$ *in time* $O((n^4 + T_{\text{dec}}(n)) \log n)$.

Combining Lemma 3.4, Lemma 3.2, and Theorem 3.1, we obtain an algorithm computing the discrete Fréchet distance under translation in time

$$O((n^4 + T(n, n^4)) \log n) = O(n^{4+2/3} \log^3 n),$$

as desired. In the remainder of this section, we provide the details of all steps mentioned above, except for Lemma 3.3 (which we prove in Section 4).

3.1 Reduction to Offline Dynamic Grid Reachability

In this section, we prove Lemma 3.2. Given polygonal curves π, σ of length n over $\mathbb{R}^2$ and $\delta > 0$, we determine whether $\delta_F(\pi, \sigma + \tau) \leq \delta$ for some $\tau \in \mathbb{R}^2$ as follows:

For any radius r and point $p \in \mathbb{R}^2$, we let $D_r(p)$ denote the disk of radius r with center p.

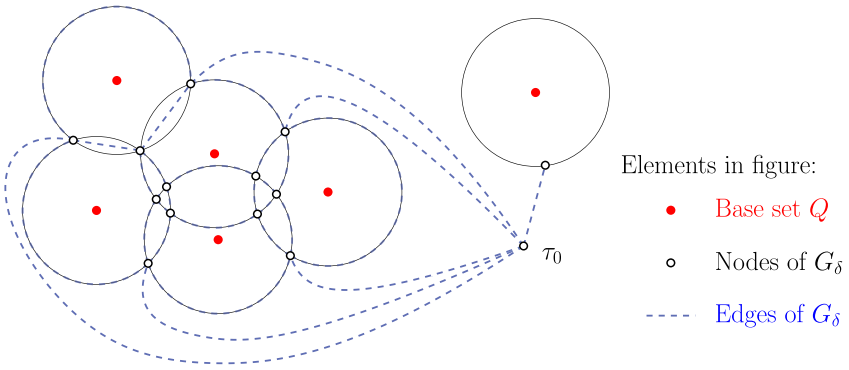


Fig. 2. Arrangement $\mathcal{A}_\delta$ and construction of G_δ.

Observation 3.5. *Let $\tau \in \mathbb{R}^2$ and define the $n \times n$ matrix M^τ over $\{0,1\}$ by*

$$M^\tau_{i,j} = 1 \qquad \Longleftrightarrow \qquad \tau \in D_\delta(\pi_i - \sigma_j).$$

We have $\delta_F(\pi, \sigma + \tau) \le \delta$ if and only if there is a monotone 1-path from $(1,1)$ to (n,n) in M^τ.

By the above observation, it suffices to check for the existence of monotone 1-paths from $(1,1)$ to (n,n) in a bounded number of matrices. To this end, let $Q := \{\pi_i - \sigma_j \mid i,j \in [n]\}$. We construct the arrangement $\mathcal{A}_\delta$ of the disks $D_\delta(q)$ for $q \in Q$, in the sense that we construct the following plane graph G_δ (cf. Figure 2): First, we include the vertices of $\mathcal{A}_\delta$ in its node set (i.e., intersections of disks $D_\delta(q), D_\delta(q')$ with $q, q' \in Q$). Second, for each $q \in Q$ for which $D_\delta(q)$ intersects no $D_\delta(q')$ for $q' \in Q \setminus \{q\}$, we include an arbitrary τ_q on the boundary of $D_\delta(q)$. Finally, we add an arbitrary vertex $\tau_0 \in \mathbb{R}^2$ lying in the outer face of $\mathcal{A}_\delta$ to the node set. Any nodes τ, τ' of G_δ are connected by an edge if they are neighboring vertices on the boundary of some face of $\mathcal{A}_\delta$; additionally, we connect τ_0 to all nodes that lie on the boundary separating the outer face from some other face. Observe that G_δ is a connected plane graph, has $O(|Q|^2) = O(n^4)$ nodes and edges, and can be constructed in time $O(n^4)$.

Note that by Observation 3.5, it suffices to check whether $\delta_F(\pi, \sigma + \tau_v) \le \delta$ for any node τ_v in[6] G_δ: for any (bounded) face f of $\mathcal{A}_\delta$, there is at least one point τ_v in G_δ that lies on the boundary of f. The corresponding matrix M^{τ_v} has at least the same 1-positions as the matrix M^τ for any $\tau \in f$ (and might have more).

To obtain a walk visiting all nodes in G_δ, we simply compute a spanning tree T of G_δ, double all edges of T, and find an Eulerian cycle starting and ending in τ_0. Denote this cycle by $\tau_0, \dots, \tau_L$ and observe that $L = O(n^4)$. Let $M_0 = M^{\tau_0}$ be the $n \times n$ all-zeroes matrix. For any $0 \le i < L$, we construct an update sequence $\bar{u}_i$ that first sets all positions (i,j) with $M^{\tau_i} = 1$ and $M^{\tau_{i+1}} = 0$ to zero, and then sets all positions (i,j) with $M^{\tau_i} = 0$ and $M^{\tau_{i+1}} = 1$ to 1. Thus, if we start with M^{τ_i} and perform the updates in $\bar{u}_i$, then at any point in time, the current matrix is dominated by either M^{τ_i} or $M^{\tau_{i+1}}$ (that is, the free positions of the current matrix are always a subset of M^{τ_i}'s free positions or a subset of $M^{\tau_{i+1}}$'s free positions), and at the end, we obtain $M^{\tau_{i+1}}$. Thus, by concatenating all updates to $\bar{u}_0, \dots, \bar{u}_{L-1}$, we obtain an instance of the Offline Dynamic Grid Reachability problem with initial matrix M_0 and update sequence $u_1, \dots, u_{L'}$ with the following property: There is some

[6]Note that our approach here deviates somewhat from the description in the introduction. This is due to the fact that for adversarial δ, we might need to consider degenerate faces consisting of a single point only; due to the parametric search that we describe in Section 3.3, we may not assume δ to avoid such degenerate cases. Traversing vertices of the arrangement instead of the faces takes care of such border cases in a natural manner.

$i \in \{0, \dots, L\}$ with $\delta_F(\pi, \sigma + \tau_i) \leq \delta$ if and only if there is some $i' \in [L']$ such that (n, n) is reachable from $(1, 1)$ via a monotone 1-path in $M_0[\![u_1, \dots, u_{i'}]\!]$. Since $\tau_0, \dots, \tau_L$ visits all nodes in G_δ, this is equivalent to testing whether $\delta_F(\pi, \sigma + \tau) \leq \delta$ for any $\tau \in \mathbb{R}^2$.

It remains to bound L'. We assume general position of the input points $P \cup S$ where $P = \{\pi_i \mid i \in [n]\}$ and $S = \{\sigma_j \mid j \in [n]\}$. Observe that there is some universal constant C such that no C points in Q lie on a common circle.[7] Thus, if we move from vertex τ_i to τ_{i+1} along an edge in G_δ, e.g., from one vertex of the boundary of some face to a neighboring vertex on that boundary, then there are at most $2C$ entries that change from M^{τ_i} to $M^{\tau_{i+1}}$, since for both τ_i and τ_{i+1}, there are at most C disks intersecting this vertex and no other entries change when moving along this edge (by construction of G_δ). Thus, $L' \leq 2CL = O(n^4)$. Consequently, given an algorithm solving Offline Dynamic Grid Reachability in time $T(n, U)$, we can determine whether $\delta_F(\pi, \sigma + \tau) \leq \delta$ for some $\tau \in \mathbb{R}^2$ in time $O(T(n, L')) = O(T(n, n^4))$.

3.2 Solving Offline Dynamic Grid Reachability

We prove Theorem 3.1 using the grid reachability data structure given in Lemma 3.3. Specifically, we claim that the following algorithm (formalized as Algorithm 1) solves Offline Dynamic Grid Reachability in time $O(n^2 + Un^{2/3} \log^2 n)$. We partition our updates $u_1, \dots, u_U$ into groups $\bar{u}_1, \dots, \bar{u}_{O(U/k)}$ containing k updates each. For any group $\bar{u}_i$, let M_i be obtained from M by performing all updates *prior* to $\bar{u}_i$. Note that $\bar{u}_i$ will update a set of at most k positions; denote this set by $\mathcal{T}_i$. We build the grid reachability data structure $\mathcal{D}_i = \mathcal{D}_{M_i^0, \mathcal{T}_i}$ with terminal set $\mathcal{T}_i$ and matrix M_i^0 obtained from M_i by setting the positions of all terminals $\mathcal{T}_i$ to 0. Observe that the state after any update within $\bar{u}_i$ corresponds to M_i^0 with some additional positions in $\mathcal{T}_i$ set to 1 (the *free terminals*). Thus, for each update within $\bar{u}_i$, we can determine whether it creates a monotone 1-path from $(1, 1)$ to (n, n) by simply determining the set $F \subseteq \mathcal{T}_i$ of free terminals at the point of this update and performing the corresponding reachability query in $\mathcal{D}_i$. It is straightforward to argue that the resulting algorithm correctly solves Offline Dynamic Grid Reachability.

To analyze the running time of Algorithm 1, note that each data structure $\mathcal{D}_{M_i^0, \mathcal{T}_i}$ has a terminal set of size at most k and each M_{i+1}^0 differs from M_i^0 in at most $2k$ entries. Thus, by Lemma 3.3, we need time $O(n^2 + k \log^2 n)$ to build $\mathcal{D}_1 = \mathcal{D}_{M_1^0, \mathcal{T}_1}$ in Line 7. The time spent for handling a single group $\bar{u}_i$ is bounded by the time to perform k queries in $\mathcal{D}_i = \mathcal{D}_{M_i^0, \mathcal{T}_i}$ plus the time to update $\mathcal{D}_i = \mathcal{D}_{M_i^0, \mathcal{T}_i}$ to $\mathcal{D}_{i+1} = \mathcal{D}_{M_{i+1}^0, \mathcal{T}_{i+1}}$, which amounts to $O(k^2 \log^3 n + n\sqrt{k} \log n + k \log^2 n) = O(k^2 \log^3 n + n\sqrt{k} \log n)$ by Lemma 3.3. Thus, in total, we obtain a running time of

$$O\left(n^2 + k \log^2 n + \frac{U}{k}\left(k^2 \log^3 n + n\sqrt{k} \log n\right)\right) = O\left(n^2 + U\left(k \log^3 n + \frac{n}{\sqrt{k}} \log n\right)\right).$$

This expression is minimized by setting $k := n^{2/3} / \log^{4/3} n$, resulting in a total running time of $O(n^2 + Un^{2/3} \log^{1+2/3} n) = O(n^2 + Un^{2/3} \log^2 n)$, as desired.

[7] To be more precise, we sketch how to argue that the general position assumption for $P \cup S$ "transfers" to Q. Assume that there exist points $q_1, \dots, q_\ell \in Q$ lying on a common circle. For all i, we must have $q_i = p_i - s_i$ for some $p_i \in P$, $s_i \in S$. First assume that $\ell = 4$ and that there is some s such that $s_i = s$ for all $i \in \{1, 2, 3, 4\}$. Then already $p_1, \dots, p_4$ lie on a common circle (it has the same radius as the original circle, and its center is translated by s), which violates the general position assumption of points in P. Otherwise, let the points $q_1, \dots, q_\ell$ be arbitrary with $\ell \geq 36$. By the first case, any s_i appears at most three times among $s_1, \dots, s_\ell$. After removing copies, we may assume without loss of generality that $q_1, \dots, q_{\ell'}$ with $\ell' \geq \ell/3$ have distinct s_i's. Similarly, we may also assume that $q_1, \dots, q_{\ell''}$ with $\ell'' \geq \ell/9 \geq 4$ have distinct p_i's as well. The fact that q_4 lies on the circle defined by q_1, q_2, q_3 can be expressed by a nonzero degree-2 polynomial $P_{q_1, q_2, q_3}(x, y)$ vanishing on q_4. Since $q_4 = p_4 - s_4$, we obtain a nonzero degree-2 polynomial $P'_{q_1, q_2, q_3}(p, s)$ vanishing on (p_4, s_4). This contradicts general position of $P \cup S$.

3.3 Parametric Search

In this section, we sketch how to use parametric search techniques (due to Megiddo [45] and Cole [25]) to reduce the optimization problem to the decision problem with small overhead, i.e., we prove Lemma 3.4. Specifically, for the readers' convenience, we describe the arguments made by Ben Avraham et al. [12] in slightly more detail.

ALGORITHM 1: Solving Offline Dynamic Grid Reachability on matrix M and update sequence $u_1, \dots, u_U$.

function OfflineDynamicGridReachability($M, u_1, \dots, u_U$)
  parameter: k
  Divide $u_1, \dots, u_U$ into $s = \lceil \frac{U}{k} \rceil$ subsequences $\bar{u}_1, \dots, \bar{u}_s$ of length k.[8]
  Initialize $M_1 \leftarrow M$
  Set $\mathcal{T}_1$ to the set of positions updated in $\bar{u}_1$.
  Let M_1^0 be obtained from M_1 by updating all positions in $\mathcal{T}_1$ to 0
  Build $\mathcal{D}_{M_1^0, \mathcal{T}_1}$
  for $i \leftarrow 1$ **to** s **do**
    for $j \leftarrow 1$ **to** k **do**
      Let $F \subseteq \mathcal{T}_i$ be the free terminals in $M_i[\![\bar{u}_i[1], \dots, \bar{u}_i[j]]\!]$
      if reachability query in $\mathcal{D}_{M_i^0, \mathcal{T}_i}$ with free terminals F is successful **then**
        return true
    Set $M_{i+1} \leftarrow M_i[\![\bar{u}_i]\!]$
    Set $\mathcal{T}_{i+1}$ to the set of positions updated in $\bar{u}_{i+1}$.[9]
    Let M_{i+1}^0 be obtained from M_{i+1} by updating all positions in $\mathcal{T}_{i+1}$ to 0
    update $\mathcal{D}_{M_i^0, \mathcal{T}_i}$ to $\mathcal{D}_{M_{i+1}^0, \mathcal{T}_{i+1}}$
  return false

Our aim in this section is to compute the discrete Fréchet distance under translation of polygonal curves π, σ of length n over $\mathbb{R}^2$, i.e., to determine

$$\delta^* := \min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau).$$

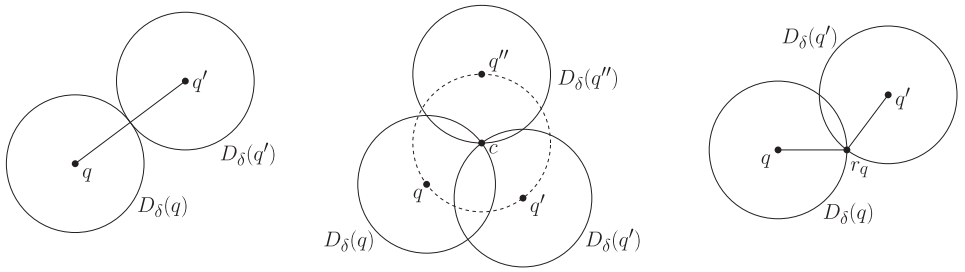
Using the decision algorithm, we can determine, for any $\delta > 0$, whether $\delta^* \leq \delta$ in time $T_{\text{dec}}(n)$. As we shall see below, there is a range of $O(n^6)$ possible values (defined by the point set of π, σ) that δ^* might attain (called *critical values*). Naively computing all critical values and performing a binary search would result in an $O((n^6 + T_{\text{dec}}(n)) \log n)$-time algorithm, which is too slow for our purposes. Instead, we use the parametric search technique to perform an implicit search over these critical values.

Conceptually, we aim to determine the combinatorial structure of the arrangement $\mathcal{A}_{\delta^*}$ defined in Section 3.1 (captured by the graph G_{δ^*}) without knowing δ^* in advance. To specify this combinatorial structure, define for every $q \in Q$ the set

$$I_\delta(q) := \{q' \in Q \setminus \{q\} \mid D_\delta(q), D_\delta(q') \text{ intersect}\}.$$

Note that for every $q' \in I_\delta(q)$, there are one or two intersection points of $D_\delta(q)$, $D_\delta(q')$, which we denote by $C_\delta^1(q, q')$ and $C_\delta^2(q, q')$ (note that we allow these points to coincide if $D_\delta(q), D_\delta(q')$ intersect in a single point only)—we assume this notation to be chosen consistently in the sense

[8] If necessary, then repeat the last element of the last group to make all groups consist of exactly k updates.
[9] We let $\bar{u}_{s+1}$ consist of k arbitrary updates, as we will never make use of these values.



(a) The elements in $L_\delta(q)$ change at radius δ that is a half-distance to some other $q' \in Q$.

(b) The ordering of $L_\delta(q)$ might change at radius δ such that $q, q', q'' \in Q$ lie on a common circle around some center c.

(c) The ordering of $L_\delta(q)$ might change at radius δ such that $D_\delta(q')$ intersects $D_\delta(q)$ in r_q.

Fig. 3. Critical values for the lists $L_\delta(q), q \in Q$.

that $C^1_\delta(q, q')$ and $C^1_\delta(q', q)$ refer to the same point (likewise for $C^2_\delta(q, q')$ and $C^2_\delta(q', q)$). We denote by $C_\delta(q)$ the set of all intersection points on the boundary of $D_\delta(q)$, i.e., $C^1_\delta(q, q')$, $C^2_\delta(q, q')$ for all $q' \in I_\delta(q)$. We obtain a list $L_\delta(q)$ by starting with the rightmost point on the boundary of $D_\delta(q)$, say, r_q, and listing all intersection points $C \in C_\delta(q)$ in counter-clockwise order. Observe that the combinatorial structure of $\mathcal{A}_\delta$ is completely specified by the lists $L_\delta(q)$ for $q \in Q$.

We wish to construct $L_{\delta^*}(q)$ for all $q \in Q$ using calls to our decision algorithm, i.e., queries of the form "Is $\delta^* \leq \delta$?" Along the way, we maintain a shrinking interval $(\alpha, \beta]$ such that $\delta^* \in (\alpha, \beta]$—our aim is that in the end $(\alpha, \beta]$ no longer contains critical values except for β, and thus $\delta^* = \beta$ can be derived. We proceed in two steps.

Step 1: Determining $I_{\delta^*}(q)$. The critical values for this step are the half-distances of all pairs $q, q' \in Q$ (cf. Figure 3(a)). We list all these values and perform a binary search over them, using our decision algorithm. Since there are at most $O(|Q|^2) = O(n^4)$ such values, we obtain an algorithm running in time $O((n^4 + T_{\text{dec}}(n)) \log n)$ returning an interval $(\alpha_1, \beta_1]$ such that $\delta^* \in (\alpha_1, \beta_1]$ and no half-distance of a pair $q, q' \in Q$ is contained in (α_1, β_1). Thus, from this point on we know $I_{\delta^*}(q)$ for all $q \in Q$ (without knowing the exact value of δ^* yet).

Step 2: Sorting $L_{\delta^*}(q)$. We use the following well-known variant of Meggido's parametric search that is due to Cole [25].

