When Lipschitz Walks Your Dog: Algorithm Engineering of the Discrete Fréchet Distance Under Translation

Karl Bringmann
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max Planck Insitute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
kbringma@mpi-inf.mpg.de

Marvin Künnemann
Max Planck Insitute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
marvin@mpi-inf.mpg.de

André Nusser
Saarbrücken Graduate School of Computer Science, Saarland University, Germany
Max Planck Insitute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
anusser@mpi-inf.mpg.de

Abstract

Consider the natural question of how to measure the similarity of curves in the plane by a quantity that is *invariant under translations* of the curves. Such a measure is justified whenever we aim to quantify the similarity of the curves' *shapes* rather than their positioning in the plane, e.g., to compare the similarity of handwritten characters. Perhaps the most natural such notion is the (discrete) *Fréchet distance under translation*. Unfortunately, the algorithmic literature on this problem yields a very pessimistic view: On polygonal curves with n vertices, the fastest algorithm runs in time $\mathcal{O}(n^{4.667})$ and cannot be improved below $n^{4-o(1)}$ unless the Strong Exponential Time Hypothesis fails. Can we still obtain an implementation that is efficient on realistic datasets?

Spurred by the surprising performance of recent implementations for the Fréchet distance, we perform algorithm engineering for the Fréchet distance under translation. Our solution combines fast, but inexact tools from continuous optimization (specifically, branch-and-bound algorithms for global Lipschitz optimization) with exact, but expensive algorithms from computational geometry (specifically, problem-specific algorithms based on an arrangement construction). We combine these two ingredients to obtain an *exact decision* algorithm for the Fréchet distance under translation. For the related task of computing the distance *value* up to a desired precision, we engineer and compare different methods. On a benchmark set involving handwritten characters and route trajectories, our implementation answers a typical query for either task in the range of a few milliseconds up to a second on standard desktop hardware.

We believe that our implementation will enable, for the first time, the use of the Fréchet distance under translation in applications, whereas previous algorithmic approaches would have been computationally infeasible. Furthermore, we hope that our combination of continuous optimization and computational geometry will inspire similar approaches for further algorithmic questions.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Fréchet Distance, Computational Geometry, Continuous Optimization, Algorithm Engineering

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.25

Related Version https://arxiv.org/abs/2008.07510

Supplementary Material https://gitlab.com/anusser/frechet_distance_under_translation

Funding *Karl Bringmann*: 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).

Acknowledgements We thank Andreas Karrenbauer for helpful discussions.

28th Annual European Symposium on Algorithms (ESA 2020).
Editors: Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders; Article No. 25; pp. 25:1–25:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Consider the following natural task: Given two handwritings of (the same or different) characters, represented as polygonal curves π, σ in the plane, determine how similar they are. To measure the similarity of two such curves, several distance notions could be used, where the most popular measure in computational geometry is given by the *Fréchet distance* $d_F(\pi, \sigma)$: Intuitively, we imagine a dog walking on π and its owner walking on σ, and define $d_F(\pi, \sigma)$ as the *shortest leash length* required to connect the dog to its owner while both walk along their curves (only forward, but at arbitrarily and independently variable speeds). In this paper, we focus on the *discrete* version, in which dog and owner do not continuously walk along the curves, but jump from vertex to vertex.[1] As a fundamental curve similarity notion that takes into account the *sequence* of the points of the curves (rather than simply the set of points, as in the simpler notion of the Hausdorff distance), the discrete Fréchet distance and variants have received considerable attention from the computational geometry community, see, e.g. [4, 19, 12, 17, 3, 8, 11, 15]. While the fastest known algorithms take time $n^{2\pm o(1)}$ on polygonal curves with at most n vertices [4, 19, 3, 11] – which is best possible under the Strong Exponential Time Hypothesis [8] – a recent line of research [6, 13, 18, 10] gives fast implementations for practical input curves.

In the setting of handwritten characters, one would expect our notion of similarity to be *invariant under translations* of the curves; after all, translating one character in the plane while fixing the position of the other should not affect their similarity. In this sense, the original Fréchet distance seems inadequate, as it does not satisfy translation invariance. However, we may canonically define a translation-invariant adaptation as the minimum Fréchet distance between π and any translation of σ, yielding the *Fréchet distance under translation*. Note that beyond computing the similarity of handwritten characters, this measure is generally applicable whenever our intuitive notion of similarity is not affected by translations, such as recognition of movement patterns[2]. In some settings, we would expect our notion to additionally be scaling- or rotation-invariant; however, this is beyond the scope of this paper, as already the Fréchet distance under translation presents previously unresolved challenges.