Lemma 3.6 (Implicit in [25]). *Let parametric values $f_1(\delta), \ldots, f_N(\delta)$ be given. Assume there is an unknown value $\delta^* > 0$ and a decision algorithm determining, given $\delta > 0$, whether $\delta^* \leq \delta$ in time $T(N)$. If we can determine $f_i(\delta^*) \leq f_j(\delta^*)$ for any $i, j \in [N]$ using only a constant number of queries to the decision algorithm, then in time $O((N+T(N)) \log n)$, we can sort $f_1(\delta^*), \ldots, f_N(\delta^*)$ and obtain an interval $(\alpha, \beta]$ such that $\delta^* \in (\alpha, \beta]$ and no critical value for the sorted order of $f_1(\delta), \ldots, f_N(\delta)$ is contained in (α, β).*

Consider first the problem of sorting $L_{\delta^*}(q)$ for some $q \in Q$. By the above technique, we only need to argue that we can determine whether some $C^a_{\delta^*}(q, q')$ with $q' \in Q, a \in \{1, 2\}$ precedes some $C^b_{\delta^*}(q, q'')$ with $q'' \in Q, b \in \{1, 2\}$ in $L_{\delta^*}(q)$. Note that $C^a_\delta(q, q'), C^b_\delta(q, q'')$ move continuously on the boundary of $D_\delta(q)$ (while δ varies) and there are only constantly many choices of δ for which any of the points $C^a_\delta(q, q'), C^b_\delta(q, q''), r_q$ coincide (and thus the order might possibly change).[10] By

[10]The important critical values for this step are the $O(|Q|^3) = O(n^6)$ radii of points with three (or more) points of Q on their boundary. See Figure 3 for an illustration of all types of critical values.

testing for these $O(1)$ critical values of δ, we can determine the order of $C^a_{\delta^*}(q,q'), C^b_{\delta^*}(q,q''), r_q$ on the boundary of $D_\delta(q)$, and thus resolve a comparison of $C^a_{\delta^*}(q,q')$ and $C^b_{\delta^*}(q,q'')$ in the order of $L_{\delta^*}(q)$ using only a constant number of calls to the decision algorithm.

Note that by arbitrarily choosing an order of Q, we may use Cole's sorting procedure (Lemma 3.6) to construct all lists $L_{\delta^*}(q), q \in Q$ simultaneously (we simply need to adapt the comparison function to compare $C^a_{\delta^*}(q,q'), C^b_{\delta^*}(\tilde{q},q'')$ according to the order of Q if $q \neq \tilde{q}$). Note that in this application of Lemma 3.6, we have $N = \sum_{q\in Q} |C_{\delta^*}(q)| = O(|Q|^2) = O(n^4)$.

It follows that in time $O((n^4 + T_{\text{dec}}(n)) \log n)$, we can obtain an interval $(\alpha_2, \beta_2]$ such that $\delta^* \in (\alpha_2, \beta_2]$, while β_2 is the only value for δ for which the combinatorial structure of $\mathcal{A}_\delta$ changes in $(\alpha_2, \beta_2]$. Thus, $\delta^* = \beta_2$, as desired.

The overall running time of the above procedure amounts to $O((n^4 + T_{\text{dec}}(n)) \log n)$, which concludes the proof of Lemma 3.4.

4 GRID REACHABILITY DATA STRUCTURE

In this section, we prove Lemma 3.3, which we restate here for convenience.

Lemma 3.3 (Grid Reachability Data Structure). *Given an $n \times n$ matrix M over $\{0,1\}$ and a set of* terminals $\mathcal{T} \subseteq [n] \times [n]$ *of size $k > 0$, there is a data structure $\mathcal{D}_{M,\mathcal{T}}$ with the following properties.*

(i) (Construction:) We can construct $\mathcal{D}_{M,\mathcal{T}}$ in time $O(n^2 + k \log^2 n)$.
(ii) (Reachability Query:) Given $F \subseteq \mathcal{T}$, we can determine in time $O(k \log^3 n)$ whether there is a monotone path from $(1,1)$ to (n,n) using only positions (i,j) with $M_{i,j} = 1$ or $(i,j) \in F$.
(iii) (Update:) Given $\mathcal{T}' \subseteq [n] \times [n]$ of size k and an $n \times n$ matrix M' over $\{0,1\}$ differing from M in at most k positions, we can update $\mathcal{D}_{M,\mathcal{T}}$ to $\mathcal{D}_{M',\mathcal{T}'}$ in time $O(n\sqrt{k} \log n + k \log^2 n)$. Here, we assume M' to be represented by the set Δ of positions in which M and M' differ.

The rough outline is as follows: We obtain the data structure by repeatedly splitting the free-space diagram into smaller blocks. This yields $O(\log n)$ levels of blocks, where in each block we store reachability information from all "inputs" to the block (i.e., the lower-left boundary) to all "outputs" of the block (i.e., the upper-right boundary). Any change in the matrix M is reflected only in $O(\log n)$ blocks containing this position, thus, we can quickly update the information. This approach was pursued already by Ben Avraham et al. [12].

In addition, however, we need to maintain reachability of all terminals $\mathcal{T}$ to the inputs and from the outputs of each block. Surprisingly, we only need an additional storage of $O(|\mathcal{T}|)$ per block. We show how to maintain this information also under updates and how it can be used by a divide and conquer approach to answer any reachability queries.

To this end, we start with some basic definitions (block structure, identifiers for each position, etc.) in Section 4.1. We can then prove the succinct characterization of terminal reachability in Section 4.2, which is the key aspect of our data structure. Given this information, we can define exactly what information we store for each block in Section 4.3. We give algorithms computing the information for some block given the information for its children in Section 4.4, which allows us to prove the initialization and update statements (i.e., ((i) and (iii) of Lemma 3.3) in Section 4.5. Finally, Section 4.6 is devoted to the reachability queries, i.e., proving ((ii) of Lemma 3.3).

4.1 Basic Structures and Definitions

Without loss of generality, we may assume that $n = 2^\kappa + 1$ for some integer $\kappa \in \mathbb{N}$. Otherwise, for any $n \times n$ matrix M over $\{0,1\}$, we could define an $n' \times n'$ matrix M' with (1) $n' = 2^\kappa + 1$ for some $\kappa \in \mathbb{N}$ with $n < n' \leq 2n$ and (2) setting $M'_{i,j} = M_{i,j}$ for all $(i,j) \in [n] \times [n]$ and setting $M'_{i,j} = 1$ if

and only if $i = j$ for all $(i, j) \in [n'] \times [n'] \setminus [n] \times [n]$. Clearly, existence of a monotone 1-path from $(1, 1)$ to (n, n) in M is equivalent to existence of a monotone 1-path from $(1, 1)$ to (n', n') in M'.

Canonical blocks. Let I, J be intervals in $[n]$ with $n = 2^\kappa + 1$. We call $I \times J \subseteq [n] \times [n]$ a *block*. In particular, we only consider blocks obtained by splitting the square $[n]\times[n]$ alternately horizontally and vertically until we are left with 2×2 blocks. Formally, we define $\mathcal{B}_0 := \{([n], [n])\}$ and construct $\mathcal{B}_{\ell+1}$ inductively by splitting each block $B \in \mathcal{B}_\ell$ as follows:

- For $\ell = 2i$ with $0 \le i < \kappa$, we have $B = (I, J)$ with $|I| = |J| = 2^{\kappa-i} + 1$. We split J into intervals J_1, J_2, where J_1 contains the first $(2^{\kappa-i-1} + 1)$ elements in J and J_2 contains the last $(2^{\kappa-i-1} + 1)$ elements in J (thus J_1 and J_2 intersect in the middle element of J). Add (I, J_1) and (I, J_2) to $\mathcal{B}_{\ell+1}$.
- For $\ell = 2i + 1$ with $0 \le i < \kappa$, we have $B = (I, J)$ with $|I| = 2^{\kappa-i} + 1$ and $|J| = 2^{\kappa-i-1} + 1$. Analogous to above, we split I into two equal-sized intervals I_1, I_2, where I_1 contains the first $(2^{\kappa-i-1} + 1)$ elements in I and I_2 contains the last $(2^{\kappa-i-1} + 1)$ elements in I. Add (I_1, J) and (I_2, J) to $\mathcal{B}_{\ell+1}$.

We let $\mathcal{B} := \bigcup_{\ell=0}^{2\kappa} \mathcal{B}_\ell$ be the set of *canonical blocks*, and call each block $B \in \mathcal{B}_\ell$ a *canonical block on level* ℓ. The blocks $B_1 = (I_1, J), B_2 = (I_2, J) \in \mathcal{B}_{\ell+1}$ (or $B_1 = (I, J_1), B_2 = (I, J_2) \in \mathcal{B}_{\ell+1}$, respectively) obtained from $B = (I, J) \in \mathcal{B}_\ell$ are called the *children* of B. See Figure 4.

Boundaries. For any $B = (I, J) \in \mathcal{B}$, we denote the lower left boundary of B as $B^- = \{\min I\}\times J \cup I \times \{\min J\}$, and call each $p \in B^-$ an *input* of B. Analogously, we denote the upper right boundary of B as $B^+ = \{\max I\} \times J \cup I \times \{\max J\}$ and call each $q \in B^+$ an *output* of B. By slight abuse of notation, we define $|\partial B| = |B^- \cup B^+|$ as the size of the boundary of B, i.e., the number of inputs and outputs of B.

If B splits into children B_1, B_2, then we call $B^{\text{mid}} = B_1^+ \cap B_2^-$ the *splitting boundary* of B.

Indices. To prepare the description of this information, we first define, for technical reasons, *indices* for all positions in $[n] \times [n]$. It allows us to give each position a unique identifier with the property that for any canonical block B, the indices yield a local ordering of the boundaries.

Observation 4.1. *Let* $\text{ind} : [n] \times [n] \to \mathbb{N}$, *where for any point* $p = (x, y) \in [n] \times [n]$, *we set* $\text{ind}(p) := (y-x)(2n)+x$. *We call* $\text{ind}(p)$ *the* index of p. *This function satisfies the following properties:*

(1) The function ind *is injective, can be computed in constant time, and given* $i = \text{ind}(p)$, *we can determine* $\text{ind}^{-1}(i) := p$ *in constant time.*
(2) For any $B \in \mathcal{B}$, ind *induces an ordering of* B^+ *in counter-clockwise order and an ordering of* B^- *in clockwise order.*

We refer to Figure 5 for an illustration of a block B, its boundaries, and the indices of all positions.

4.2 Reachability Characterization

Our aim is to construct a data structure $\mathcal{D}_{M,\mathcal{T}} = (\mathcal{D}_{M,\mathcal{T}}(B))_{B\in\mathcal{B}}$, where $\mathcal{D}_{M,\mathcal{T}}(B)$ succinctly describes reachability (via monotone 1-paths) between the boundaries B^-, B^+ and the terminals $\mathcal{T}_B := \mathcal{T} \cap B$ inside B. In particular, we show that we only require space $O(|\partial B| + |\mathcal{T}_B|)$ to represent this information.

To prepare this, we start with a few simple observations that yield a surprisingly simple characterization of reachability from any terminal to the boundary.

Compositions of crossing paths. We say that we reach q from p, written $p \rightsquigarrow q$, if there is a traversal $T = (t_1, \dots, t_\ell)$ with $t_1 = p$, $t_\ell = q$, and t_i is free for all $1 < i < \ell$ (note that we do not require t_1 and t_ℓ to be free). We call such a slightly adapted notion of traversal a *reach traversal*. By

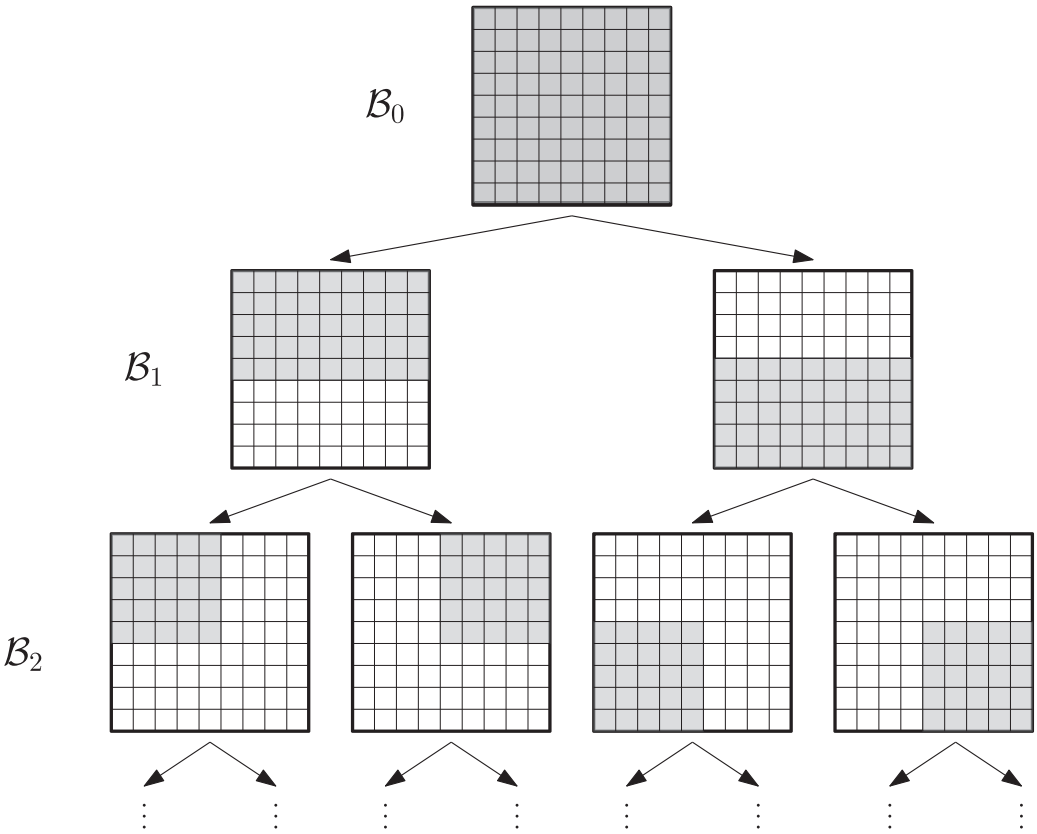


Fig. 4. The sets of canonical blocks $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{2\kappa}$. We alternate between horizontal and vertical splits. Note that child blocks overlap at their boundary.

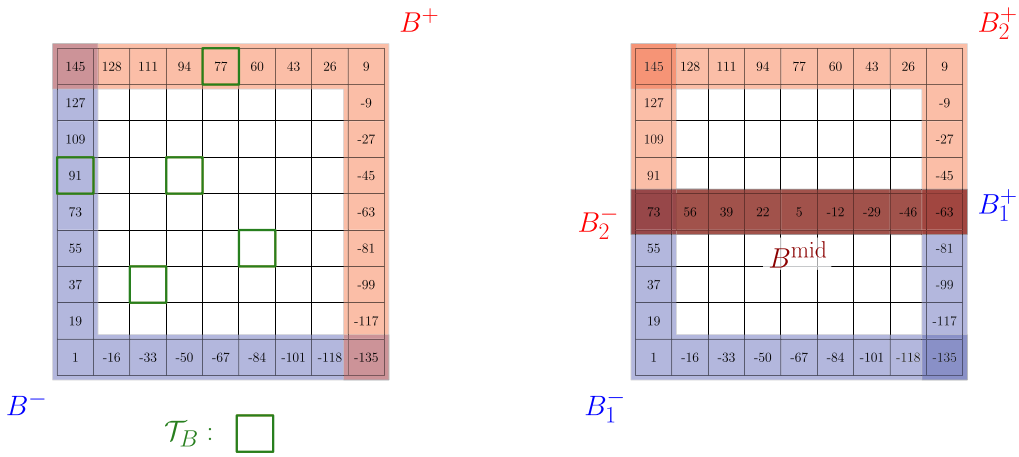


Fig. 5. Structure of a block B.

connecting the points of T by straight lines, we may view T also as a polygonal curve in $\mathbb{R}^2$. The following property is a standard observation for problems related to the Fréchet distance.

Observation 4.2. *Let T_1, T_2 be reach traversals from p_1 to q_1 and from p_2 to q_2, respectively. Then if T_1 and T_2 intersect, we have $p_1 \rightsquigarrow q_2$ (and, symmetrically, $p_2 \rightsquigarrow q_1$).*

Proof. Let $t \in [n] \times [n]$ be a free position in which T_1, T_2 intersect (observe that such a point with integral coordinates must exist unless $p_1 = p_2$ or $q_1 = q_2$; in the latter case, the claim is trivial). Note that t splits T_1, T_2 into $T_1 = T_1^a \circ T_1^b$ and $T_2 = T_2^a \circ T_2^b$ such that T_1^a, T_2^a are reach traversals ending in t and T_1^b, T_2^b are reach traversal starting in t. By concatenating T_1^a and T_2^b, we obtain a reach traversal from p_1 to q_2. Symmetrically, $T_2^a \circ T_1^b$ proves $p_2 \rightsquigarrow q_1$. □

Let $B \in \mathcal{B}$ and recall that $\mathrm{ind}(\cdot)$ orders B^+ counter-clockwise. For any $p \in B$, we define $\mathrm{A}(p) := \min\{\mathrm{ind}(q) \mid q \in B^+, p \rightsquigarrow q\}$, and analogously $\mathrm{Z}(p) := \max\{\mathrm{ind}(q) \mid q \in B^+, p \rightsquigarrow q\}$ (note that $\mathrm{A}(p)$ and $\mathrm{Z}(p)$ correspond to the lowest/rightmost and highest/leftmost pointer, respectively, in Reference [6, Section 3.2]). These two values define a corresponding *reachability interval* $\mathcal{I}(p) := [\mathrm{A}(p), \mathrm{Z}(p)]$ that contains all $q \in B^+$ with $p \rightsquigarrow q$. In the following analysis, we slightly abuse notation by also using $\mathrm{ind}(p)$ to denote the corresponding (unique) position $p \in [n] \times [n]$.

Definition 4.3. Let $p \in B$ with $\infty > \mathrm{A}(p), \mathrm{Z}(p) > -\infty$ and fix any reach traversals T_A, T_Z from p to $\mathrm{A}(p)$ and $\mathrm{Z}(p)$ such that we can write

$$T_A = P_{\mathrm{com}} \circ P'_A,$$
$$T_Z = P_{\mathrm{com}} \circ P'_Z,$$

for some polygonal curves $P_{\mathrm{com}}, P'_A, P'_Z$ with P'_A, P'_Z non-intersecting. Let $\mathcal{F}$ be the face enclosed by P'_A, P'_Z and the path from $\mathrm{A}(p)$ to $\mathrm{Z}(p)$ on B^+ (if $\mathrm{A}(p) = \mathrm{Z}(p)$, we let $\mathcal{F}$ be the empty set). We define the *reach region* of p as

$$\mathcal{R}(p) := \mathcal{F} \cup P_{\mathrm{com}}.$$

We refer to Figure 6 for an illustration. Observe that the desired traversals T_A, T_Z for defining $\mathcal{R}(p)$ always exist: For any reach traversals T'_A, T'_Z from p to $\mathrm{A}(p)$ and $\mathrm{Z}(p)$, respectively, consider the latest point in which T'_A, T'_B intersect, say, t. We can define reach traversals T_A and T_Z by following T'_A until t and then following the remainder of T'_A or T'_Z to reach $\mathrm{A}(p)$ or $\mathrm{Z}(p)$, respectively. These traversals satisfy the conditions by construction. (Strictly speaking, any feasible choice for T_A, T_Z gives a potentially different reach region $\mathcal{R}(p)$. However, any fixed choice will be sufficient for our proofs, e.g., choosing lexicographically smallest/largest traversals.)

The following property generalizes an insightful property of reachability from the inputs to the outputs (cf. Reference [6, Lemma 10] and Reference [12, Corollary 4.2]) to a similar property for reachability from arbitrary block positions to the outputs, using the same argument of crossing traversals.

Proposition 4.4. *Let $p, p' \in B$, $q \in B^+$ with $\mathrm{ind}(q) \in \mathcal{I}(p)$ and $p' \notin \mathcal{R}(p)$. Then $p' \rightsquigarrow q$ implies $p \rightsquigarrow q$.*

Proof. The claim holds trivially if $\mathrm{ind}(q) = \mathrm{A}(p)$ or $\mathrm{ind}(q) = \mathrm{Z}(p)$. Thus, we may assume that $\mathrm{A}(p) < \mathrm{Z}(p)$, which implies that the face $\mathcal{F}$ in $\mathcal{R}(p)$ is nonempty with $q \in \mathcal{F}$ and $p' \notin \mathcal{F}$. Hence, any reach traversal T from p' to q must cross the boundary of $\mathcal{F}$, in particular, the path $P(T_A)$ or $P(T_Z)$, where T_A, T_Z both originate in p. By Observation 4.2, this yields $p \rightsquigarrow q$. □

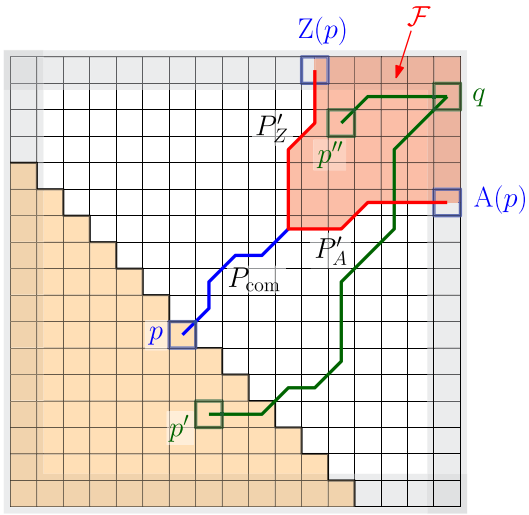


Fig. 6. Illustration of $\mathcal{R}(p)$, Proposition 4.4 and Lemma 4.5: Any reach traversal from $p' \notin \mathcal{R}(p)$ must cross P'_A or P'_Z to reach q. However, if $p'' \rightsquigarrow q$ but $p'' \in \mathcal{R}(p)$, then q might not be reachable from p. A sufficient condition for $p' \notin \mathcal{R}(p)$ is that $p' \neq p$ and $L(p') \leq L(p)$ (indicated by the orange triangular area).

Reachability Labelling. We define a total order on nodes in B that allows us to succinctly represent reachability towards B^+ for any subset $S \subseteq B$ in space $\tilde{O}(|S| + |B^+|)$. The key is a labelling $L : [n] \times [n] \to \mathbb{N}$, defined by $L((x, y)) = x + y$, that we call the *reachability labelling*. For an illustration of the following lemma, we refer to Figure 6.

Lemma 4.5. *Let $p = (x, y), p' = (x', y') \in B$ with $L(p') \leq L(p)$ and $q \in B^+$ with $\text{ind}(q) \in \mathcal{I}(p)$. Then $p' \rightsquigarrow q$ implies $p \rightsquigarrow q$.*

Proof. The proof idea is to show that $L(p') \leq L(p)$ implies that $p' \notin \mathcal{R}(p)$, and hence Proposition 4.4 shows the claim. Note that by monotonicity of reach traversals, any point $r = (r_x, r_y) \in \mathcal{R}(p)$ satisfies $r_x \geq x$ and $r_y \geq y$. Thus, $p' \in \mathcal{R}(p)$ only if $x' \geq x$, $y' \geq y$, but this together with $x' + y' = L(p') \leq L(p) = x + y$ implies $(x', y') = (x, y)$. Summarizing, we either have $p = p'$, which trivially satisfies the claim, or $p' \notin \mathcal{R}(p)$, which yields the claim by Proposition 4.4. □

For any $S \subseteq B$, this labelling enables a surprisingly succinct characterization of which terminals in S have reach traversals to which outputs in B^+ by the following lemma (greatly generalizing a simpler characterization[11] for the special case of $S = B^-$, cf. Reference [6, Lemma 10] and Reference [12, implicit in Lemma 4.4]). This is one of our key insights.

Corollary 4.6. *Let $q \in B^+$ and define $\ell(q) := \min\{L(p) \mid p \in B, p \rightsquigarrow q\}$. Then for any $p \in B$, we have*

$$p \rightsquigarrow q \qquad \text{if and only if} \qquad \text{ind}(q) \in \mathcal{I}(p) \text{ and } \ell(q) \leq L(p).$$

[11] In our language, this characterization is as follows: For any $p \in B^-$, $q \in B^+$, we have $p \rightsquigarrow q$ if and only if $\text{ind}(q) \in \mathcal{I}(p)$ and there is some $p' \in B^-$ with $p' \rightsquigarrow q$. It is easy to see that this characterization no longer holds if we replace B^- by an arbitrary subset $S \supseteq B^-$; our approach instead relies on the reachability labelling to obtain a succinct and algorithmically tractable characterization.

Proof. Clearly, $p \rightsquigarrow q$ implies, by definition of $\mathrm{A}(p), \mathrm{Z}(p)$, and $\ell(q)$, that $\mathrm{A}(p) \leq \mathrm{ind}(q) \leq \mathrm{Z}(p)$ and $\ell(q) \leq L(p)$.

Conversely, assume that $\mathrm{ind}(q) \in \mathcal{I}(p)$ and $\ell(q) \leq L(p)$. Take any $p' \in B$ with $p' \rightsquigarrow q$ and $\ell(q) = L(p')$. Thus, we have $L(p') = \ell(q) \leq L(p)$, $\mathrm{ind}(q) \in \mathcal{I}(p)$ and $p' \rightsquigarrow q$, which satisfies the requirements of Lemma 4.5, yielding $p \rightsquigarrow q$. □

Given this characterization, we obtain a highly succinct representation of reachability information. Specifically, to represent the information that terminals in S have reach traversals to which outputs in B^+, we simply need to store $\ell(q)$ for all $q \in B^+$ as well as the interval $\mathcal{I}(p)$ for all $p \in S$. Thus, the space required to store this information amounts to only $O(|\partial B| + |S|)$, which greatly improves over a naive $O(|\partial B| \cdot |S|)$-sized tabulation.

Reverse Information. By defining $L^{\mathrm{rev}}((x,y)) = -L((x,y)) = -x - y$, we obtain a labelling with symmetric properties. In particular, define $\mathrm{A}^{\mathrm{rev}}(q) := \min\{\mathrm{ind}(p) \mid p \in B^-, p \rightsquigarrow q\}$, $\mathrm{Z}^{\mathrm{rev}}(q) := \max\{\mathrm{ind}(p) \mid p \in B^-, p \rightsquigarrow q\}$ and the corresponding *reverse reachability interval* $\mathcal{I}^{\mathrm{rev}}(q) := [\mathrm{A}^{\mathrm{rev}}(q), \mathrm{Z}^{\mathrm{rev}}(q)]$. It is straightforward to prove the following symmetric variant of Corollary 4.6.

Corollary 4.7. *Let $p \in B^-$ and define $\ell^{\mathrm{rev}}(p) := \min\{L^{\mathrm{rev}}(q) \mid q \in B, p \rightsquigarrow q\}$. Then for any $q \in B$, we have*

$$p \rightsquigarrow q \qquad \text{if and only if} \qquad \mathrm{ind}(p) \in \mathcal{I}^{\mathrm{rev}}(q) \text{ and } \ell^{\mathrm{rev}}(p) \leq L^{\mathrm{rev}}(q).$$

Summary of Reachability Characterization. As a convenient reference, we collect here the main notation and results introduced in this section.

Summary 4.8. *For any $p \in B$, the* reachability interval $\mathcal{I}(p)$ *is defined as* $[\mathrm{A}(p), \mathrm{Z}(p)]$ *with*

$$\mathrm{A}(p) = \min\{\mathrm{ind}(q) \mid q \in B^+, p \rightsquigarrow q\},$$
$$\mathrm{Z}(p) = \max\{\mathrm{ind}(q) \mid q \in B^+, p \rightsquigarrow q\}.$$

(Note that $\mathcal{I}(p)$ might be empty if $\mathrm{A}(p) = \infty, \mathrm{Z}(p) = -\infty$.) For any $q \in B^+$, its reachability level $\ell(q)$ *is defined as*

$$\ell(q) = \min\{L(p) \mid p \in B, p \rightsquigarrow q\},$$

where $L((x,y)) = x + y$. For any $p \in B, q \in B^+$, we have the reachability characterization that

$$p \rightsquigarrow q \qquad \text{if and only if} \qquad \mathrm{ind}(q) \in \mathcal{I}(p) \text{ and } \ell(q) \leq L(p).$$

For any $q \in B$, we have the reverse reachability interval $\mathcal{I}^{\mathrm{rev}}(q) = [\mathrm{A}^{\mathrm{rev}}(q), \mathrm{Z}^{\mathrm{rev}}(q)]$ *with*

$$\mathrm{A}^{\mathrm{rev}}(q) = \min\{\mathrm{ind}(p) \mid p \in B^-, p \rightsquigarrow q\},$$
$$\mathrm{Z}^{\mathrm{rev}}(q) = \max\{\mathrm{ind}(p) \mid p \in B^-, p \rightsquigarrow q\}.$$

(Again, $\mathcal{I}^{\mathrm{rev}}(q)$ might be empty if $\mathrm{A}^{\mathrm{rev}}(q) = \infty, \mathrm{Z}^{\mathrm{rev}}(q) = -\infty$.) For any $p \in B^-$, its reverse reachability level $\ell^{\mathrm{rev}}(p)$ *is defined as*

$$\ell^{\mathrm{rev}}(p) = \min\{L^{\mathrm{rev}}(q) \mid q \in B, p \rightsquigarrow q\},$$

where $L^{\mathrm{rev}}((x,y)) = -x - y$. For any $p \in B^-, q \in B$, we have the reachability characterization that

$$p \rightsquigarrow q \qquad \text{if and only if} \qquad \mathrm{ind}(p) \in \mathcal{I}^{\mathrm{rev}}(q) \text{ and } \ell^{\mathrm{rev}}(p) \leq L^{\mathrm{rev}}(q).$$

4.3 Information Stored at Canonical Block B

Using the characterization given in Corollaries 4.6 and 4.7, we can now describe which information we need to store for any canonical block $B \in \mathcal{B}$.

Definition 4.9. Let $B \in \mathcal{B}$. The *information stored at B* (which we denote as $\mathcal{D}_{M,\mathcal{T}}(B)$) consists of the following information: First, we store *forward reachability information* consisting of,

- for every $p \in B^- \cup \mathcal{T}_B$, the interval $\mathcal{I}(p)$, and
- for every $q \in B^+$, the reachability level $\ell(q)$.

Symmetrically, we store *reverse reachability information* consisting of,

- for every $q \in B^+ \cup \mathcal{T}_B$, the interval $\mathcal{I}^{\text{rev}}(q)$, and
- for every $p \in B^-$, the reverse reachability level $\ell^{\text{rev}}(p)$.

Finally, if B has children $B_1, B_2 \in \mathcal{B}$, where B_1 is the lower or left sibling of B_2, then we additionally store

- an orthogonal range minimization data structure $\mathcal{OR}_B$ storing, for each *free* $q \in B^{\text{mid}} = B_1^+ \cap B_2^-$, the value $\ell_2^{\text{rev}}(q)$ under the key $(\text{ind}(q), \ell_1(q))$. Here, $\ell_1(q)$ denotes the forward reachability level in B_1, and $\ell_2^{\text{rev}}(q)$ denotes the reverse reachability level in B_2.

4.4 Computing Information at Parent From Information at Children

We show how to construct the information stored at the blocks quickly in a recursive fashion.

Lemma 4.10. *Let $B \in \mathcal{B}$ with children B_1, B_2. Given the information stored at B_1 and B_2, we can compute the information stored at B in time $O((|\partial B| + |\mathcal{T}_B|) \log |\partial B|)$.*

Proof. Without loss of generality, we assume that B_1, B_2 are obtained from B by a vertical split (the other case is analogous)—let $B_{\mathbf{l}}, B_{\mathbf{r}}$ denote the left and right child, respectively. As a convention, we equip the information stored at $B_{\mathbf{l}}, B_{\mathbf{r}}$ with the subscript $\mathbf{l}, \mathbf{r}$, respectively, and write the information stored at B without subscript. Furthermore, we let $B^{\text{mid}}_{\text{free}}$ denote the set of *free* positions of the splitting boundary $B^{\text{mid}} = B_{\mathbf{l}}^+ \cap B_{\mathbf{r}}^-$.

Computation of $\mathcal{I}(p)$. Let $p \in B^- \cup \mathcal{T}_B$ be arbitrary. We first explain how to compute $\text{A}(p)$ (see Figure 7 for an illustration). If $p \in B_{\mathbf{r}}$, then $\text{A}(p) = \text{A}_{\mathbf{r}}(p)$, since by monotonicity any $q \in B^+$ with $p \rightsquigarrow q$ satisfies $q \in B_{\mathbf{r}}^+$. Thus, it remains to consider $p \notin B_{\mathbf{r}}$.

We claim that for $p \notin B_{\mathbf{r}}$, we have $A(p) = \min\{A_1(p), A_2(p)\}$, where

$$A_1(p) \coloneqq \min_{\substack{q \in B^+ \cap B_{\mathbf{l}}, \\ p \rightsquigarrow q}} \text{ind}(q)$$

$$A_2(p) \coloneqq \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ p \rightsquigarrow j}} \; \min_{\substack{q \in B^+ \cap B_{\mathbf{r}}, \\ j \rightsquigarrow q}} \text{ind}(q).$$

Indeed, this follows, since each path starting in $p \in B_{\mathbf{l}}$ and ending in B^+ must end in $B_{\mathbf{l}}$, or cross B^{mid} at some free $j \in B^{\text{mid}}$ and end in $B_{\mathbf{r}}$.

To compute $A_1(p)$ note that Corollary 4.6 yields $A_1(p) = \min\{\text{ind}(q) \mid q \in B^+ \cap B_{\mathbf{l}}, \text{ind}(q) \in [\text{A}_{\mathbf{l}}(p), \text{Z}_{\mathbf{l}}(p)], \ell_{\mathbf{l}}(q) \le L(p)\}$, which can be expressed as an orthogonal range minimization query.

Likewise, to compute $A_2(p)$, note that $B^+ \cap B_{\mathbf{r}} = B_{\mathbf{r}}^+$. Thus,

$$\text{A}_2(p) = \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ p \rightsquigarrow j}} \; \min_{\substack{q \in B_{\mathbf{r}}^+, \\ j \rightsquigarrow q}} \text{ind}(q) = \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ p \rightsquigarrow j}} \text{A}_{\mathbf{r}}(j) = \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ \text{ind}(j) \in \mathcal{I}_{\mathbf{l}}(p), \ell_{\mathbf{l}}(j) \le L(p)}} \text{A}_{\mathbf{r}}(j),$$

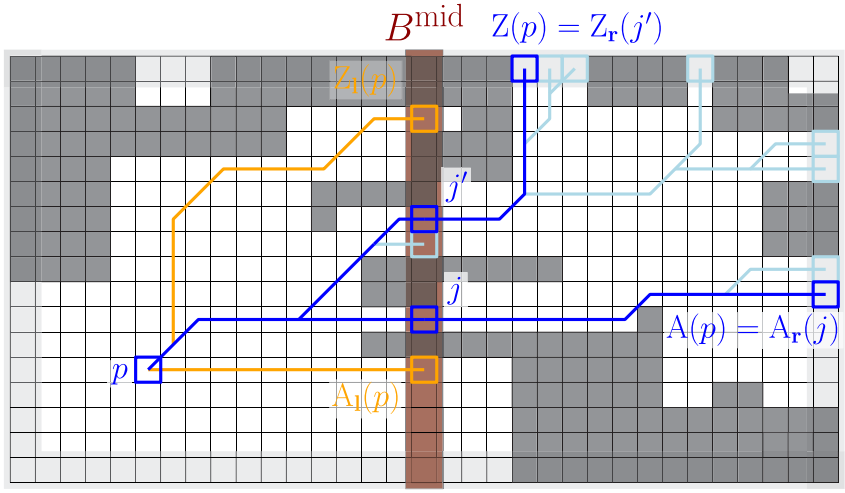


Fig. 7. Computation of $\mathcal{I}(p)$. To determine the smallest (largest) reachable index on $B^+ \cap B_{\mathbf{r}}$, we optimize, over all $j \in B^{\text{mid}}$ with $p \rightsquigarrow j$, the smallest (largest) reachable index $\mathrm{A}_{\mathbf{r}}(j)$ ($\mathrm{Z}_{\mathbf{r}}(j)$) on $B^+_{\mathbf{r}}$. In this diagram, bright (dark) cells show free (non-free) positions. B^{mid} is the boundary shared by B_l (left of B^{mid}) and B_r (right of B^{mid}); see Figure 5 as a reminder of how we split boxes.

where the second and last equalities follow from the definition of $\mathrm{A}_{\mathbf{r}}$ and Corollary 4.6, respectively. It follows that we can compute $\mathrm{A}_2(p)$ using a simple orthogonal range minimization query.

Switching the roles of minimization and maximization, we obtain the analogous statements for computing $\mathrm{Z}(p)$. We summarize the resulting algorithm for computing the reachability intervals $\mathcal{I}(p)$ for all $p \in B^- \cup \mathcal{T}_B$ formally in Algorithm 2. Its correctness follows from the arguments above.

Let us analyze the running time of Algorithm 2: Observe that $|B^{\text{mid}}_{\text{free}}| \leq |\partial B|$. Thus, we can construct the orthogonal range data structures $\mathcal{OR}_{\mathrm{A}}$, $\mathcal{OR}_{\mathrm{Z}}$, and $\mathcal{OR}_{\text{top}}$ in time $O(|\partial B| \log |\partial B|)$ (see Section 2.3). For each $p \in B^- \cup \mathcal{T}_B$, we perform at most a constant number of two-dimensional orthogonal range minimization/maximization queries, which takes time $O(\log |\partial B|)$, followed by constant-time computation. The total running time amounts to $O((|\partial B| + |\mathcal{T}_B|) \log |\partial B|)$.

Computation of $\ell(q)$. Let $q \in B^+$ be arbitrary. If $q \in B_{\mathbf{l}}$, then $\ell(q) = \ell_{\mathbf{l}}(p)$, since by monotonicity every $p \in B$ with $p \rightsquigarrow q$ is contained in $B_{\mathbf{l}}$. Thus, we may assume that $q \notin B_{\mathbf{l}}$.