Can we compute the Fréchet distance under translation quickly? The existing theoretical work yields a rather pessimistic outlook: For the discrete Fréchet distance under translation in the plane, the currently fastest algorithm runs in time $\mathcal{O}(n^{4.667})$, and any algorithm requires time $n^{4-o(1)}$ under the Strong Exponential Time Hypothesis [9]. These high polynomial bounds appear prohibitive in practice, and have likely impeded algorithmic uses of this similarity measure. (For the continuous analogue, the situation appears even worse, as the fastest algorithm has a significantly higher worst-case bound of $\mathcal{O}(n^8 \log n)$; we thus solely consider the discrete version in this work.) Given the surprising performance of recent Fréchet distance implementations on realistic curves [35, 10], can we still hope for faster algorithms on realistic inputs also for its translation-invariant version?

Our problem. Towards making the Fréchet distance under translation applicable for practical applications, we engineer a fast implementation and analyze it empirically on realistic input sets. Perhaps surprisingly, our fastest solution for the problem combines inexact

[1] We give a precise definition in Section 2.

[2] One may argue that the similarity of movement patterns also depends on the speed/velocity of the motion. In principle, we can also incorporate such information into any Fréchet-distance-based measure by introducing an additional dimension.

continuous optimization techniques with an exact, but expensive problem-specific approach from computational geometry to obtain an *exact decision* algorithm. We discuss our approach in Section 3 and present the details of our decision algorithm in Section 4. We develop our approach also for the related, but different task to compute the distance value up to a given precision in Section 5, and evaluate our solutions for both settings in comparison to baseline approaches in Section 6.

Further related work. Variations of the distance measure studied in this paper arise by choosing (1) the discrete or continuous Fréchet distance, (2) the dimension d of the ambient Euclidean space, and (3) a class of transformations, e.g., translations, rotations, scaling, or arbitrary linear transformations. A detailed treatment of algorithms for this class of distance measures can be found in [34]. The earliest example of a problem in this class is the continuous Fréchet distance under translations in dimension $d = 2$, which was introduced by Alt et al. [5] together with an $\mathcal{O}(n^8 \log n)$-time algorithm.

In this paper we focus on the discrete Fréchet distance under translation in the plane. This problem was first studied by Mosig and Clausen [31], who gave an $\mathcal{O}(n^4)$ algorithm for approximating the discrete Fréchet distance under rigid motions. Subsequently, Jiang et al. [29] presented an $\mathcal{O}(n^6 \log n)$-time algorithm for the exact Fréchet distance under translation. Their running time was improved by Ben Avraham et al. to $\mathcal{O}(n^5 \log n)$ [7], and then by Bringmann et al. to $\mathcal{O}(n^{4.667})$ [9]. A conditional lower bound of $n^{4-o(1)}$ can be found in [9].

Algorithm engineering efforts for the Fréchet distance were initiated by the SIGSPATIAL GIS Cup 2017 [35], where the task was to implement a nearest neighbor data structure for curves under the Fréchet distance; see [6, 13, 18] for the top three submissions. The currently fastest implementation of the Fréchet distance is due to Bringmann et al. [10]. Further recent directions of Fréchet-related algorithm engineering include k-means clustering of trajectories [14] and locality sensitive hashing of trajectories [16].

2 Preliminaries

Throughout the paper, we consider the Euclidean plane and denote the Euclidean norm by $\|\cdot\|$. A *polygonal curve* π is a sequence $\pi = (\pi_1, \dots, \pi_n)$ of vertices $\pi_i \in \mathbb{R}^2$. For any $\tau \in \mathbb{R}^2$, we write $\pi + \tau$ for the translated curve $(\pi_1 + \tau, \dots, \pi_n + \tau)$.

For any curves $\pi = (\pi_1, \dots, \pi_n), \sigma = (\sigma_1, \dots, \sigma_m)$, we define their *discrete Fréchet distance* as follows. A *traversal* is a sequence $T = (\,(p_1, s_1), \dots, (p_t, s_t)\,)$ of pairs $(p_i, s_i) \in [n] \times [m]$ such that $(p_1, s_1) = (1, 1)$, $(p_t, s_t) = (n, m)$ and $(p_{i+1}, s_{i+1}) \in \{(p_i{+}1, s_i), (p_i, s_i{+}1), (p_i{+}1, s_i{+}1)\}$ for all $1 \leq i < t$. The width of a traversal is $\max_{i=1,\dots,|T|} \|\pi_{p_i} - \sigma_{s_i}\|$. The discrete Fréchet distance is then defined as the smallest width over all traversals, i.e.,

$$d_F(\pi, \sigma) := \min_{\text{traversal } T} \max_{i=1,\dots,|T|} \|\pi_{p_i} - \sigma_{s_i}\| .$$

As we only consider the discrete Fréchet distance in this paper, we drop "discrete" in the remainder. To avoid confusion, we also refer to it as the *fixed-translation* Fréchet distance.