We claim that for $q \notin B_{\mathbf{l}}$, we have $\ell(q) = \min\{\ell_1(q), \ell_2(q)\}$, where

$$\ell_1(q) \coloneqq \min_{\substack{p \in B_{\mathbf{r}}, \\ p \rightsquigarrow q}} L(p)$$

$$\ell_2(q) \coloneqq \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ j \rightsquigarrow q}} \; \min_{\substack{p \in B_{\mathbf{l}}, \\ p \rightsquigarrow j}} L(p).$$

Indeed, this follows, since each path starting in B and ending in $q \in B_{\mathbf{r}}$ must start in $B_{\mathbf{r}}$, or start in $B_{\mathbf{l}}$ and cross B^{mid} at some free $j \in B^{\text{mid}}$.

Observe that the definition of $\ell_1(q)$ coincides with the definition of $\ell_{\mathbf{r}}(q)$. Thus, it only remains to compute $\ell_2(q)$. We write

$$\ell_2(q) = \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ j \rightsquigarrow q}} \; \min_{\substack{p \in B_{\mathbf{l}}, \\ p \rightsquigarrow j}} L(p) = \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ j \rightsquigarrow q}} \ell_{\mathbf{l}}(j) = \min_{\substack{j \in B^{\text{mid}}_{\text{free}}, \\ \text{ind}(j) \in \mathcal{I}^{\text{rev}}_{\mathbf{r}}(q), \ell^{\text{rev}}_{\mathbf{r}}(j) \leq L^{\text{rev}}(q)}} \ell_{\mathbf{l}}(j),$$

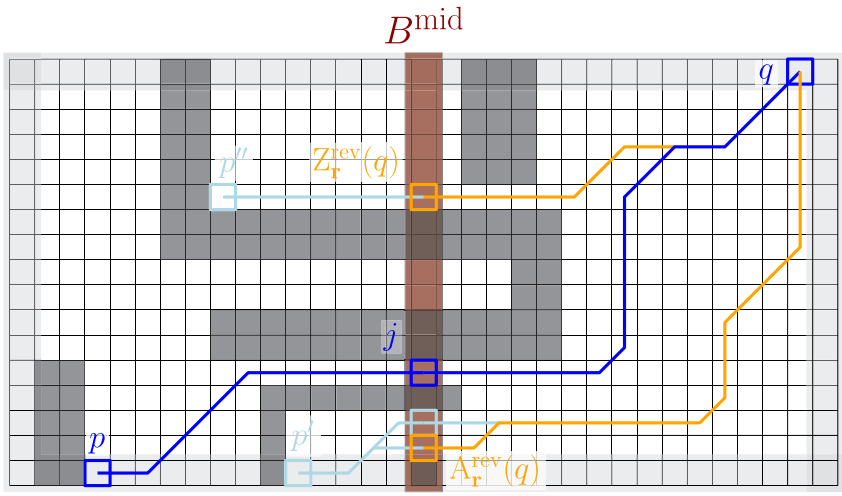


Fig. 8. Computation of $\ell(q)$. To determine the smallest label of a position in $B_{\mathbf{l}}$ reaching q, we optimize, over all $j \in B^{\text{mid}}$ with $j \rightsquigarrow q$, the smallest label $\ell_{\mathbf{l}}(j)$ of a position $p \in B_{\mathbf{l}}$ reaching j.

ALGORITHM 2: Computing $\mathcal{I}(p) = [\mathrm{A}(p), \mathrm{Z}(p)]$ for all $p \in B^- \cup \mathcal{T}_B$.

Build $\mathcal{OR}_{\mathrm{A}}$ storing $\mathrm{A}_{\mathbf{r}}(j)$ under the key $(\mathrm{ind}(j), \ell_{\mathbf{l}}(j))$ for all $j \in B^{\text{mid}}_{\text{free}}$ (for minimization queries)
Build $\mathcal{OR}_{\mathrm{Z}}$ storing $\mathrm{Z}_{\mathbf{r}}(j)$ under the key $(\mathrm{ind}(j), \ell_{\mathbf{l}}(j))$ for all $j \in B^{\text{mid}}_{\text{free}}$ (for maximization queries)
Build $\mathcal{OR}_{\text{top}}$ storing $\mathrm{ind}(q)$ under the key $(\mathrm{ind}(q), \ell_{\mathbf{l}}(q))$ for all $q \in B^+ \cap B_{\mathbf{l}}$ (for both queries)
for $p \in (B^- \cup \mathcal{T}_B)$ **do**
    if $p \in B_{\mathbf{r}}$ **then**
        $\mathcal{I}(p) \leftarrow \mathcal{I}_{\mathbf{r}}(p)$
    else
        $\mathrm{A}_1(p) \leftarrow \mathcal{OR}_{\text{top}}.\min([\mathrm{A}_{\mathbf{l}}(p), \mathrm{Z}_{\mathbf{l}}(p)] \times (-\infty, L(p)])$
        $\mathrm{A}_2(p) \leftarrow \mathcal{OR}_{\mathrm{A}}.\min([\mathrm{A}_{\mathbf{l}}(p), \mathrm{Z}_{\mathbf{l}}(p)] \times (-\infty, L(p)])$
        $\mathrm{A}(p) \leftarrow \min\{\mathrm{A}_1(p), \mathrm{A}_2(p)\}$

        $\mathrm{Z}_1(p) \leftarrow \mathcal{OR}_{\text{top}}.\max([\mathrm{A}_{\mathbf{l}}(p), \mathrm{Z}_{\mathbf{l}}(p)] \times (-\infty, L(p)])$
        $\mathrm{Z}_2(p) \leftarrow \mathcal{OR}_{\mathrm{Z}}.\max([\mathrm{A}_{\mathbf{l}}(p), \mathrm{Z}_{\mathbf{l}}(p)] \times (-\infty, L(p)])$
        $\mathrm{Z}(p) \leftarrow \max\{\mathrm{Z}_1(p), \mathrm{Z}_2(p)\}$

where the second and last equalities follow from the definition of $\ell_{\mathbf{l}}(j)$ and Corollary 4.7, respectively. It follows that we can compute $\ell_2(p)$ using a simple orthogonal range minimization query. For an illustration of $\ell_2(q)$, we refer to Figure 8.

We summarize the resulting algorithm for computing the reachability levels $\ell(q)$ for all $q \in B^+$ formally in Algorithm 3. Its correctness follows from the arguments above.

To analyze the running time of Algorithm 3, observe that $|B^{\text{mid}}_{\text{free}}| \leq |\partial B|$. Thus, we can construct $\mathcal{OR}_\ell$ in time $O(|\partial B| \log |\partial B|)$ (see Section 2.3). For each $q \in B^+$, we then perform at most one minimization query to $\mathcal{OR}_\ell$ in time $O(\log |\partial B|)$, followed by a constant-time computation. Thus, the total running time amounts to $O(|\partial B| \log |\partial B|)$.

ALGORITHM 3: Computing $\ell(q)$ for all $q \in B^+$.

```
Build OR_ℓ storing ℓ_l(j) under the key (ind(j), ℓ_r^rev(j)) for all j ∈ B_free^mid (for minimization queries)
for q ∈ B^+ do
    if q ∈ B_l then
        ℓ(q) ← ℓ_l(q)
    else
        ℓ_2(q) ← OR_ℓ.min([A_r^rev(q), Z_r^rev(q)] × (−∞, L^rev(q)])
        ℓ(q) ← min{ℓ_r(q), ℓ_2(q)}
```

Computation of reverse information. Switching the direction of reach traversals (which switches roles of inputs and outputs, B_l and B_r, etc.) as well as L and L^rev, we can use the same algorithms to compute the reverse reachability information in the same running time of $O((|\partial B| + |\mathcal{T}_B|) \log |\partial B|)$.

Computation of $\mathcal{OR}_B$. Finally, we need to construct the two-dimensional orthogonal range minimization data structure $\mathcal{OR}_B$: Recall that $\mathcal{OR}_B$ stores, for each $q \in B^\mathrm{mid}_\mathrm{free}$, the value $\ell^\mathrm{rev}_\mathrm{r}(q)$ under the key $(\mathrm{ind}(q), \ell_\mathrm{l}(q))$ for minimization queries. Since $|B^\mathrm{mid}_\mathrm{free}| \leq |\partial B|$, this can be done in time $O(|\partial B| \log |\partial B|)$ (cf. Section 2.3).

Summary. In summary, we can compute the information stored at B (according to Definition 4.9) from the information stored at B_1 and B_2 in time $O((|\partial B| + |\mathcal{T}_B|) \log |\partial B|)$, as desired. □

4.5 Initialization and Updates

We show how to construct our reachability data structure (using Lemma 4.10 that shows how to compute the information stored at some canonical block B given the information stored at both children). Specifically, the following lemma proves ((i) of Lemma 3.3.

Lemma 4.11. *We can construct $\mathcal{D}_{M,\mathcal{T}}$ in time $O(n^2 + |\mathcal{T}| \log^2 n)$.*

Proof. We use the obvious recursive algorithm to build $\mathcal{D}_{M,\mathcal{T}}$ in a bottom-up fashion using Lemma 4.10. Recall that $n = 2^\kappa + 1$ for some $\kappa \in \mathbb{N}$. Note that for the blocks $B \in \mathcal{B}_{2\kappa}$ in the lowest level, we can compute the information stored in B in constant time, which takes time $O(|\mathcal{B}_{2\kappa}|) = O(n^2)$ in total.

It remains to bound the running time to compute $\mathcal{D}_{M,\mathcal{T}}(B)$ for $B \in \mathcal{B}_\ell$ for $0 \leq \ell < 2\kappa$. Observe that this running time is bounded by $O(\sum_{\ell=0}^{2\kappa-1} \sum_{B \in \mathcal{B}_\ell} c_B)$ by Lemma 4.10, where $c_B := |\partial B| \log |\partial B| + |\mathcal{T}_B| \log |\partial B|$.

Let $0 \leq \ell < 2\kappa$. By construction, we have $|\mathcal{B}_\ell| = 2^\ell$. Furthermore, for any $B \in \mathcal{B}_\ell$, observe that its side lengths are bounded by $2^{\kappa - \lfloor \ell/2 \rfloor} + 1$, and thus $|\partial B| \leq 4 \cdot 2^{\kappa - \lfloor \ell/2 \rfloor} \leq 2^{\kappa - \ell/2 + 3}$. Hence, we may compute

$$\begin{aligned}
\sum_{\ell=0}^{2\kappa-1} \sum_{B \in \mathcal{B}_\ell} |\partial B| \log |\partial B| &\leq \sum_{\ell=0}^{2\kappa-1} |\mathcal{B}_\ell| 2^{\kappa - \ell/2 + 3} (\kappa - \ell/2 + 3) \\
&= \sum_{\ell=0}^{2\kappa-1} 2^{\kappa + \ell/2 + 3} (\kappa - \ell/2 + 3) \qquad (1) \\
&\leq 2 \left(\sum_{i=0}^{\kappa} 2^{\kappa + i + 3} (\kappa - i + 3) \right) \\
&= 2(2^{2(\kappa+3)} - 2^{\kappa+3}(\kappa + 5)) = O(2^{2\kappa}) = O(n^2).
\end{aligned}$$

Furthermore, we have

$$\sum_{\ell=0}^{2\kappa-1} \sum_{B\in\mathcal{B}_\ell} |\mathcal{T}_B| \log |\partial B| \leq \sum_{\ell=0}^{2\kappa-1} 4|\mathcal{T}|(\kappa - \ell/2 + 3) = O(|\mathcal{T}|\kappa^2) = O(|\mathcal{T}| \log^2 n),$$

where we used that $\sum_{B\in\mathcal{B}_\ell} |\mathcal{T}_B| \leq 4|\mathcal{T}|$ (as any position in $[n] \times [n]$ is shared by at most 4 blocks at the same level). In total, we obtain a running time bound of $O(n^2 + |\mathcal{T}| \log^2 n)$, as desired. □

With very similar arguments, we can prove ((iii) of Lemma 3.3).

Lemma 4.12. *Let M, M' be any $n\times n$ 0-1-matrices differing in at most k positions and $\mathcal{T}, \mathcal{T}' \subseteq [n]\times[n]$ be any sets of terminals of size k. Given the data structure $\mathcal{D}_{M,\mathcal{T}}$, the set $\mathcal{T}'$, as well as the set Δ of positions in which M and M' differ, we can update $\mathcal{D}_{M,\mathcal{T}}$ to $\mathcal{D}_{M',\mathcal{T}'}$ in time $O(n\sqrt{k}\log n + k\log^2 n)$.*

Proof. Set $X := \Delta \cup \mathcal{T} \cup \mathcal{T}'$ and note that $|X| = O(k)$. Observe that for any $B \in \mathcal{B}$ with $B \cap X = \emptyset$, we have $\mathcal{D}_{M,\mathcal{T}}(B) = \mathcal{D}_{M',\mathcal{T}'}(B)$, since the information stored at this block does not depend on any changed entry in M and does not contain any of the old or new terminals. Thus, we only need to update $\mathcal{D}_{M,\mathcal{T}}(B)$ to $\mathcal{D}_{M',\mathcal{T}'}(B)$ for all $B \in \mathcal{B}$ with $B\cap X \neq \emptyset$. We do this by computing the information for these blocks in a bottom-up fashion analogously to Lemma 4.11. Specifically, for any lowest-level block $B \in \mathcal{B}_{2\kappa}$ with $B \cap X \neq \emptyset$, we can compute the information stored in B in constant time. Since there are at most $4|X|$ such blocks, this step takes time $O(|X|) = O(k)$ in total.

It remains to bound the running time to compute $\mathcal{D}_{M,\mathcal{T}}(B)$ for $B \in \mathcal{B}_\ell$ with $B \cap X \neq \emptyset$, where $0 \leq \ell < 2\kappa$. For any such B, we let again $c_B := |\partial B| \log |\partial B| + |\mathcal{T}_B| \log |\partial B|$. Observe that the running time for the remaining task is thus bounded by $O(\sum_{\ell=0}^{2\kappa-1} \sum_{B\in\mathcal{B}_\ell, B\cap X\neq\emptyset} c_B)$ by Lemma 4.10.

We do a case distinction into $0 \leq \ell < \bar{\ell}$ and $\bar{\ell} \leq \ell < 2\kappa$ where $\bar{\ell} := \lfloor \log k \rfloor$. For the first case, we bound

$$\begin{aligned}
\sum_{\ell=0}^{\bar{\ell}-1} \sum_{\substack{B\in\mathcal{B}_\ell,\\ B\cap X\neq\emptyset}} |\partial B| \log |\partial B| &\leq \sum_{\ell=0}^{\bar{\ell}-1} \sum_{B\in\mathcal{B}_\ell} |\partial B| \log |\partial B| \\
&\leq \sum_{i=0}^{\bar{\ell}-1} 2^{\kappa+i/2+3}(\kappa - i/2 + 3) \\
&\leq \left(\sum_{i=0}^{\bar{\ell}-1} 2^{i/2}\right) 2^{\kappa+3}\kappa = (1+\sqrt{2})(2^{\bar{\ell}/2}-1)2^{\kappa+3}\kappa = O(\sqrt{k}n\log n),
\end{aligned}$$

where the second inequality is derived as in Equation (1). Recall that for any $0 \leq \ell < 2\kappa$, there are at most $4|X|$ blocks $B \in \mathcal{B}_\ell$ with $B \cap X \neq \emptyset$ and for any $B \in \mathcal{B}_\ell$, we have $|\partial B| \leq 2^{\kappa-\ell/2+3}$. We compute

$$\begin{aligned}
\sum_{\ell=\bar{\ell}}^{2\kappa-1} \sum_{\substack{B\in\mathcal{B}_\ell,\\ B\cap X\neq\emptyset}} |\partial B| \log |\partial B| &\leq \sum_{\ell=\bar{\ell}}^{2\kappa-1} 4|X| 2^{\kappa-\ell/2+3}(\kappa - \ell/2 + 3) \\
&\leq 4|X| 2^{\kappa-\bar{\ell}/2+3}(\kappa+3) \cdot \sum_{\ell=0}^{2\kappa-\bar{\ell}-1} 2^{-\ell/2} \\
&= O\left(|X| \frac{n}{\sqrt{k}} \log n\right) = O(\sqrt{k}n\log n).
\end{aligned}$$

Furthermore, as in the proof of Lemma 4.11, we again compute

$$\sum_{\ell=0}^{2\kappa-1} \sum_{\substack{B\in\mathcal{B}_\ell,\\ B\cap X\neq\emptyset}} |\mathcal{T}_B| \log |\partial B| \leq \sum_{\ell=0}^{2\kappa-1} 4|\mathcal{T}|(\kappa - \ell/2 + 3) = O(|\mathcal{T}|\kappa^2) = O(|\mathcal{T}| \log^2 n).$$

Thus, in total we obtain a running time of $O(k + n\sqrt{k}\log n + |\mathcal{T}|\log^2 n) = O(n\sqrt{k}\log n + k\log^2 n)$. □

4.6 Reachability Queries

It remains to show how to use the information stored at all canonical blocks to answer reachability queries quickly. Specifically, the following lemma proves ((ii) of Lemma 3.3).

Lemma 4.13. *Given $\mathcal{D}_{M,\mathcal{T}} = (\mathcal{D}_{M,\mathcal{T}}(B))_{B\in\mathcal{B}}$, we can answer reachability queries for $F \subseteq \mathcal{T}$ in time $O(|\mathcal{T}|\log^3 n)$.*

Recall that we aim to determine whether there is a monotone path in M using only positions (i, j) with $M_{i,j} = 1$ or $(i, j) \in F$, i.e., we view F as a set of *free terminals* (typically, $(i, j) \in F$ is a *non-free* position). In this section, we assume, without loss of generality, that $(1, 1), (n, n) \in \mathcal{T}_B$ (whenever we construct/update to the data structure $\mathcal{D}_{M,\mathcal{T}}$, we may construct/update to $\mathcal{D}_{M,\mathcal{T}\cup\{(1,1),(n,n)\}}$ in the same asymptotic running time).

For any block $B \in \mathcal{B}$, $S \subseteq F \subseteq \mathcal{T}_B$, we define the function Reach(B, S, F) that returns the set

$$R := \{t \in F \mid \exists f_1, \ldots, f_\ell \in F : f_1 \in S, f_\ell = t, f_1 \rightsquigarrow f_2 \rightsquigarrow \cdots \rightsquigarrow f_\ell\},$$

i.e., we interpret S as a set of admissible *starting positions* for a reach traversal and ask for the set of positions reachable from S using only free positions or free terminals. We call any such position *F-reachable from S*. (Recall that in the definition of $p \rightsquigarrow q$, only the intermediate points on a reach traversal from p and q are required to be free, while the endpoints p and q are allowed to be non-free.)

We show that Reach(B, S, F) can be computed in time $O(|\mathcal{T}_B|\log^3 n)$. Given this, we can answer any reachability query in the same asymptotic running time: The reachability query asks whether there is a sequence $f_1, \ldots, f_\ell \in F \cup \{(1, 1), (n, n)\}$ such that (i) $f_1 = (1, 1)$ and $f_\ell = (n, n)$, (ii) both $(1, 1)$ and (n, n) are free positions or contained in F and (iii) $f_1 \rightsquigarrow f_2 \rightsquigarrow \cdots \rightsquigarrow f_\ell$. Since (ii) can be checked in constant time, it remains to determine whether

$$(n, n) \in \text{Reach}([n] \times [n],\ \{(1, 1)\},\ F \cup \{(1, 1), (n, n)\}).$$

4.6.1 Computation of Reach(B, S, F). To compute *Reach*(B, S, F), we work on the recursive block structure of $\mathcal{D}_{M,\mathcal{T}}$. Specifically, consider any canonical block $B \in \mathcal{B}$ (containing some free terminal) with children B_1, B_2. The (somewhat simplified) approach is the following: We first (recursively) determine all free terminals that are F-reachable from S in B_1 and call this set R_1. Then, we determine all free terminals in B_2 that are (directly) reachable from R_1 and call this set T_2. Finally, we (recursively) determine all free terminals in B_2 that are F-reachable from $T_2 \cup (S \cap B_2)$ and call this set R_2. The desired set of free terminals that are F-reachable from S is then $R_1 \cup R_2$. The main challenge in this process is the computation of the set T_2; this task is solved by the following lemma.

Lemma 4.14. *Let $B \in \mathcal{B}$ be a block with children B_1, B_2. Given $S \subseteq B_1 \setminus B^{\text{mid}}$ and $F \subseteq B_2 \setminus B^{\text{mid}}$ with $S, F \subseteq \mathcal{T}_B$, we can compute the set*

$$T = \{t \in F \mid \exists s \in S : s \rightsquigarrow t\}.$$

in time $O(|\mathcal{T}_B|\log^2 n)$. We call this procedure SingleStepReach(B, S, F).

We postpone the proof of this lemma to Section 4.6.2 and first show how this yields an algorithm for Reach, and thus, for reachability queries.

Proof of Lemma 4.13. We claim that Algorithm 4 computes R in time $O(|\mathcal{T}| \log^3 n)$.

ALGORITHM 4: Computing Reach(B, S, F) for $B \in \mathcal{B}$, $S \subseteq F \subseteq \mathcal{T}_B$.

```
1:  function Reach(B, S, F)
2:      if F = ∅ then
3:          return ∅
4:      else if B is a 2 × 2 block then
5:          Compute R by checking all possibilities
6:          return R
7:      else                                   ▷ B splits into child blocks B_1, B_2
8:          S_1 ← S ∩ B_1, S_2 ← S ∩ B_2
9:          R_1 ← Reach(B_1, S_1, F ∩ B_1)
10:         T_2 ← SingleStepReach(B, R_1 \ B^mid, F \ B_1)
11:         R_2 ← Reach(B_2, S_2 ∪ T_2 ∪ (R_1 ∩ B^mid), F ∩ B_2)
12:         return R_1 ∪ R_2
```

To ease the analysis, we introduce the shorthand that $s \rightsquigarrow_F t$ if and only if there are $f_1, \dots, f_\ell \in F$ with $f_1 = s$, $f_\ell = t$ and $f_1 \rightsquigarrow f_2 \rightsquigarrow \cdots \rightsquigarrow f_\ell$, i.e., t is F-reachable from s.

We show that Algorithm 4 computes $R = \{t \in F \mid \exists s \in S, s \rightsquigarrow_F t\}$ inductively: The base case for 2×2 blocks B holds trivially. Otherwise, by inductive assumption, we have

$$R_1 = \{t \in F \cap B_1 \mid \exists s \in S \cap B_1, s \rightsquigarrow_{F \cap B_1} t\}.$$

Note that by definition of SingleStepReach, we furthermore have

$$T_2 = \{t \in F \setminus B_1 \mid \exists s \in R_1 \setminus B^{\text{mid}}, s \rightsquigarrow t\}.$$

Finally, by inductive assumption,

$$\begin{aligned} R_2 = \{&t \in F \cap B_2 \mid \exists s \in S \cap B_2, s \rightsquigarrow_{F \cap B_2} t\} \cup \\ \{&t \in F \cap B_2 \mid \exists s \in T_2, s \rightsquigarrow_{F \cap B_2} t\} \cup \\ \{&t \in F \cap B_2 \mid \exists s \in R_1 \cap B^{\text{mid}}, s \rightsquigarrow_{F \cap B_2} t\}. \end{aligned}$$

First, we show that any $t \in R_1 \cup R_2$ is contained in R: If $t \in R_1$, then there is some $s \in S \cap B_1 \subseteq S$ with $s \rightsquigarrow_{F \cap B_1} t$ (trivially implying $s \rightsquigarrow_F t$), and thus $t \in R$. Likewise, if $t \in T_2$, then there is some $t' \in R_1 \setminus B^{\text{mid}}$ with $t' \rightsquigarrow t$. Since $t' \in R_1$, there must exist some $s \in S$ with $s \rightsquigarrow_F t'$. Thus, $s \rightsquigarrow_F t'$ and $t' \rightsquigarrow t$ yields $s \rightsquigarrow_F t$ and $t \in R$. Finally, if $t \in R_2$, then there exists some t' with $t' \rightsquigarrow_{F \cap B_2} t$ and either $t' \in S \cap B_2$, $t' \in T_2$, or $t' \in R_1 \cap B^{\text{mid}}$. In all these cases, there is some $s \in S$ with $s \rightsquigarrow_F t'$. Hence, $s \rightsquigarrow_F t'$ and $t' \rightsquigarrow_{F \cap B_2} t$ imply $s \rightsquigarrow_F t$, placing t in R.

We proceed to show the converse direction that any $t \in R$ is contained in $R_1 \cup R_2$: Let $s \in S$ with $s \rightsquigarrow_F t$. If $t \in F \cap B_1$, then $s \rightsquigarrow_F t$ is equivalent to $s \rightsquigarrow_{F \cap B_1} t$ and $s \in S \cap B_1$ (by monotonicity). Thus, $t \in R_1$. It only remains to consider the case that $t \in F \setminus B_1$. If $s \in S \cap B_2$, then again my monotonicity $s \rightsquigarrow_{F \cap B_2} t$ must hold, which implies $t \in R_2$. Otherwise, we have $s \in S \setminus B_2$. Since additionally $t \in F \setminus B_1$, there must exist either (1) some $r \in F \cap B^{\text{mid}}$ with $s \rightsquigarrow_{F \cap B_1} r$ and $r \rightsquigarrow_{F \cap B_2} t$ or (2) some $t' \in F \setminus B_2$, $t'' \in F \setminus B_1$ with $s \rightsquigarrow_{F \cap B_1} t' \rightsquigarrow t'' \rightsquigarrow_{F \cap B_2} t$ (by monotonicity). For (1), note that $r \in R_1$ (as shown above), and thus $t \in R_2$. For (2), note that $t' \in R_1 \setminus B_2 = R_1 \setminus B^{\text{mid}}$ (as $s \in S \cap B_1, t' \in F \setminus B_2$ and $s \rightsquigarrow_{F \cap B_1} t'$), $t'' \in T_2$ (as $t' \in R_1 \setminus B^{\text{mid}}$, $t'' \in F \setminus B_1$ and $t' \rightsquigarrow t''$), and finally $t \in R_2$ (as $t'' \in T_2$ and $t'' \rightsquigarrow_{F \cap B_2} t$), as desired.

We analyze the running time of a call $\text{Reach}(B_0, S_0, F_0)$. Let $T(B) = O(|\mathcal{T}_B| \log^2 n)$ denote the running time of $\text{SingleStepReach}(B, S, F)$ for arbitrary S, F. Observe that the running time of $\text{Reach}(B_0, S_0, F_0)$ is bounded by

$$\sum_{\substack{B \in \mathcal{B}, \\ \mathcal{T}_B \neq \emptyset}} O(T(B)), \tag{2}$$

as for $\mathcal{T}_B = \emptyset$, we have $F \subseteq \mathcal{T}_B = \emptyset$, which is a base case of $\text{Reach}(\cdot)$. To bound the above term, fix any ℓ, and note that for any $t \in \mathcal{T}$, there are at most 4 level-ℓ blocks $B \in \mathcal{B}_\ell$ with $t \in \mathcal{T}_B$ (if t is on the boundary of some block $B \in \mathcal{B}_\ell$, then it is shared between different blocks; however, any position is shared by at most 4 blocks). Thus, $\sum_{B \in \mathcal{B}_\ell} |\mathcal{T}_B| \leq 4|\mathcal{T}|$. Thus, (2) is bounded by

$$\sum_{\ell=0}^{2\log(n-1)} \sum_{\substack{B \in \mathcal{B}_\ell, \\ \mathcal{T}_B \neq \emptyset}} O(|\mathcal{T}_B| \log^2 n) = \sum_{\ell=0}^{2\log(n-1)} O(|\mathcal{T}| \log^2 n) = O(|\mathcal{T}| \log^3 n). \qquad \square$$

4.6.2 Computation of SingleStepReach(B, S, F). It remains to prove Lemma 4.14 to conclude the proof of Lemma 4.13.

Proof of Lemma 4.14. Consider $B \in \mathcal{B}$. We only consider the case that B is split vertically (the other case is symmetric); let $B_\mathbf{l}, B_\mathbf{r}$ denote its left and right sibling, respectively. Let $S \subseteq B_\mathbf{l} \setminus B^{\text{mid}}$, $F \subseteq B_\mathbf{r} \setminus B^{\text{mid}}$ with $S, F \subseteq \mathcal{T}_B$ be arbitrary. We use notation (subscripts $\mathbf{l}, \mathbf{r}$, etc.) as in the proof of Lemma 4.10.

Observe that for any $s \in S, f \in F$, we have that $s \rightsquigarrow f$ if and only if there exists some $j \in B^{\text{mid}}_{\text{free}}$ with $s \rightsquigarrow j$ and $j \rightsquigarrow f$. To introduce some convenient conventions, let $J^{\text{mid}} = \{j_1, \ldots, j_N\}$, where $j_1, \ldots, j_N$ is the sorted sequence of $\text{ind}(q)$ with $q \in B^{\text{mid}}_{\text{free}}$. We call $J \subseteq J^{\text{mid}}$ an interval of J^{mid} if $J = \{j_a, j_{a+1}, \ldots, j_b\}$ for some $1 \leq a \leq b \leq N$ and write it as $J = [j_a, j_b]_{J^{\text{mid}}}$ (i.e., $[j_a, j_b]_{J^{\text{mid}}}$ simply disregards any indices in $[j_a, j_b]$ representing positions not in $B^{\text{mid}}_{\text{free}}$).

Consider any interval J of J^{mid} with the property that for all $s \in S$ we either have $J \cap \mathcal{I}_\mathbf{l}(s) = J$ or $J \cap \mathcal{I}_\mathbf{l}(s) = \emptyset$ and for all $f \in F$ we either have $J \cap \mathcal{I}^{\text{rev}}_\mathbf{r}(f) = J$ or $J \cap \mathcal{I}^{\text{rev}}_\mathbf{r}(f) = \emptyset$. We call such a J an *(S, F)-reach-equivalent* interval. Note that by splitting J^{mid} right before and right after all points $\text{A}(s), \text{Z}(s)$ with $s \in S$ and $\text{A}^{\text{rev}}(f), \text{Z}^{\text{rev}}(f)$ with $f \in F$, we obtain a partition of J^{mid} into (S, F)-reach-equivalent intervals $J_1, \ldots, J_\ell$ with $\ell = O(|S \cup F|) = O(|\mathcal{T}_B|)$.[12]

Claim 4.15. *Let J be an (S, F)-reach-equivalent interval J. Let R^J be the set of $t \in F$ reachable from S via J, i.e., $R^J := \{t \in F \mid \exists s \in S, j \in J : s \rightsquigarrow j \rightsquigarrow t\}$. Define*

$$\ell^J := \min_{\substack{j \in J, \\ \exists s \in S : s \rightsquigarrow j}} \ell^{\text{rev}}_\mathbf{r}(j).$$

We have

$$R^J = \{t \in F \mid J \subseteq \mathcal{I}^{\text{rev}}_\mathbf{r}(t), \ell^J \leq L^{\text{rev}}(t)\}. \tag{3}$$

Proof. See Figure 9 for an illustration. Indeed, for any $t \in F$ with $J \subseteq \mathcal{I}^{\text{rev}}_\mathbf{r}(t)$ and $\ell^J \leq L^{\text{rev}}(t)$, consider any $j \in J$ with $\ell^{\text{rev}}_\mathbf{r}(j) = \ell^J$ and $s \rightsquigarrow j$ for some $s \in S$. Then, we have $j \in J \subseteq \mathcal{I}^{\text{rev}}_\mathbf{r}(t)$ and $\ell^{\text{rev}}_\mathbf{r}(j) = \ell^J \leq L^{\text{rev}}(t)$. Thus, by Corollary 4.7, $j \rightsquigarrow t$, which together with $s \rightsquigarrow j$ implies $s \rightsquigarrow j \rightsquigarrow t$, as desired. For the converse, let $t \in F$ with $s \rightsquigarrow j \rightsquigarrow t$ for some $s \in S, j \in J$. Then by definition of ℓ^J, we obtain $\ell^J \leq \ell^{\text{rev}}_\mathbf{r}(j)$. Furthermore, by Corollary 4.7, $j \rightsquigarrow t$ implies that $j \in \mathcal{I}^{\text{rev}}_\mathbf{r}(t)$ with

[12] To be more precise, we start with the partition $\mathcal{J}$ consisting of the singleton J^{mid}. We then iterate over any point j among $\text{A}(s)$, $\text{Z}(s)$, $s \in S$ and $\text{A}^{\text{rev}}(f)$, $\text{Z}^{\text{rev}}(f)$, $f \in F$ and replace the interval $J = [j_a, j_b]_{J^{\text{mid}}} \in \mathcal{J}$ containing j by the three intervals $[j_a, j)_{J^{\text{mid}}}$, $\{j\}$, $(j, j_b]_{J^{\text{mid}}}$, where the first and the last interval may be empty.

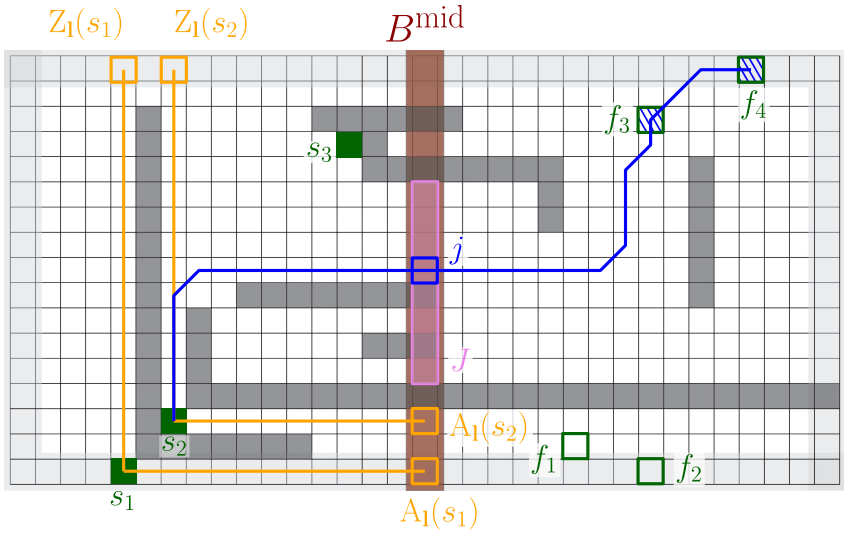


Fig. 9. Computation of R^J for an (S,F)-reach equivalent interval J. Intuitively, we first determine, among indices in J reachable from some $s \in S$, the index $j \in J$ with the best reachability towards F. We then determine all $f \in F$ reachable from j (indicated by hatched boxes).

$L^{\text{rev}}(t) \geq \ell_{\mathbf{r}}^{\text{rev}}(j) \geq \ell^J$. Note that $j \in \mathcal{I}_{\mathbf{r}}^{\text{rev}}(t)$ implies $J \subseteq \mathcal{I}_{\mathbf{r}}^{\text{rev}}(t)$ (as J is (S,F)-reach-equivalent), thus, we obtain that $J \subseteq \mathcal{I}_{\mathbf{r}}^{\text{rev}}(t)$ and $\ell^J \leq \ell_{\mathbf{r}}^{\text{rev}}(j)$, as desired. □

Thus, after computing ℓ^J, an orthogonal range reporting query can be used to report all $t \in F$ reachable from S via J. To compute ℓ^J, we observe that for any $j \in J$, we have

$$\exists s \in S : s \rightsquigarrow j \quad \overset{\text{Cor. 4.6}}{\iff} \quad \exists s \in S : j \in \mathcal{I}_1(s), \ell_1(j) \leq L(s) \quad \iff \quad \ell_1(j) \leq \max_{\substack{s \in S,\\ j \in \mathcal{I}_1(s)}} L(s) =: L^j.$$

Noting (by (S,F)-reach-equivalence of J) that $j \in \mathcal{I}_1(s)$ if and only if $J \subseteq \mathcal{I}_1(s)$, we have that L^j is independent of $j \in J$, and, in particular, equal to

$$L^J := \max_{\substack{s \in S,\\ J \subseteq \mathcal{I}_1(s)}} L(s), \tag{4}$$

which can be computed by a single orthogonal range minimization query. Equipped with this value, we may determine ℓ^J as

$$\ell^J = \min_{\substack{j \in J,\\ \ell_1(j) \leq L^J}} \ell_{\mathbf{r}}^{\text{rev}}(j). \tag{5}$$

Note that given ℓ^J, we may determine R^J by a single orthogonal range reporting query; by Equation (3).

We obtain the algorithm specified in Algorithm 5, whose correctness we summarize as follows: in the ith loop, we consider the ith (S,F)-reach-equivalent interval $J_i =: [a_i, b_i]_{J^{\text{mid}}}$ in the above partition of J^{mid}. Observe that we determine L^{J_i} according to its definition in Equation (4), and ℓ^{J_i} according to Equation (5). Finally, we include in the set R_i all elements of R^{J_i} that have not yet been reported in a previous iteration, exploiting (3).

Let us analyze the running time. Recall that $\ell = O(|S \cup F|) = O(|\mathcal{T}_B|)$. Thus, we can compute $J_1, \ldots, J_\ell$ in time $O(\ell \log \ell) = O(|\mathcal{T}_B| \log n)$. Furthermore, as discussed in Section 2.3, we can build

ALGORITHM 5: Computing SingleStepReach(B, S, F) for $B \in \mathcal{B}$, $S \subseteq B_{\mathrm{l}} \setminus B^{\mathrm{mid}}$, $F \subseteq B_{\mathrm{r}} \setminus B^{\mathrm{mid}}$.

function SingleStepReach(B, S, F)
  Compute a partition of J^{mid} into (S, F)-reach-equivalent intervals $J_1, \ldots, J_\ell$
  Build $\mathcal{OR}_S$ storing $L(s)$ under the key $(\mathrm{A}_{\mathrm{l}}(s), \mathrm{Z}_{\mathrm{l}}(s))$ for all $s \in S$ (for maximization queries)
  Build $\mathcal{OR}_F$ storing $\mathrm{ind}(f)$ under the key $(\mathrm{A}^{\mathrm{rev}}_{\mathrm{r}}(f), \mathrm{Z}^{\mathrm{rev}}_{\mathrm{r}}(f), L^{\mathrm{rev}}(f))$ for all $f \in F$ (for decremental range reporting queries)
  Recall: (precomputed) $\mathcal{OR}_B$ stores $\ell^{\mathrm{rev}}_2(q)$ under the key $(\mathrm{ind}(q), \ell_1(q))$ for all $q \in B^{\mathrm{mid}}_{\mathrm{free}}$ (for minimization queries)
  for $i = 1, \ldots, \ell$ **do** ▷ consider $J = [a_i, b_i]_{J^{\mathrm{mid}}}$
    $L^J \leftarrow \mathcal{OR}_S.\max((-\infty, a_i] \times [b_i, \infty))$
    $\ell^J \leftarrow \mathcal{OR}_B.\min([a_i, b_i] \times (-\infty, L^J])$
    $R_i \leftarrow \mathcal{OR}_F.\mathrm{report}((-\infty, a_i] \times [b_i, \infty) \times [\ell^J, \infty))$
    $\mathcal{OR}_F.\mathrm{delete}(R_i)$
  return $\bigcup_{i=1}^{\ell} R_i$

$\mathcal{OR}_S$ in time $O(|S| \log |S|) = O(|\mathcal{T}_B| \log n)$ to support maximization queries in time $O(\log |S|) = O(\log n)$. Also, we can build $\mathcal{OR}_F$ in time $O(|F| \log^2 |F|) = O(|\mathcal{T}_B| \log^2 |\mathcal{T}_B|)$ to support deletions in time $O(\log^2 |F|) = O(\log^2 |\mathcal{T}_B|)$ and queries in time $O(\log^2 |F| + k) = O(\log^2 |\mathcal{T}_B| + k)$ where k denotes the number of reported elements. Observe that $\mathcal{OR}_B$ is already precomputed, as it belongs to the information stored at block B (see Definition 4.9).