As the canonically translation-invariant variant of the discrete Fréchet distance, we define the *discrete Fréchet distance under translation* as $d_{\text{trans-}F}(\pi, \sigma) := \min_{\tau \in \mathbb{R}^2} d_F(\pi, \sigma + \tau)$. We typically view the problem as a two-dimensional optimization problem with objective function $f(\tau) := d_F(\pi, \sigma + \tau)$. Specifically, we consider the task to decide $\min_{\tau \in \mathbb{R}^2} f(\tau) \leq \delta$*?* (*exact decider*) or to return a value in the range $[(1 - \epsilon) \min_{\tau \in \mathbb{R}^2} f(\tau), (1 + \epsilon) \min_{\tau \in \mathbb{R}^2} f(\tau)]$ (*approximate value computation*, multiplicative version). In fact, for implementation reasons (for a discussion, see the full version of the paper), our implementation returns a value in $[\min_{\tau\in\mathbb{R}^2} f(\tau) - \epsilon, \min_{\tau\in\mathbb{R}^2} f(\tau) + \epsilon]$ (*approximate value computation*, additive version) using a straightforward adaptation of our approach.

Apart from a black-box Fréchet oracle answering decision queries $d_F(\pi, \sigma + \tau) \le \delta$?, our algorithms only exploit the following simple properties:

▶ **Observation 1** (Lipschitz property). *The objective function f is 1-Lipschitz, i.e., $|f(\tau) - f(\tau + \tau')| \le \|\tau'\|$.*

Proof. Note that for any $\pi_i, \sigma_j, \tau, \tau' \in \mathbb{R}^2$, we have

$$|\|\pi_i - (\sigma_j + \tau + \tau')\| - \|\pi_i - (\sigma_j + \tau)\|| \le \|\tau'\|$$

by triangle inequality. Thus, the widths of any traversal T for $\pi, \sigma + \tau$ and $\pi, \sigma + \tau + \tau'$ differ by at most $\|\tau'\|$, which immediately yields the observation. ◀

We obtain a simple 2-approximation of the Fréchet distance under translation as follows.

▶ **Observation 2.** *Let $\tau_{\text{start}} := \pi_1 - \sigma_1$ be the translation of σ that aligns the first points of π and σ. Then $d_F(\pi, \sigma + \tau_{\text{start}}) \le 2 \cdot d_{\textit{trans-F}}(\pi, \sigma)$.*

Analogously, for $\tau_{\text{end}} := \pi_n - \sigma_m$, we have $d_F(\pi, \sigma + \tau_{\text{end}}) \le 2 \cdot d_{\textit{trans-F}}(\pi, \sigma)$.

Proof. Let $\delta^* := d_{\text{trans-}F}(\pi, \sigma)$ and let τ^* be such that $d_F(\pi, \sigma + \tau^*) = \delta^*$, which implies in particular that $\|\pi_1 - (\sigma_1 + \tau^*)\| \le \delta^*$. Thus, $\|\tau_{\text{start}} - \tau^*\| = \|\pi_1 - (\sigma_1 + \tau^*)\| \le \delta^*$. Thus by Observation 1, we obtain $d_F(\pi, \sigma + \tau_{\text{start}}) \le d_F(\pi, \sigma + \tau^*) + \delta^* = 2\delta^*$. ◀

Note that the above observation gives a formal guarantee of a simple heuristic: translate the curves such that the start points match, and compute the corresponding fixed-translation Fréchet distance. Unfortunately, this worst-case guarantee is tight[3] – a correspondingly large discrepancy is also observed on our data sets.

3 Our Approach: Lipschitz meets Fréchet

To obtain a fast exact decider, we approach the problem from two different angles: First, we review previous problem-specific approaches to the Fréchet distance under translation, all relying on the construction of an arrangement of circles as an essential tool from computational geometry. Second, we cast the problem into the framework of global Lipschitz optimization with its rich literature on fast, numerical solutions. In isolation, both approaches are inadequate to obtain a fast, exact decider (as the arrangement can be prohibitively large even for realistic data sets, and black-box Lipschitz optimization methods cannot return an exact optimum). We then describe how to combine both approaches to obtain a fast implementation of an exact decider for the discrete Fréchet distance under translation in the plane. We evaluate our approach, including comparisons to (typically computationally infeasible) baseline approaches, on a data set that we craft from sets of handwritten character and (synthetic) GPS trajectories used in the ACM SIGSPATIAL GIS Cup 2017 [2, 1]. We believe that our approach will inspire similar combinations of fast, inexact methods from continuous optimization with expensive, but exact approaches from computational geometry also in other contexts.