We perform $\ell = O(|\mathcal{T}_B|)$ iterations of the following form: First, we make a query to $\mathcal{OR}_S$ running in time $O(\log n)$, followed by a query to $\mathcal{OR}_B$ running in time $O(\log |\partial B|) = O(\log n)$. Then, we obtain a set R_i by a reporting query to $\mathcal{OR}_F$ running in time $O(\log^2 |\mathcal{T}_B| + |R_i|)$. Afterwards, we delete all reported elements, which takes time $O(|R_i| \log^2 |\mathcal{T}_B|)$. Thus, the total running time is bounded by $O(|\mathcal{T}_B| \log n + \sum_{i=1}^{\ell} |R_i| \log^2 |\mathcal{T}_B|)$. Observe that we report each element in $\mathcal{T}_B$ at most once, which results in $\sum_{i=1}^{\ell} |R_i| \leq |\mathcal{T}_B|$. Hence, the total running time is bounded by $O(|\mathcal{T}_B|(\log n + \log^2 |\mathcal{T}_B|)) = O(|\mathcal{T}_B| \log^2 n)$, as desired. □

5 CONDITIONAL LOWER BOUND

In this section we prove a lower bound of $n^{4-o(1)}$ for the discrete Fréchet distance under translation for two curves of length $n \in \mathbb{R}^2$ conditional on the Strong Exponential Time Hypothesis, or more precisely, the 4-OV Hypothesis. To this end, we reduce 4-OV to the discrete Fréchet distance under translation.

Let us first have a closer look at 4-OV. Given four sets of N vectors $V_1, \ldots, V_4 \subseteq \{0, 1\}^D$, the 4-OV problem can be expressed as

$$\exists v_1 \in V_1, \ldots, v_4 \in V_4 \; \forall j \in [D] \; \exists i \in \{1, \ldots, 4\} : v_i[j] = 0. \tag{6}$$

Recall from the introduction that we encode choosing the vectors $v_1, \ldots, v_4$ by the canonical translation $\tau = (\tau_1, \tau_2) = (h_1 \cdot \epsilon, h_2 \cdot \epsilon)$ with $h_1, h_2 \in \{0, \ldots, N^2 - 1\}$ for some constant $\epsilon > 0$ that is sufficiently small. To be concrete, let

$$\epsilon := 0.001/N^4$$

for the remaining section. Choosing $v_1 \in V_1$ and $v_2 \in V_2$, we define

$$h_1 := h(v_1, v_2) := \mathrm{ind}(v_1) + \mathrm{ind}(v_2) \cdot N,$$

where $\mathrm{ind}(v_i)$ is the index of vector v_i in the set V_i; similarly for $v_3 \in V_3, v_4 \in V_4$ we define $h_2 := h(v_3, v_4)$. To perform the reduction, we want to construct two curves π and σ whose Fréchet

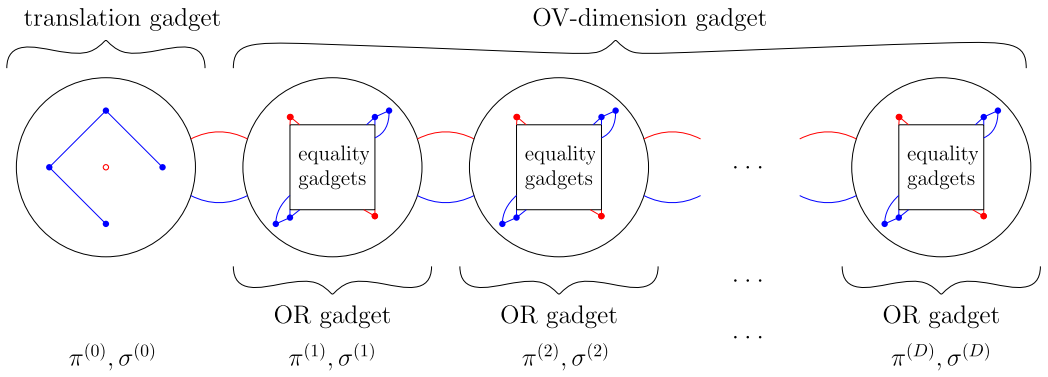


Fig. 10. Overview of how the different gadgets are used in the curves that result from the reduction. We use one translation gadget, one OV-dimension gadget, D OR gadgets, and $O(ND)$ equality gadgets.

distance decision for some δ is equivalent to the following expression, which is equivalent to Equation (6):

$$\exists \tau \in \mathbb{R}^2\ \forall j \in [D]\ \exists i \in \{1,2\}, v \in V_{2i-1}, v' \in V_{2i} : (v[j] = 0 \vee v'[j] = 0) \wedge (h(v,v') \cdot \epsilon = \tau_i). \quad (7)$$

The expressions (6) and (7) are equivalent as the three quantifiers encode the same choices and we evaluate if there exists a zero in one of the chosen vectors. In Equation (7) we additionally need to make sure that the translation chosen by the outermost quantifier indeed is consistent with the vectors that are chosen by the innermost quantifier, which is done by requiring $h(v,v') \cdot \epsilon = \tau_i$.

We can further transform this expression to make it easier to create gadgets for the reduction:

$$\exists \tau \in [0,(N^2-1)\cdot\epsilon] \times [0,(N^2-1)\cdot\epsilon] : \bigwedge_{j\in[D]} \bigvee_{\substack{i\in\{1,2\} \\ v\in V_{2i-1}, v'\in V_{2i}: \\ v[j]=0 \text{ or } v'[j]=0}} [h(v,v') \cdot \epsilon = \tau_i].$$

According to this formula, we will construct gadgets. However, we cannot exactly ensure the equality $h(v,v') \cdot \epsilon = \tau_i$. Therefore, we resort to an approximate equality that still fulfills the intended usage of mapping translations to vector choices. The approximate values just snap to the closest canonical values. The gadgets we construct are the following:

- *Translation gadget:* It ensures that $\tau \in [-\frac{1}{4}\cdot\epsilon, (N^2-\frac{3}{4})\cdot\epsilon] \times [-\frac{1}{4}\cdot\epsilon, (N^2-\frac{3}{4})\cdot\epsilon]$, i.e., we are always close to the points in the ϵ-grid of translations that choose our vectors $v_1, \ldots, v_4$.
- *OV-dimension gadget:* AND over all $j \in [D]$.
- *OR gadget:* The big OR in the formula.
- *Equality gadget:* This gadget is only traversable if the two vectors it was created for correspond to τ, i.e., it ensures that $h(v,v') \cdot \epsilon \approx \tau_i$.

We use the above-mentioned gadgets as follows: The constructed curves π and σ start with the translation gadget consisting of the curves $\pi^{(0)}, \sigma^{(0)}$. They are followed by D different parts that form the OV-dimension gadget. Each of the D parts is an OR gadget and we call the respective curves $\pi^{(j)}$ and $\sigma^{(j)}$ for $j \in [D]$. Each of the OR gadgets $(\pi^{(j)}, \sigma^{(j)})$ contains several equality gadgets. We will use different variations of the equality gadget (one for each set of vectors $V_1, \ldots, V_4$) but they are all of very similar structure. We need four different types of equality gadgets, because for a certain $v_i \in V_i$ a part of the gadget is only inserted if $v_i[d] = 0$. Thus, if we traverse an equality gadget later, then we know that it corresponds to one zero entry and also to the current translation. See Figure 10 for an overview of the whole construction.

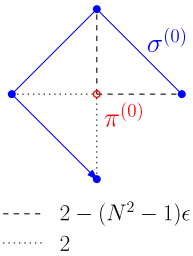


Fig. 11. Translation gadget.

Without loss of generality, assume that for all dimensions $j \in [D]$ at least one vector in $V_1 \cup \cdots \cup V_4$ contains a 0 in dimension j. Now, we give the detailed construction of the gadgets and the proofs of correctness. The instance of the discrete Fréchet distance under translation that we construct in the reduction uses a threshold distance of $\delta = 2 + \frac{1}{4}\epsilon$, i.e., we want to know for the constructed curves π and σ if their discrete Fréchet distance under translation is not more than δ.

Translation Gadget. First, we have to restrict the possible translations. To this end, we build a gadget to ensure $\tau \in \mathcal{T}$ for

$$\mathcal{T} := \left[-\frac{1}{4}\cdot\epsilon, \left(N^2 - \frac{3}{4}\right)\cdot\epsilon\right] \times \left[-\frac{1}{4}\cdot\epsilon, \left(N^2 - \frac{3}{4}\right)\cdot\epsilon\right].$$

This is realized by a gadget where curve $\pi^{(0)}$ consists of only one vertex and curve $\sigma^{(0)}$ consists of four vertices:

$$\pi^{(0)} := \langle(0,0)\rangle,$$
$$\sigma^{(0)} := \langle(2-(N^2-1)\epsilon, 0), (0, 2-(N^2-1)\epsilon), (-2,0), (0,-2)\rangle.$$

This gadget is sketched in Figure 11.

Lemma 5.1. *Given two curves $\pi = \pi^{(0)} \circ \pi'$ and $\sigma = \sigma^{(0)} \circ \sigma'$ (with $\pi^{(0)}, \sigma^{(0)}$ as defined above), such that each $p \in \pi^{(0)}$ has distance greater than 10 to each $p' \in \pi'$, the following holds:*

(i) if $\tau \in [0, (N^2-1)\epsilon] \times [0, (N^2-1)\epsilon]$, then $\delta_F(\pi^{(0)}, \sigma^{(0)} + \tau) \leq \delta$
(ii) if $\delta_F(\pi, \sigma + \tau) \leq \delta$, then $\tau \in [-\frac{1}{4}\cdot\epsilon, (N^2 - \frac{3}{4})\cdot\epsilon] \times [-\frac{1}{4}\cdot\epsilon, (N^2 - \frac{3}{4})\cdot\epsilon$

Proof. We start with showing (i), so assume $\tau \in [0, (N^2-1)\epsilon] \times [0, (N^2-1)\epsilon]$. Note that the maximal distance $\max_{q\in\sigma^{(0)}} \max_\tau \|\pi^{(0)} - (q+\tau)\|$ is an upper bound on $\delta_F(\pi^{(0)}, \sigma^{(0)} + \tau)$. By a simple calculation, we obtain the desired result:

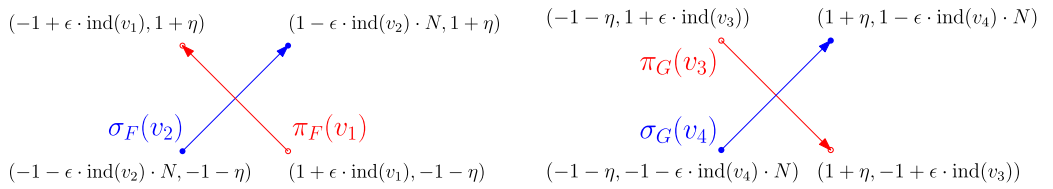
$$\max_{q\in\sigma^{(0)}} \max_\tau \left\|\pi^{(0)} - (q+\tau)\right\| < \sqrt{2^2 + \epsilon^2 N^4} < \sqrt{2^2 + \epsilon + \frac{1}{16}\epsilon^2} = 2 + \frac{1}{4}\epsilon,$$

where we used $\epsilon \leq N^{-4}$.

Now, we prove (ii). Note that the start points of π and $\sigma + \tau$ have to be in distance $\leq \delta$, thus $\tau \in [-5, 5]^2$ (using a very rough estimate). As all points of π' are further than 10 from any point in $\pi^{(0)}$ and thus all points on the postfix π' are further than δ from $\sigma^{(0)} + \tau$, we have to stay in $\pi^{(0)}$ while traversing $\sigma^{(0)}$. Thus, the following inequalities hold for $\tau_i > (N^2 - \frac{3}{4})\epsilon$ or $\tau_i < -\frac{1}{4}\epsilon$ and $i \in \{1, 2\}$ (where $\|v\|_\infty$ denotes the infinity norm of v):

$$\delta_F(\pi, \sigma+\tau) \geq \delta_F(\pi^{(0)}, \sigma^{(0)} + \tau) \geq \max_{i\in[4]} \left\{\left\|\pi_1^{(0)} - (\sigma_i^{(0)} + \tau)\right\|_\infty\right\} > \delta,$$

which is the contrapositive of (ii). □



(a) Equality Gadget for V_1 and V_2 **(b)** Equality Gadget for V_3 and V_4

Fig. 12. The equality gadgets for F and G. The equality gadgets F' and G' are simply shifted.

OV-dimension Gadget. For every 4-OV dimension $j \in [D]$, we construct separate gadgets $\pi^{(1)}, \ldots, \pi^{(D)}$ for π and $\sigma^{(1)}, \ldots, \sigma^{(D)}$ for σ. We want to connect these gadgets in a way that the two curves are in distance not more than δ if and only if all gadgets have distance not more than δ for a given translation τ. This is done by simply placing the gadgets in distance greater than $\delta + N^2 \cdot \epsilon$ from each other and concatenating them.

Lemma 5.2. *Given a translation $\tau \in \mathcal{T}$ and curves $\pi = \pi^{(1)}, \ldots, \pi^{(D)}$ and $\sigma = \sigma^{(1)}, \ldots, \sigma^{(D)}$ where for all $j \in [D]$ all points of $\pi^{(j)}$ are further than $\delta + 2N^2 \cdot \epsilon$ from each point of $\sigma^{(j')}$ with $j \neq j'$, then $\delta(\pi, \sigma + \tau) \leq \delta$ if and only if $\delta_F(\pi^{(j)}, \sigma^{(j)} + \tau) \leq \delta$ for all $j \in [D]$.*

Proof. First, note that whatever τ we choose in the given range, $\sigma^{(j)} + \tau$ is still in distance greater than δ from every $\pi^{(j')}$ with $j' \neq j$.

Now, assume that for all $j \in [D]$ the curves $\pi^{(j)}, \sigma^{(j)} + \tau$ have distance at most δ. Then, we can traverse the gadgets in order and do simultaneous jumps between them. Thus, also the distance of the whole curves π and $\sigma + \tau$ is at most δ. For the other direction, assume that for at least one $j \in [D]$ the distance between $\pi^{(j)}$ and $\sigma^{(j)} + \tau$ is greater than δ. On the one hand, if we do not traverse simultaneously (i.e., at one point the traversal is in $\pi^{(j)}$ and $\sigma^{(j')}$ for $j \neq j'$), then due to large distances of $\pi^{(j)}, \sigma^{(j')} + \tau$ for $j \neq j'$, we have distance greater than δ for this traversal. On the other hand, if we traverse $\pi^{(j)}$ and $\sigma^{(j)}$ together for all j, then we also have distance greater than δ due to the gadget with distance greater than δ. □

For the remaining gadgets we define for convenience:

$$\eta := 3 \cdot N^2 \epsilon.$$

Equality Gadget. An equality gadget $F(v_1, v_2)$ for the vectors $v_1 \in V_1, v_2 \in V_2$ is a pair of two line segments, $\pi_F(v_1)$ and $\sigma_F(v_2)$, (see Figure 12(a)):

$$\pi_F(v_1) := \langle (1 + \epsilon \cdot \mathrm{ind}(v_1), -1 - \eta), (-1 + \epsilon \cdot \mathrm{ind}(v_1), 1 + \eta) \rangle,$$
$$\sigma_F(v_2) := \langle (-1 - \epsilon \cdot \mathrm{ind}(v_2) \cdot N, -1 - \eta), (1 - \epsilon \cdot \mathrm{ind}(v_2) \cdot N, 1 + \eta) \rangle.$$

Note that this gives us N^2 different gadgets consisting of $2N$ different line segments. We later use the line segments $\pi_F(v_1)$ in π and the line segments $\sigma_F(v_2)$ in σ where they can be combined to form an equality gadget.

Lemma 5.3. *Given curves $\pi_F(v_1), \sigma_F(v_2)$ for some $v_1 \in V_1$ and $v_2 \in V_2$, and given a translation $\tau \in \mathcal{T}$, the following properties hold:*

(i) if $\tau_1 = \epsilon \cdot (\mathrm{ind}(v_1) + \mathrm{ind}(v_2) \cdot N)$, then $\delta_F(\pi_F(v_1), \sigma_F(v_2) + \tau) \leq \delta$,
(ii) if $\delta_F(\pi_F(v_1), \sigma_F(v_2) + \tau) \leq \delta$, then $|\epsilon \cdot (\mathrm{ind}(v_1) + \mathrm{ind}(v_2) \cdot N) - \tau_1| \leq \frac{1}{3}\epsilon$.

Proof. To prove (i), it suffices to give a valid traversal. We traverse $\pi_F(v_1) = (p_1, p_2)$ and $\sigma_F(v_2) = (q_1, q_2)$ simultaneously. Thus, we just want an upper bound on the distance between the

(translated) first points $p_1, q_1 + \tau$ and the distance between the (translated) second points $p_2, q_2 + \tau$ to get an upper bound on $\delta_F(\pi_F(v_1), \sigma_F(v_2) + \tau)$. These distances are

$$\|p_1 - (q_1 + \tau)\|^2 = (2 + \epsilon \cdot \text{ind}(v_1) + \epsilon \cdot \text{ind}(v_2) \cdot N - \tau_1)^2 + \tau_2^2 \quad = 4 + \tau_2^2 \leq 4 + \epsilon + \frac{1}{16}\epsilon^2 = \delta^2,$$
$$\|p_2 - (q_2 + \tau)\|^2 = (-2 + \epsilon \cdot \text{ind}(v_1) + \epsilon \cdot \text{ind}(v_2) \cdot N - \tau_1)^2 + \tau_2^2 = 4 + \tau_2^2 \leq \delta^2,$$

where we used $|\tau_2| \leq N^2\epsilon$ and thus $\tau_2^2 \leq N^4\epsilon^2 \leq \epsilon$, since $\epsilon \leq N^{-4}$. Both distances are at most δ and thus the discrete Fréchet distance is at most δ as well.

For proving (ii), first note that the first (respectively, second) point of $\pi_F(v_1)$ is far from the second (respectively, first) point of $\sigma_F(v_2)$, due to $\eta \geq N^2\epsilon$. Thus, we have to traverse the gadget simultaneously. Let $\Delta \coloneqq \epsilon \cdot \text{ind}(v_1) + \epsilon \cdot \text{ind}(v_2) \cdot N - \tau_1$, it remains to show that $\Delta \leq \frac{1}{3}\epsilon$. For p_1, q_1, we then get

$$\begin{aligned}
\|p_1 - (q_1 + \tau)\|^2 = \quad & (2 + \epsilon \cdot \text{ind}(v_1) + \epsilon \cdot \text{ind}(v_2) \cdot N - \tau_1)^2 + \tau_2^2 && \leq && (2 + \tfrac{1}{4}\epsilon)^2 \\
\Leftrightarrow \quad & (2 + \Delta)^2 + \tau_2^2 && \leq && 4 + \epsilon + \tfrac{1}{16}\epsilon^2 \\
\Leftrightarrow \quad & 4 + 4\Delta + \Delta^2 + \tau_2^2 && \leq && 4 + \epsilon + \tfrac{1}{16}\epsilon^2 \\
\Rightarrow \quad & 4\Delta && \leq && \epsilon + \tfrac{1}{16}\epsilon^2 \\
\Rightarrow \quad & \Delta && \leq && \tfrac{1}{4}\epsilon + \tfrac{1}{64}\epsilon^2 \leq \tfrac{1}{3}\epsilon.
\end{aligned}$$

The last inequality follows from plugging in $\epsilon = 0.001/N^4$ and using the fact that $N \geq 1$. With a similar calculation for p_2, q_2 we obtain that $\Delta \geq -\frac{1}{3}\epsilon$, and thus $|\Delta| \leq \frac{1}{3}\epsilon$. □

Now, we introduce three gadgets that have the same properties as the equality gadget but are slightly different. The aim is to have four types of gadgets that are pairwise further than a discrete Fréchet distance of δ apart such that we can use them together in one big OR expression.

Shifted Equality Gadget. As described in the introduction of this section, we want to use the curves $\pi_F(v_1), \sigma_F(v_2)$ in case $v_1[j] = 0$ and we need an additional gadget for $v_2[j] = 0$. However, those two gadgets should not be too close such that the curves cannot be matched but also not too far such that the OR gadget (which we introduce later) still works. Thus, we introduce another gadget $F'(v_1, v_2)$ that consists of a pair of curves $\pi_{F'}(v_1), \sigma_{F'}(v_2)$ that are just shifted versions of $\pi_F(v_1), \sigma_F(v_2)$; shifted by $N^2\epsilon$ in the first dimension. More formally,

$$\pi_{F'}(v_1) \coloneqq \pi_F(v_1) + (N^2\epsilon, 0),$$
$$\sigma_{F'}(v_2) \coloneqq \sigma_F(v_2) + (N^2\epsilon, 0).$$

Before proving the desired properties, we introduce the remaining two variants of the equality gadget.

Equality Gadget for V_3 and V_4. The above introduced equality gadgets only work for vectors in V_1 and V_2 but we also need a gadget for vectors in V_3 and V_4. Therefore, we introduce the gadget $G(v_3, v_4)$, which is a mirrored equality gadget consisting of the curves $\pi_G(v_3)$ and $\sigma_G(v_4)$ (see Figure 12(b)):

$$\pi_G(v_3) \coloneqq \langle(-1 - \eta, 1 + \epsilon \cdot \text{ind}(v_3)), (1 + \eta, -1 + \epsilon \cdot \text{ind}(v_3))\rangle,$$
$$\sigma_G(v_4) \coloneqq \langle(-1 - \eta, -1 - \epsilon \cdot \text{ind}(v_4) \cdot N), (1 + \eta, 1 - \epsilon \cdot \text{ind}(v_4) \cdot N)\rangle.$$

Shifted Equality Gadget for V_3 and V_4. We define $G'(v_3, v_4)$ similarly to $F'(v_1, v_2)$, i.e., we shift the curves of G by $N^2\epsilon$, but in contrast to F', we shift it in the second dimension. More formally:

$$\pi_{G'}(v_3) \coloneqq \pi_G(v_3) + (0, N^2\epsilon),$$
$$\sigma_{G'}(v_4) \coloneqq \sigma_G(v_4) + (0, N^2\epsilon).$$

Due to the similar structure of the curve pairs of $F(v_1, v_2)$ and $F'(v_1, v_2), G(v_3, v_4), G'(v_3, v_4)$, analogous statements to Lemma 5.3 also hold for the curve pairs from $F'(v_1, v_2), G(v_3, v_4)$, and $G'(v_3, v_4)$. Specifically, we have:

Lemma 5.4. *Given curves $\pi_{F'}(v_1), \sigma_{F'}(v_2)$ for some $v_1 \in V_1$ and $v_2 \in V_2$, and given a translation $\tau \in \mathcal{T}$, the following properties hold:*

(i) if $\tau_1 = \epsilon \cdot (\mathrm{ind}(v_1) + \mathrm{ind}(v_2) \cdot N)$, then $\delta_F(\pi_{F'}(v_1), \sigma_{F'}(v_2) + \tau) \leq \delta$,
(ii) if $\delta_F(\pi_{F'}(v_1), \sigma_{F'}(v_2) + \tau) \leq \delta$, then $|\epsilon \cdot (\mathrm{ind}(v_1) + \mathrm{ind}(v_2) \cdot N) - \tau_1| \leq \frac{1}{3}\epsilon$.

Lemma 5.5. *Given curves $\pi_G(v_3), \sigma_G(v_4)$ for some $v_3 \in V_3$ and $v_4 \in V_4$, and given a translation $\tau \in \mathcal{T}$, the following properties hold:*

(i) if $\tau_2 = \epsilon \cdot (\mathrm{ind}(v_3) + \mathrm{ind}(v_4) \cdot N)$, then $\delta_F(\pi_G(v_3), \sigma_G(v_4) + \tau) \leq \delta$,
(ii) if $\delta_F(\pi_G(v_3), \sigma_G(v_4) + \tau) \leq \delta$, then $|\epsilon \cdot (\mathrm{ind}(v_3) + \mathrm{ind}(v_4) \cdot N) - \tau_2| \leq \frac{1}{3}\epsilon$.

Lemma 5.6. *Given curves $\pi_{G'}(v_3), \sigma_{G'}(v_4)$ for some $v_3 \in V_3$ and $v_4 \in V_4$, and given a translation $\tau \in \mathcal{T}$, the following properties hold:*

(i) if $\tau_2 = \epsilon \cdot (\mathrm{ind}(v_3) + \mathrm{ind}(v_4) \cdot N)$, then $\delta_F(\pi_{G'}(v_3), \sigma_{G'}(v_4) + \tau) \leq \delta$,
(ii) if $\delta_F(\pi_{G'}(v_3), \sigma_{G'}(v_4) + \tau) \leq \delta$, then $|\epsilon \cdot (\mathrm{ind}(v_3) + \mathrm{ind}(v_4) \cdot N) - \tau_2| \leq \frac{1}{3}\epsilon$.

We now show that all subcurves of different equality gadgets are pairwise further apart than δ. Here, we say that the curves $\pi_F(v_1)$ and $\sigma_F(v_2)$ have *type F*. Similarly, the other curves constructed above have type F', G, or G'.

Lemma 5.7. *For any vectors $v_1 \in V_1, \ldots, v_4 \in V_4$ and any translation $\tau \in \mathcal{T}$, for any curves $\pi \in \{\pi_F(v_1), \pi_{F'}(v_1), \pi_G(v_3), \pi_{G'}(v_3)\}$ and $\sigma \in \{\sigma_F(v_2), \sigma_{F'}(v_2), \sigma_G(v_4), \sigma_{G'}(v_4)\}$ of different type, we have $\delta_F(\pi, \sigma + \tau) > \delta$.*

Proof. We first consider $\pi_{F'}(v_1)$ and $\sigma_F(v_2)$. Consider the first point of $\sigma_F(v_2)$ which we call q. This point is further than $2 + N^2\epsilon$ from both points of $\pi_{F'}(v_1)$. When translating σ with $\tau \in \mathcal{T}$, the distance is still greater than $2 + \frac{3}{4}\epsilon > \delta$. Thus, $\sigma_F(v_2)$ and $\pi_{F'}(v_1)$ are in discrete Fréchet distance greater than δ for any valid τ.

Similarly, consider $\pi_F(v_1)$ and $\sigma_{F'}(v_2)$, and let p be the second point of $\pi_F(v_1)$. The point p has distance greater than $2 + \epsilon$ from $\sigma_{F'}(v_2)$. With translation $\tau \in \mathcal{T}$ this distance is still greater than $2 + \frac{3}{4}\epsilon > \delta$ and thus $\pi_F(v_1)$ and $\sigma_{F'}(v_2)$ are in discrete Fréchet distance greater than δ for any valid τ. The proof for types G and G' is symmetric.

Now, we prove the lemma for types F and G. First note that every point of $\pi_F(v_1)$ is in distance $1 + \eta$ of the first coordinate axis and every point of $\sigma_G(v_4)$ is in distance $1 + \eta$ of the second coordinate axis. Additionally, no point of $\pi_F(v_1)$ is closer than $1 - 2N^2\epsilon$ to the second coordinate axis while no point of $\sigma_G(v_4)$ is closer than $1 - 2N^2\epsilon$ to the first coordinate axis. This means that every point of $\pi_F(v_1)$ is in distance at least $2 + \eta - 2N^2\epsilon = 2 + N^2\epsilon$ of any point of $\sigma_G(v_4)$. Even with translation this distance is at least $2 + \frac{3}{4}\epsilon > \delta$. Thus, also the discrete Fréchet distance is greater than δ. The proofs for the remaining cases are symmetric. □

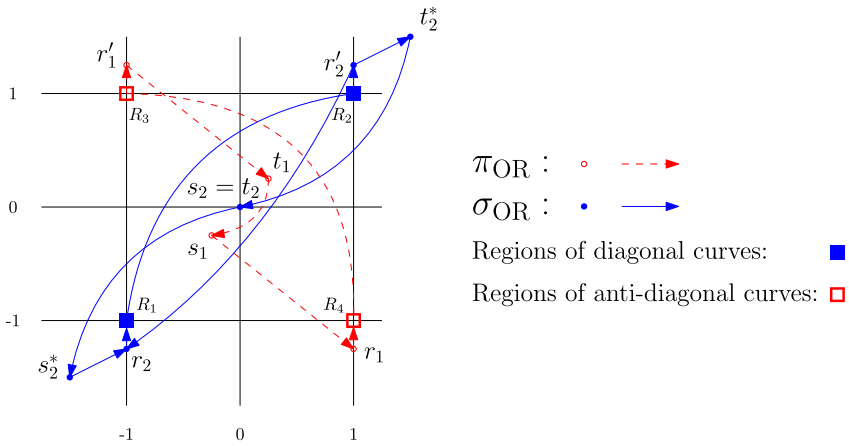


Fig. 13. The OR gadget for general diagonal and anti-diagonal curves.

We moreover observe that our equality gadgets lie in very restricted regions. Specifically, call a curve *diagonal* if all of its vertices are in $R_1 \cup R_2$ with

$$R_1 := [-1-2\eta, -1+2\eta]^2,\ R_2 := [1-2\eta, 1+2\eta]^2,$$

and we call it *anti-diagonal* if all of its vertices are contained in $R_3 \cup R_4$ with

$$R_3 := [-1-2\eta, -1+2\eta] \times [1-2\eta, 1+2\eta],\ R_4 := [1-2\eta, 1+2\eta] \times [-1-2\eta, -1+2\eta].$$

See Figure 13. Also, note that the order in which the curves visit the regions is not specified in the definition of (anti-)diagonal.

Observation 5.8. *The constructed curves $\pi_F(v_1), \pi_{F'}(v_1), \pi_G(v_3), \pi_{G'}(v_3)$ are anti-diagonal, and the curves $\sigma_F(v_2), \sigma_{F'}(v_2), \sigma_G(v_4), \sigma_{G'}(v_4)$ are diagonal.*

Proof. We observe that each coordinate of a vertex of any of these curves differs from 1 or -1 by at most $\varepsilon N^2 + \max\{\eta, \varepsilon N^2\}$, by bounding $0 \le \mathrm{ind}(v_i) \le N$. Recalling $\eta = 3 \cdot N^2\epsilon$, we have $\varepsilon N^2 \le \eta$. Therefore, any coordinate differs from 1 or -1 by at most 2η, that is, each coordinate lies in $R_1 \cup R_2 \cup R_3 \cup R_4$. The general shape of being (anti-)diagonal can be inferred from Figure 12. □

We are now ready to describe the last gadget. For proving its correctness, we will essentially only use the diagonal and anti-diagonal property of the curves.

OR Gadget. We construct an OR gadget over diagonal and anti-diagonal curves that we will later apply to equality gadgets. Before introducing the gadget itself, we define various auxiliary points whose meaning will become clear later. Here we keep notation close to Reference [14], although the details of our construction are quite different.

$$s_1 := \left(-\tfrac{1}{4}, -\tfrac{1}{4}\right),\ t_1 := \left(\tfrac{1}{4}, \tfrac{1}{4}\right),\ r_1 := \left(\tfrac{99}{100}, -\tfrac{5}{4}\right),\ r_1' := \left(-\tfrac{99}{100}, \tfrac{5}{4}\right),$$
$$s_2 := (0,0),\ s_2^* := \left(-\tfrac{3}{2}, -\tfrac{3}{2}\right),\ t_2^* := \left(\tfrac{3}{2}, \tfrac{3}{2}\right),\ t_2 := (0,0),\ r_2 := \left(-\tfrac{99}{100}, -\tfrac{5}{4}\right),\ r_2' := \left(\tfrac{99}{100}, \tfrac{5}{4}\right).$$

Now, given diagonal curves $\hat{\sigma}^1, \ldots, \hat{\sigma}^\ell$ and anti-diagonal curves $\hat{\pi}^1, \ldots, \hat{\pi}^k$, we define the two curves of the OR gadget as

$$\pi_{\mathrm{OR}} := \bigcirc_{i \in [k]} s_1 \circ r_1 \circ \hat{\pi}^i \circ r_1' \circ t_1,$$
$$\sigma_{\mathrm{OR}} := s_2 \circ s_2^* \circ \left(\bigcirc_{j \in [\ell]} r_2 \circ \hat{\sigma}^j \circ r_2' \right) \circ t_2^* \circ t_2.$$

See Figure 13 for a visualization. Now let us prove correctness of the gadget.

Lemma 5.9. *Given an OR gadget over diagonal curves $\hat{\sigma}^1, \ldots, \hat{\sigma}^\ell$ and anti-diagonal curves $\hat{\pi}^1, \ldots, \hat{\pi}^k$, for any translation $\tau \in \mathcal{T}$, we have $\delta_F(\pi_{\text{OR}}, \sigma_{\text{OR}} + \tau) \leq \delta$ if and only if $\delta_F(\hat{\pi}^i, \hat{\sigma}^j + \tau) \leq \delta$ for some i, j.*

Proof. We first observe that for none of the auxiliary points $p \in \pi_{\text{OR}}$ and $q \in \sigma_{\text{OR}}$, we have that $\|p - q\| \in [1.99, 2.01]$. This can be verified by calculating all distances, but we omit this due to readability of the proof. Also observe that $\mathcal{T} \subset [-0.001, 0.001]$ and $\delta \in [2, 2.001]$. It follows from the above observations that the translation τ does not change whether auxiliary points are closer than δ or not. Thus, we can ignore the translation for distances between auxiliary points in this proof. For reference, we state which auxiliary points are closer than δ for all $\tau \in \mathcal{T}$. For each auxiliary point in π_{OR}, we list its close auxiliary points in σ_{OR}:

$$\begin{aligned} s_1 &: \quad s_2, s_2^*, t_2, r_2, r_2', \\ t_1 &: \quad s_2, t_2, r_2, r_2', \\ r_1 &: \quad s_2, t_2, r_2, \\ r_1' &: \quad s_2, t_2, r_2'. \end{aligned}$$

All other pairs are in distance greater than δ. Note that for the remainder of the proof, we do not have to consider the specific value for τ anymore.

We first show that if $\delta_F(\hat{\pi}^i, \hat{\sigma}^j + \tau) \leq \delta$ for some i, j, then $\delta_F(\pi_{\text{OR}}, \sigma_{\text{OR}} + \tau) \leq \delta$ by giving a valid traversal. We start in $s_1, s_2 + \tau$. Then, we traverse π_{OR} until the copy of s_1 that comes before the subcurve $\hat{\pi}^i$. While staying in s_1, we traverse $\sigma_{\text{OR}} + \tau$ until we reach the copy of $r_2 + \tau$ right before the subcurve $\hat{\sigma}^j + \tau$. We then do one step on π_{OR} to r_1. Now, we step to the first nodes of $\hat{\pi}^i$ and $\hat{\sigma}^j + \tau$ simultaneously, and then traverse these two subcurves in distance δ, which is possible due to $\delta_F(\hat{\pi}^i, \hat{\sigma}^j + \tau) \leq \delta$. We then step to the copies of r_1' and $r_2' + \tau$ simultaneously. We then step to t_1 on π_{OR}, while staying at $r_2' + \tau$ in $\sigma_{\text{OR}} + \tau$. Subsequently, while staying in t_1, we traverse $\sigma_{\text{OR}} + \tau$ until we reach its last point, namely, $t_2 + \tau$. Now, we can traverse the remainder of π_{OR}. One can check that this traversal stays within distance δ.