---

3 To see this, take any segment in the plane and let π traverse it in one direction, and σ in the other. Then the heuristic would return as estimate two times the segment length (the distance of the translated end points), while the optimal translation aligns the segments and achieves the segment length as Fréchet distance.

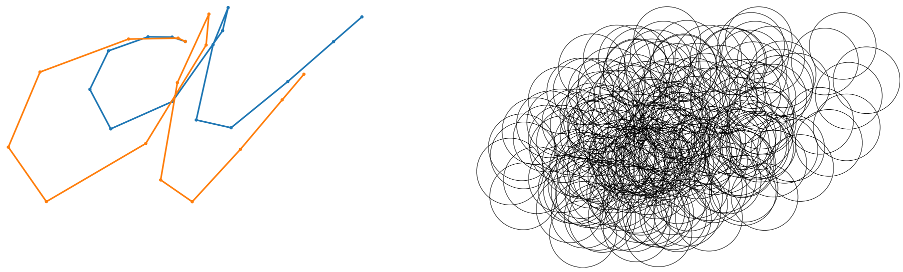


Figure 1 Example curves π, σ (left) together with their arrangement $\mathcal{A}_\delta$ (right), $\delta = d_{\text{trans-}F}(\pi, \sigma)$.

3.1 View I: Arrangement-based Algorithms

Previous algorithms for the Fréchet distance under translation in the plane work as follows. Given two polygonal curves π, σ and a decision distance δ, consider the set of circles

$$\mathcal{C} := \{C_\delta(\pi_i - \sigma_j) \mid \pi_i \in \pi,\ \sigma_j \in \sigma\},$$

where $C_r(p)$ denotes the circle of radius $r \in \mathbb{R}$ around $p \in \mathbb{R}^2$. Define the arrangement $\mathcal{A}_\delta$ as the partition of $\mathbb{R}^2$ induced by $\mathcal{C}$. The decision of $d_F(\pi, \sigma + \tau) \leq \delta$ is then uniform among all $\tau \in \mathbb{R}$ in the same face of $\mathcal{A}_\delta$ (for a detailed explanation, we refer to [7, Section 3] or [9]). Thus, it suffices to check, for each face f of $\mathcal{A}_\delta$, an arbitrarily chosen translation $\tau_f \in f$. Specifically, the Fréchet distance under translation is bounded by δ if and only if there is some face f of $\mathcal{A}_\delta$ such that $d_F(\pi, \sigma + \tau_f) \leq \delta$. Since the arrangement $\mathcal{A}_\delta$ has size $\mathcal{O}(n^4)$ and can be constructed in time $\mathcal{O}(n^4)$ [29], using the standard $\mathcal{O}(n^2)$-time algorithm for the fixed-translation Fréchet distance [19, 4] to decide $d_F(\pi, \sigma + \tau_f) \leq \delta$ for each face f, we immediately arrive at an $\mathcal{O}(n^6)$-time algorithm.

Subsequent improvements [7, 9] speed up the decision of $d_F(\pi, \sigma + \tau_f) \leq \delta$ for all faces f by choosing an appropriate ordering of the translations τ_f and designing data structures that avoid recomputing some information for "similar" translations, leading to an $\mathcal{O}(n^{4.667})$-time algorithm. Still, these works rely on computing the arrangement $\mathcal{A}_\delta$ of worst-case size $\Theta(n^4)$, and a conditional lower bound indeed rules out $\mathcal{O}(n^{4-\epsilon})$-time algorithms [9].

Drawback: The arrangement size bottleneck. Despite the worst-case arrangement size of $\Theta(n^4)$ and the conditional lower bound in [9], which indeed constructs such large arrangements, one might hope that realistic instances often have much smaller arrangements. If so, a combination with a practical implementation of the fixed-translation Fréchet distance could already give an algorithm with reasonable running time. Unfortunately, this is not the case: our experiments in this paper exhibit typical arrangement sizes between 10^6 to 10^8 for curves of length $n \approx 200$, see Figure 5 in Section 6. Also see Figure 1 which illustrates a large arrangement already on curves with 15 vertices, subsampled from our benchmark sets of realistic curves.

This renders a purely arrangement-based approach infeasible: As existing implementations for the Fréchet distance typically answer queries within few microseconds, we would expect an average decision time between a few seconds and several minutes already for a single decision query for the Fréchet distance under translation. Thus, a reasonable approximation of the distance value via binary search would take between a minute and over an hour.

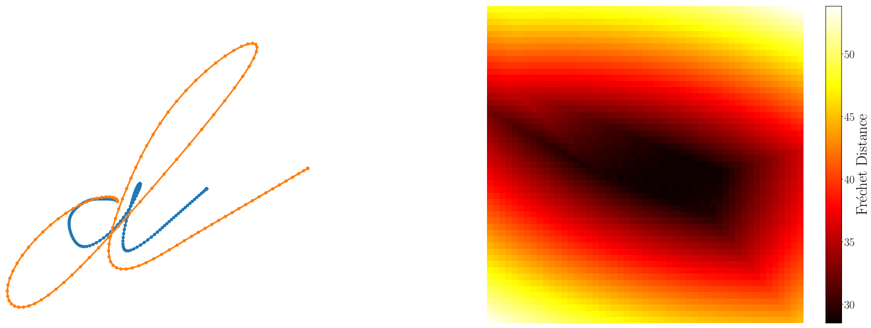


Figure 2 Example curves π, σ (left) together with a plot of the resulting non-convex objective function $f(\tau) = d_F(\pi, \sigma + \tau)$. For a closer look at the area close to the optimal translation (and highly non-convex small-scale artefacts), we refer to Figure 3.

3.2 View II: A Global Lipschitz Optimization problem

A second view on the Fréchet distance under translation results from a simple observation: For any polygonal curves π, σ and any translation $\tau \in \mathbb{R}^2$, we have $|d_F(\pi, \sigma+\tau) - d_F(\pi, \sigma)| \leq \|\tau\|_2$, see Section 2. As a consequence, the Fréchet distance under translation is the minimum of a function $f(\tau) \coloneqq d_F(\pi, \sigma + \tau)$ that is 1-Lipschitz (i.e., $|f(x) - f(x+y)| \leq \|y\|_2$ for all x, y). This suggests to study the problem also from the viewpoint of the generic algorithms developed for optimizing Lipschitz functions by the continuous optimization community.

Following the terminology of [25], in an *unconstrained bivariate global Lipschitz optimization problem*, we are given an objective function $f : \mathbb{R}^2 \to \mathbb{R}$ that is 1-Lipschitz, and the aim is to minimize $f(x)$ over $x \in B \coloneqq [a_1, b_1] \times [a_2, b_2]$; we can access f only by evaluating it on (as few as possible) points $x \in B$. Note that in this abstract setting, we cannot optimize f exactly, so we are additionally given an error parameter $\epsilon > 0$ and the precise task is to find a point $x \in B$ such that $f(x) \leq \min_{z \in B} f(z) + \epsilon$.

Global Lipschitz optimization techniques have been studied from an algorithmic perspective for at least half a century [32]. This suggest to explore the use of the fast algorithms developed in this context to obtain at least an *approximate* decider for the discrete Fréchet distance under translation. Indeed, our problem fits into the above framework, if we take the following considerations into account:

(1) Finite Box Domain: While we seek to minimize $f(\tau) = d_F(\pi, \sigma + \tau)$ over $\tau \in \mathbb{R}^2$, the above formulation assumes a finite box domain B. To reconcile this difference, observe that any translation τ achieving a Fréchet distance of at most δ must translate the first (last) point of σ such that the first (last) point of π is within distance at most δ. Thus, any feasible translation τ must be contained in the intersection of the two corresponding disks, and we can use any bounding box of this intersection as our box domain B.

(2) (Approximate) Decision Problem: While we seek to decide "$\min_\tau f(\tau) \leq \delta$", the above formulation solves the corresponding minimization problem. Note that approximate minimization can be used to *approximately* solve the decision problem, but *exactly* solving the decision problem is impossible in the above framework.

(3) Oracle Access to $f(\tau)$: Evaluation of $f(\tau)$ corresponds to computing the Fréchet distance of π and $\sigma + \tau$, for which we can use previous fast implementations [6, 13, 18, 10]. (Actually, these algorithms were designed to answer decision queries of the form "$f(\tau) \leq \delta$?"; we discuss this aspect at the end of this section.)