We now show that if $\delta_F(\pi_{\text{OR}}, \sigma_{\text{OR}} + \tau) \leq \delta$, then there exist i, j such that $\delta_F(\hat{\pi}^i, \hat{\sigma}^j + \tau) \leq \delta$. Pick any valid traversal for which $\delta_F(\pi_{\text{OR}}, \sigma_{\text{OR}} + \tau) \leq \delta$. We reconstruct in the following how it passed through π_{OR} and $\sigma_{\text{OR}} + \tau$. Consider the point when $s_2^* + \tau$ is reached. At that point, we have to be in some copy of s_1, as this is the only type of node of π_{OR} that is in distance at most δ from $s_2^* + \tau$. Let $\hat{\pi}^i$ be the subcurve right after this copy of s_1. When we step to the copy of r_1 right after this s_1, there are only three types of nodes from $\sigma_{\text{OR}} + \tau$ in distance δ: $s_2 + \tau, t_2 + \tau, r_2 + \tau$. Note that we already passed $s_2 + \tau$, and we cannot have reached $t_2 + \tau$ yet, as $t_2^* + \tau$ is neither in reach of s_1 nor r_1. Thus, we are in $r_2 + \tau$. Let the curve right after $r_2 + \tau$ be $\hat{\sigma}^j + \tau$. The only option now is to do a simultaneous step to the first nodes of $\hat{\pi}^i$ and $\hat{\sigma}^j + \tau$. Now, consider the point when either r_1' or $r_2' + \tau$ is first reached. All points of $\hat{\pi}^i$ are far from $r_2' + \tau$ and all points of $\hat{\sigma}^j + \tau$ are far from r_1' and thus we have to be in r_1' and $r_2' + \tau$ at the same time. This implies that we traversed $\hat{\pi}^i$ and $\hat{\sigma}^j + \tau$ from the start to the end nodes in distance δ and therefore $\delta_F(\hat{\pi}^i, \hat{\sigma}^j + \tau) \leq \delta$. □

Assembling $\pi^{(j)}$ and $\sigma^{(j)}$. Now, we can apply the OR gadget to the equality gadgets in the following way: For each of the D dimensions, we construct an OR gadget. The OR gadget for dimension $j \in [D]$ contains as anti-diagonal curves all $\pi_F(v_1)$ with $v_1[j] = 0$, all $\pi_{F'}(v_1)$, all $\pi_G(v_3)$ with $v_3[j] = 0$, and all $\pi_{G'}(v_3)$; and as diagonal curves it contains all $\sigma_F(v_2)$, all $\sigma_{F'}(v_2)$ with $v_2[j] = 0$, all $\sigma_G(v_4)$, and all $\sigma_{G'}(v_4)$ with $v_4[j] = 0$. Note that these curves fulfill the requirements stated in Observation 5.8 for usage in the OR gadget as (anti-)diagonal curves. We denote the resulting curves by $\pi^{(j)}$ and $\sigma^{(j)}$, and we write $H(j) = (\pi^{(j)}, \sigma^{(j)})$. This yields the following lemma:

Lemma 5.10. *Given a 4-OV instance $V_1, \dots, V_4$, and consider the corresponding OR gadget $H(j) = (\pi^{(j)}, \sigma^{(j)})$ for some $j \in [D]$. It holds that:*

(i) For any vectors $v_1 \in V_1, \dots, v_4 \in V_4$ with $v_1[j] \cdot v_2[j] \cdot v_3[j] \cdot v_4[j] = 0$ we have $\delta_F(\pi^{(j)}, \sigma^{(j)} + \tau) \leq \delta$ for $\tau = ((\text{ind}(v_1) + \text{ind}(v_2) \cdot N) \cdot \epsilon, (\text{ind}(v_3) + \text{ind}(v_4) \cdot N) \cdot \epsilon)$.

(ii) If $\delta_F(\pi^{(j)}, \sigma^{(j)} + \tau) \leq \delta$ for some $\tau \in \mathcal{T}$, then

- $\exists v_1 \in V_1, v_2 \in V_2 : v_1[j] \cdot v_2[j] = 0$ *and* $|\epsilon \cdot (\text{ind}(v_1) + \text{ind}(v_2) \cdot N) - \tau_1| \leq \frac{1}{3}\epsilon$
  or
- $\exists v_3 \in V_3, v_4 \in V_4 : v_3[j] \cdot v_4[j] = 0$ *and* $|\epsilon \cdot (\text{ind}(v_3) + \text{ind}(v_4) \cdot N) - \tau_2| \leq \frac{1}{3}\epsilon$.

Proof. For (i), from $v_1[j] \cdot v_2[j] \cdot v_3[j] \cdot v_4[j] = 0$ it follows that at least one gadget of $F(v_1, v_2)$, $F'(v_1, v_2)$, $G(v_3, v_4)$, $G'(v_3, v_4)$ is contained in $H(j)$. By Lemmas 5.3 to 5.6, we know that the discrete Fréchet distance of this gadget is small. By Lemma 5.9 it then follows that $\delta_F(\pi^{(j)}, \sigma^{(j)} + \tau) \leq \delta$.

For (ii), from $\delta_F(\pi^{(j)}, \sigma^{(j)} + \tau) \leq \delta$ it follows by Lemmas 5.9 and 5.7 that there exists a gadget Γ for which the discrete Fréchet distance is at most δ. From Lemmas 5.3 to 5.6 it then follows that

$$|\epsilon \cdot (\text{ind}(v_1) + \text{ind}(v_2) \cdot N) - \tau_1| \leq \frac{1}{3}\epsilon \quad \text{or} \quad |\epsilon \cdot (\text{ind}(v_3) + \text{ind}(v_4) \cdot N) - \tau_2| \leq \frac{1}{3}\epsilon$$

for some vectors $v_1 \in V_1, \dots, v_4 \in V_4$. As Γ is contained in the OR gadget, we additionally have that $v_1[j] \cdot v_2[j] = 0$ or $v_3[j] \cdot v_4[j] = 0$, respectively. □

Final Curves. The final curves π and σ are now defined as follows: We start with the translation gadget $\pi^{(0)}$ ($\sigma^{(0)}$). Then the curves $\pi^{(j)}$ ($\sigma^{(j)}$) follow for $j \in [D]$. Note that we have to translate these curves to fulfill the requirements of Lemmas 5.1 and 5.2, thus, we translate $\pi^{(j)}$ ($\sigma^{(j)}$) by $(100 \cdot j, 0)$. More explicitly, the final curves are

$$\pi \coloneqq \pi^{(0)} \bigcirc_{j \in [D]} \pi^{(j)} + (100 \cdot j, 0),$$
$$\sigma \coloneqq \sigma^{(0)} \bigcirc_{j \in [D]} \sigma^{(j)} + (100 \cdot j, 0).$$

We are now ready to prove Theorem 1.2. Recall its statement:

Also recall that it suffices to prove a lower bound under the 4-OV hypothesis. For clarity of structure, we split the proof into Lemma 5.11 and Lemma 5.12, which together imply Theorem 1.2.

Lemma 5.11. *Given a YES-instance of 4-OV, the curves π and σ constructed in our reduction have discrete Fréchet distance under translation at most δ, i.e., $\min_\tau \delta_F(\pi, \sigma + \tau) \leq \delta$.*

Proof. Let $v_1 \in V_1, \dots, v_4 \in V_4$ be orthogonal vectors and let $\tau = ((\text{ind}(v_1) + \text{ind}(v_2) \cdot N) \cdot \epsilon, (\text{ind}(v_3) + \text{ind}(v_4) \cdot N) \cdot \epsilon)$ be the translation corresponding to those vectors. From Lemma 5.1, we know that $\delta_F(\pi^{(0)}, \sigma^{(0)} + \tau) \leq \delta$, and thus there is a valid traversal to the endpoints of the translation gadget. Then, we simultaneously step to the start of $\pi^{(1)}$ and $\sigma^{(1)}$. From Lemma 5.10, we know that there also exist traversals of $\pi^{(1)}, \dots, \pi^{(D)}$ and $\sigma^{(1)} + \tau, \dots, \sigma^{(D)} + \tau$ of distance at most δ. It follows from Lemma 5.2 that we can also traverse those gadgets sequentially in distance δ and thus $\delta_F(\pi, \sigma + \tau) \leq \delta$. □

Lemma 5.12. *If the curves π and σ constructed in our reduction have discrete Fréchet distance under translation at most δ, then the given 4-OV instance is a YES-instance.*

Proof. Let τ be a translation such that $\delta_F(\pi, \sigma + \tau) \leq \delta$. We know from Lemma 5.1 that $\tau \in \mathcal{T}$. Furthermore, from Lemma 5.2, we know that for all $j \in [D]$ it holds that $\delta_F(\pi^{(j)}, \sigma^{(j)} + \tau) \leq \delta$. It follows from Lemma 5.10 that for every $j \in [D]$ there exist $v_1 \in V_1, v_2 \in V_2$ such that $v_1[j] \cdot v_2[j] = 0$ and $|\epsilon \cdot (\text{ind}(v_1) + \text{ind}(v_2) \cdot N) - \tau_1| \leq \frac{1}{3}\epsilon$ or there exist $v_3 \in V_3, v_4 \in V_4$ such that $v_3[j] \cdot v_4[j] = 0$ and $|\epsilon \cdot (\text{ind}(v_3) + \text{ind}(v_4) \cdot N) - \tau_2| \leq \frac{1}{3}\epsilon$. Therefore, every dimension $j \in [D]$ gives us constraints on either v_1, v_2 or v_3, v_4. Due to Lemma 5.10 these constraints have to be consistent. If in total this

gives us constraints for $v_1, \ldots, v_4$, then we are done. Otherwise, if this only gives us constraints for v_1, v_2, then we already found v_1, v_2, which are orthogonal, and thus we can pick arbitrary $v_3 \in V_3, v_4 \in V_4$ to obtain an orthogonal set of vectors. The case of only v_3, v_4 being constrained is symmetric. □

Proof of Theorem 1.2. The Strong Exponential Time Hypothesis implies the k-OV hypothesis. The reduction above from a 4-OV instance of size N over $\{0,1\}^D$ to an instance of the discrete Fréchet distance under translation in $\mathbb{R}^2$ results in two curves of length $O(D \cdot N)$. Lemmas 5.11 and 5.12 show correctness of this reduction. Hence, any $O(n^{4-\epsilon})$-time algorithm for the discrete Fréchet distance under translation would imply an algorithm for 4-OV in time $O((D \cdot N)^{4-\epsilon}) = O(\text{poly}(D) \cdot N^{4-\epsilon})$, refuting the k-OV hypothesis. □

6 CONCLUSION

In this work, we designed an improved algorithm for the discrete Fréchet distance under translation running in time $\tilde{O}(n^{14/3}) = \tilde{O}(n^{4.66\cdots})$. As a crucial subroutine, we developed an improved algorithm for offline dynamic grid reachability. Additionally, we presented a conditional lower bound of $n^{4-o(1)}$ based on the Strong Exponential Time Hypothesis, which, despite not yet matching our upper bound, strongly separates the discrete Fréchet distance under translation from the standard discrete Fréchet distance.

Our use of offline dynamic grid reachability yields further motivation for studying the offline setting of dynamic algorithms for potential use as subroutines in static algorithms. Problems left open by this article include: (1) Closing the gap between our upper and conditional lower bound. This might require a solution to offline dynamic grid reachability with polylogarithmic amortized update time. (2) Generalizing our bounds to $d = 1$ or higher dimensions $d \geq 3$, as in this article, we only considered curves in the plane. While generalizing our algorithm to $d = 1$ or $d \geq 3$ seems rather straightforward but technical, obtaining strong conditional lower bounds for these cases is more interesting. (3) Considering different transformations such as scaling, rotation, or affine transformations in general; here, we only treated translations. Significantly new ideas seem necessary to obtain meaningful lower bounds for other transformations. (4) Determine whether the time complexity of variants of the discrete Fréchet distance, such as the continuous or weak Fréchet distance, have similar or different relationships to their translation-invariant analogues.

REFERENCES

[1] Amir Abboud and Karl Bringmann. 2018. Tighter connections between formula-sat and shaving logs. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP'18)*.

[2] Amir Abboud and Søren Dahlgaard. 2016. Popular conjectures as a barrier for dynamic planar graph algorithms. In *Proceedings of the IEEE 57th Symposium on Foundations of Computer Science (FOCS'16)*. 477–486.

[3] Ittai Abraham, Shiri Chechik, and Cyril Gavoille. 2012. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC'12)*. 1199–1218.

[4] P. Agarwal, R. Ben Avraham, H. Kaplan, and M. Sharir. 2013. Computing the discrete Fréchet distance in subquadratic time. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*. 156–167.

[5] Helmut Alt and Maike Buchin. 2010. Can we compute the similarity between surfaces? *Discr. Comput. Geom.* 43, 1 (2010), 78–99.

[6] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.* 5, 1–2 (1995), 78–99.

[7] Helmut Alt, Christian Knauer, and Carola Wenk. 2001. Matching polygonal curves with respect to the Fréchet distance. In *Proceedings of the 18th Symposium on Theoretical Aspects of Computer Science (STACS'01)*. 63–74.

[8] Tetsuo Asano, David Kirkpatrick, Kotaro Nakagawa, and Osamu Watanabe. 2014. $\widetilde{O}(\sqrt{n})$-Space and polynomial-time algorithm for planar directed graph reachability. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS'14)*. 45–56.

[9] Ryo Ashida and Kotaro Nakagawa. 2018. $\widetilde{O}(n^{1/3})$-space algorithm for the grid graph reachability problem. In *Proceedings of the 34th International Symposium on Computational Geometry (SoCG'18)*. 5:1–5:13.
[10] Julian Baldus and Karl Bringmann. 2017. A fast implementation of near neighbors queries for Fréchet distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17)*. ACM, New York, NY. DOI: https://doi.org/10.1145/3139958.3140062
[11] Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. 2015. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Trans. Algor.* 11, 4 (2015), 29.
[12] Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. 2015. A faster algorithm for the discrete Fréchet distance under translation. ArXiv preprint http://arxiv.org/abs/1501.03724 (2015).
[13] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On map-matching vehicle tracking data. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05)*. 853–864.
[14] Karl Bringmann. 2014. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science (FOCS'14)*. 661–670.
[15] Karl Bringmann, Marvin Künnemann, and André Nusser. 2019. Fréchet distance under translation: conditional hardness and an algorithm via offline dynamic grid reachability. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*. 2902–2921. DOI: https://doi.org/10.1137/1.9781611975482.180
[16] Karl Bringmann, Marvin Künnemann, and AndrÃ© Nusser. 2019. Walking the dog fast in practice: Algorithm engineering of the Fréchet distance. In *Proceedings of the 35th International Symposium on Computational Geometry*, Vol. 129, Gill Barequet and Yusu Wang (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 17:1–17:21. DOI: 10.4230/LIPIcs.SoCG.2019.17
[17] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. 2011. Detecting commuting patterns by clustering subtrajectories. *Int. J. Comput. Geom. Appl.* 21, 3 (2011), 253–282.
[18] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. 2014. Four soviets walk the dog—with an application to alt's conjecture. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. 1399–1413.
[19] Kevin Buchin, Maike Buchin, and Yusu Wang. 2009. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms (SODA'09)*. 645–654.
[20] Kevin Buchin, Yago Diez, Tom van Diggelen, and Wouter Meulemans. 2017. Efficient trajectory queries under the Fréchet Distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17)*. ACM, New York, NY. DOI: https://doi.org/10.1145/3139958.3140064
[21] Erin Wolf Chambers, Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite. 2010. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Comput. Geom.* 43, 3 (2010), 295–311.
[22] Timothy M. Chan and Konstantinos Tsakalidis. 2017. dynamic orthogonal range searching on the ram, revisited. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG'17)*. 28:1–28:13.
[23] Bernard Chazelle. 1988. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.* 17, 3 (1988), 427–462.
[24] L. Paul Chew and Klara Kedem. 1992. Improvements on geometric pattern matching problems. In *Algorithm Theory - SWAT '92 (Lecture Notes in Computer Science)*, Otto Nurmi and Esko Ukkonen (Eds.). Springer Berlin, 318–325.
[25] Richard Cole. 1987. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM* 34, 1 (1987), 200–208.
[26] A. F. Cook and Carola Wenk. 2010. Geodesic Fréchet distance inside a simple polygon. *ACM Trans. Algor.* 7, 1 (2010), 193–204.
[27] Krzysztof Diks and Piotr Sankowski. 2007. Dynamic plane transitive closure. In *Proceedings of the 15th European Symposium on Algorithms (ESA'07)*. 594–604.
[28] Anne Driemel and Sariel Har-Peled. 2013. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Comput.* 42, 5 (2013), 1830–1866.
[29] Fabian Dütsch and Jan Vahrenhold. 2017. A filter-and-refinement-algorithm for range queries based on the Fréchet distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17)*. ACM, New York, NY. DOI: https://doi.org/10.1145/3139958.3140063
[30] Alon Efrat, Piotr Indyk, and Suresh Venkatasubramanian. 2001. Pattern matching for sets of segments. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*. 295–304.
[31] Thomas Eiter and Heikki Mannila. 1994. *Computing Discrete Fréchet Distance*. Technical Report CD-TR 94/64. Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria.
[32] Chenglin Fan and Benjamin Raichel. 2017. Computing the Fréchet gap distance. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG'17)*, Vol. 77. 42:1–42:16.

[33] Omrit Filtser and Matthew J. Katz. 2015. The Discrete Fréchet Gap. ArXiv preprint http://arxiv.org/abs/1506.04861 (2015).
[34] Omrit Filtser and Matthew J. Katz. 2018. Algorithms for the discrete Fréchet distance under translation. In *Proceedings of the 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT'18) (Leibniz International Proceedings in Informatics (LIPIcs))*, David Eppstein (Ed.), Vol. 101. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 20:1–20:14. DOI: https://doi.org/10.4230/LIPIcs.SWAT.2018.20
[35] Harold N. Gabow, Jon Louis Bentley, and Robert Endre Tarjan. 1984. Scaling and related techniques for geometry problems. In *Proceedings of the 16th ACM Symposium on Theory of Computing (STOC'84)*. 135–143.
[36] Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. 1993. The upper envelope of Voronoi surfaces and its applications. *Discr. Comput. Geom.* 9, 3 (1993), 267–291.
[37] Russell Impagliazzo and Ramamohan Paturi. 2001. On the complexity of k-SAT. *J. Comput. Syst. Sci.* 62 (2001), 367–375.
[38] Piotr Indyk. 2002. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proceedings of the 18th Symposium on Computer Geometry (SoCG'02)*. 102–106.
[39] Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, and Piotr Sankowski. 2017. Decremental single-source reachability in planar digraphs. In *Proceedings of the 49th ACM Symposium on Theory of Computing*. 1108–1121.
[40] Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. 2011. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC'11)*. 313–322.
[41] Minghui Jiang, Ying Xu, and Binhai Zhu. 2008. Protein structure–structure alignment with discrete Fréchet distance. *J. Bioinform. Comput. Biol.* 6, 01 (2008), 51–64.
[42] Philip N. Klein and Sairam Subramanian. 1998. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica* 22, 3 (1998), 235–249.
[43] Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. 2011. Fréchet distance with speed limits. *Comput. Geom.* 44, 2 (2011), 110–120.
[44] Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. 2014. Improved algorithms for partial curve matching. *Algorithmica* 69, 3 (July 2014), 641–657. DOI: https://doi.org/10.1007/s00453-013-9758-3
[45] Nimrod Megiddo. 1983. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM* 30, 4 (1983), 852–865.
[46] Christian Worm Mortensen. 2006. Fully dynamic orthogonal range reporting on RAM. *SIAM J. Comput.* 35, 6 (2006), 1494–1525.
[47] Axel Mosig and Michael Clausen. 2005. Approximately matching polygonal curves with respect to the Fréchet distance. *Comput. Geom.: Theor. Applic.* 30, 2 (2005), 113–127.
[48] Mario E. Munich and Pietro Perona. 1999. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of the 7th International Conference on Computer Vision (ICCV'99)*. 108–115.
[49] W. J. Rucklidge. 1996. Lower bounds for the complexity of the graph of the Hausdorff distance as a function of transformation. *Discr. Comput. Geom.* 16, 2 (Feb. 1996), 135–153. DOI: https://doi.org/10.1007/BF02716804
[50] Sairam Subramanian. 1993. A fully dynamic data structure for reachability in planar digraphs. In *Proceedings of the 1st European Symposium on Algorithms (ESA'93)*. 372–383.
[51] Carola Wenk. 2002. *Shape Matching in Higher Dimensions*. Ph.D. Dissertation. Freie Universität Berlin.
[52] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2–3 (2005), 357–365.

Received April 2019; revised June 2020; accepted April 2021