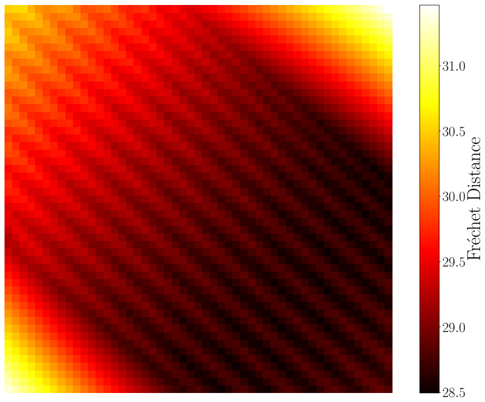


Figure 3 Highly non-convex artefacts of the objective function at a local scale, resulting particularly from the notion of traversals in the *discrete* Fréchet distance.

In Figure 2, we illustrate our view of the Fréchet distance under translation as Lipschitz optimization problem. As the figure suggests, on many realistic instances, the problem appears well-behaved (almost convex) at a global scale; using the Lipschitz property, one should be able to quickly narrow down the search space to small regions of the search space[4]. Particularly for this task, it is very natural to consider branch-and-bound approaches, as pioneered by Galperin [20, 21, 22, 23] and formalized by Horst and Tuy [26, 27, 28], since these have been applied very successfully for low-dimensional Global Lipschitz optimization (and non-convex optimization in general).

On a high level, in this approach we maintain a global upper bound $\tilde{\delta}$ and a list of search boxes $B_1, \dots, B_b$ with lower bounds $\ell_1, \dots, \ell_b$ (i.e., $\min_{\tau \in B_i} f(\tau) \geq \ell_i$) obtained via the Lipschitz condition. We iteratively pick some search box B_i and first try to improve the global upper bound $\tilde{\delta}$ or the local lower bound ℓ_i using a small number of queries $f(\tau)$ with $\tau \in B_i$ (and exploiting the Lipschitz property). If the local lower bound exceeds the global upper bound, i.e., $\ell_i > \tilde{\delta}$, we drop the search box B_i, otherwise, we split B_i into smaller search boxes. The procedure stops as soon as $\tilde{\delta} \leq (1+\epsilon) \min_i \ell_i$, which proves that $\tilde{\delta}$ gives a $(1+\epsilon)$-approximation to the global minimum.

Specifically, we arrive at the following branch-and-bound strategy proposed by Gourdin, Hansen and Jaumard [24]. We specify it by giving the rules with which it (i) attempts to update the global upper bound, (ii) selects the next search box from the set of current search boxes, (iii) splits a search box if it remains active after bounding, and (iv) determines the local lower bounds.[5]

(i) **Upper Bounding Rule:** We evaluate f at the center τ_i of the current search box B_i.

(ii) **Selection Rule:** We pick the search box with the smallest lower bound (ties are broken arbitrarily).

(iii) **Branching Rule:** We split the current search box along its longest edge into 2 equal-sized subproblems.

(iv) **Lower Bounding Rule:** We obtain the local lower bound ℓ_i as $f(\tau_i) - d$ where d is the half-diameter of the current box. (Since f is 1-Lipschitz, we indeed have $\min_{\tau \in B_i} f(\tau) \geq \ell_i$.)

[4] For an illustration that highly non-convex behavior may still occur at a local level, we refer to Figure 3.

[5] See [25] for a precise formalization of the generic branch-and-bound algorithm that leaves open the instantiation of these rules. In any case, we give a self-contained description of our algorithms in Section 4 and 5.

One may observe that the chosen selection rule (also known as Best-Node First) is a no-regret strategy in the sense that no other selection rule, *even with prior knowledge of the global optimum*, considers fewer search boxes (see, e.g., [36, Section 7.4]).

Drawback: Inexactness. Unfortunately, the above branch-and-bound approach for Lipschitz optimization fundamentally cannot return an exact global optimum, and thus yields only an approximate decider.

In a somewhat similar vein, in the above framework we assume that we can evaluate $f(\tau)$ quickly. Previous implementations for the fixed-translation Fréchet distance focus on the decision problem "$f(\tau) \leq \delta$?", not on determining the value $f(\tau)$. Both precise computations (via parametric search) or approximate computations (using a binary search up to a desired precision) are significantly more costly, raising the question how to make optimal use of the cheaper decision queries.

4 Contribution I: An Exact Decider by Combining Both Views

Our first main contribution is engineering an exact decider for the discrete Fréchet distance under translation by combining the two approaches. On a high level, we *globally* perform the branch-and-bound strategy described in the Lipschitz optimization view in Section 3.2, but use as a base case a *local* version of the arrangement-based algorithms of Section 3.1 once the arrangement size in a search box is sufficiently small. As each search box is thus resolved exactly, this yields an exact decider. More precisely, our final algorithm is a result of the following steps and adaptations:

(1) **Fréchet Decision Oracle.** We adapt the currently fastest implementation of a decider for the continuous fixed-translation Fréchet distance [10] to the discrete fixed-translation Fréchet distance. Furthermore, to handle many queries for the same curve pair *under different translations* quickly, we incorporate an implicit translation so that curves do not need to be explicitly translated for each query translation τ.

(2) **Objective Function Evaluation.** For our exact decider, the branch-and-bound strategy in Section 3.2 simplifies significantly: We do not maintain a global upper bound and local lower bounds ℓ_i, but for each box only test whether $f(\tau_i) \leq \delta$ (if so, we return YES) or whether $f(\tau_i) > \delta + d$ (this corresponds to updating the local lower bound beyond δ, i.e., we may drop the box completely). Therefore, we may use an arbitrary selection rule. Note that we only require decision queries to the fixed-translation Fréchet algorithm.

(3) **Base Case.** We implement a *local* arrangement-based algorithm: For a given search box B_i, we (essentially) construct the arrangement $\mathcal{A} \cap B_i$ using CGAL [33], and test, for each face f of $\mathcal{A} \cap B_i$, some translation $\tau' \in f$ for $f(\tau') \leq \delta$. This yields the algorithm that we may use as a base case.

(4) **Base Case Criterion.** For each search box, we compute an estimate of its arrangement complexity. If this estimate is smaller than a (tunable) parameter γ_{size}, or the depth of the branch-and-bound recursion for the current search box exceeds a parameter γ_{depth}, then we use the localized arrangement-based algorithm.

(5) **Benchmark and Choice of Parameters.** We choose the size and depth parameters $\gamma_{\text{size}}, \gamma_{\text{depth}}$ guided by a benchmark set that we create from a set of handwritten characters and synthetic GPS trajectories.

The pseudocode of the resulting algorithm is shown in Algorithm 1. For a more detailed description of our Fréchet-under-translation decider, we refer to the full version of this paper.

Algorithm 1 Algorithm for deciding the Fréchet distance under translation. We use τ_B to denote the center of the box B and d_B to denote the length of the diagonal of B.

```
procedure Decider(π, σ, δ)
    decide trivial NO instances with empty initial search box quickly
    Q ← Fifo(initial search box)
    while Q ≠ ∅ do
        B ← extract front of search box queue Q
        if FréchetDistance(π, σ + τ_B) > δ + d_B/2 then          ▷ Lower Bounding
            skip B
        if FréchetDistance(π, σ + τ_B) ≤ δ then                  ▷ Upper Bounding
            return YES

        u ← upper bound on arrangement size inside B
        if u = 0 then                                             ▷ Arrangement-based Base Case
            skip B
        else if u ≤ γ_size or layer of B is γ_depth then
            if local arrangement-based algorithm on π, σ, δ, B returns YES then
                return YES
            else
                skip B

        halve B along longest edge and push resulting child boxes to Q    ▷ Branching
    return NO
```

5 Contribution II: Computation of the Distance Value

In this section we present our second main contribution: an algorithm for computing the value of the Fréchet distance under translation. Thus, we now focus on the functional task of computing the value $d_{\text{trans-}F}(\pi, \sigma) = \min_{\tau \in \mathbb{R}^2} d_F(\pi, \sigma + \tau)$, in contrast to the previously discussed decision problem "$d_{\text{trans-}F}(\pi, \sigma) \leq \delta$?". In theory, one could use the paradigm of parametric search [30], see [7, 9] for details for the discrete case. However, it is rarely used in practice as it is non-trivial to code, and computationally costly. Instead, as in most conceivable settings an estimate with small multiplicative error $(1 \pm \epsilon)$ with, e.g., $\epsilon = 10^{-7}$, suffices, we consider the problem of computing an estimate in $(1 \pm \epsilon) d_{\text{trans-}F}(\pi, \sigma)$.

There are several possible approaches to obtain an approximation with multiplicative error $(1 \pm \epsilon)$ for arbitrarily small $\epsilon > 0$:

1. **ϵ-approximate Set:** A natural approach underlying previous approximation algorithms [5] is to generate a set of $f(1/\epsilon)$ candidate translations T such that the best translation $\tau \in T$ gives a $(1+\epsilon)$-approximation for the Fréchet distance under translation. Unfortunately, in the plane such a set is of size $\Theta(1/\epsilon^2)$, which is prohibitively large for approximation guarantees such as $\epsilon = 10^{-7}$.
2. **Binary Search via Decision Problem:** A further canonical approach is to reduce the $(1+\epsilon)$-approximate computation task to the decision problem using a binary search.
3. **Lipschitz-only Optimization:** The main drawback of the generic Lipschitz optimization algorithms discussed in Section 3.2 was that they cannot be used to derive an exact answer. This drawback no longer applies for approximate value computation. We can thus use a pure branch-and-bound algorithm for global Lipschitz optimization. Interestingly,

Algorithm 2 Algorithm of our Lipschitz-Meets-Fréchet (LMF) algorithm for approximate value computation. We use τ_B to denote the center of the box and d_B to denote the length of the diagonal.

procedure LMF(π, σ)
    Preprocessing: build data structures for fast arrangement estimation and construction
    compute initial distance interval $[\delta_{\text{LB}}, \delta_{\text{UB}}]$ containing $d_{\text{trans-}F}(\pi, \sigma)$
    initialize global upper bound $\tilde{\delta} \leftarrow \delta_{\text{UB}}$
    $Q \leftarrow$ PriorityQueue(initial search box B_1 with local lower bound $\ell_{B_1} \leftarrow \delta_{\text{LB}}$)
    while $Q \neq \emptyset$ **do**
        $B \leftarrow$ box with smallest local lower bound ℓ_B in Q
        if $\tilde{\delta} \leq \ell_B(1+\epsilon)$ **then**
            skip B
        if FréchetDistance($\pi, \sigma + \tau_B$) $\leq \tilde{\delta}$ **then** ▷ *Upper/Lower Bounding*
            compute value $d_F(\pi, \sigma + \tau_B)$ with high precision and update $\tilde{\delta}$ and ℓ_B
        else
            if FréchetDistance($\pi, \sigma + \tau_B$) $> \tilde{\delta} + d_B/2$ **then**
                skip B
            compute value $d_F(\pi, \sigma + \tau_B)$ with coarse precision and update ℓ_B
        if $\tilde{\delta} \leq \ell_B(1+\epsilon)$ **then**
            skip B
        $u \leftarrow$ upper bound on arrangement size inside B for $\delta \in [\ell_B, \tilde{\delta}]$
        if $u = 0$ **then** ▷ *Arrangement-based Base Case*
            skip B
        else if $u \leq \gamma_{\text{size}}$ or layer of B is γ_{depth} **then**
            update $\tilde{\delta}$ via binary search over arrangement algorithm on B and $\delta \in [\ell_B, \tilde{\delta}]$
            skip B

        push child boxes of B to Q with local lower bounds set to ℓ_B ▷ *Branching*
    return $\tilde{\delta}$

experiments reveal that on many natural instances, significant time is spent branching inside very small regions. This suggests following our combined approach also for the value computation setting.

4. **Our solution, Lipschitz-meets-Fréchet:** We follow our approach of combining Lipschitz optimization with arrangement-based algorithms (described in Section 3) to compute a $(1+\epsilon)$-approximation of the distance value. As opposed to the decision algorithm, we indeed maintain a global upper bound $\tilde{\delta}$ and local lower bounds ℓ_i for each search box B_i. To update these bounds, we approximately evaluate the objective function $f(\tau)$ using a tuned binary search[6] over the fixed-translation Fréchet decider algorithm. We stop branching in a search box B_i if either the global upper bound $\tilde{\delta}$ is at most $\ell_i(1+\epsilon)$, or a base case criterion similar to the decision setting applies. As selection strategy, we employ the no-regret strategy of choosing the box with the smallest lower bound first. The base case performs a binary search using the local arrangement-based *decision* algorithm; thus, our upper bound on the arrangement size must hold for *all* δ in the search interval. The pseudocode of our solution is shown in Algorithm 2.

[6] We tune the binary search by distinguishing the precision with which we want to evaluate $f(\tau)$; intuitively, it pays off to evaluate $f(\tau)$ with high precision if this evaluation yields a better global upper bound, while for improvements of a local lower bound, a cheaper evaluation with coarser precision suffices.

For a more detailed description of the algorithm, we refer to the full version of this paper. As our experiments reveal, our solution generally outperforms the above described alternatives, see Section 6.

6 Experiments

To engineer and evaluate our approach, we provide a benchmark on the basis of the curve datasets that were used to evaluate the currently fastest fixed-translation Fréchet decider implementation in [10]. In particular, this curve set involves a set of handwritten characters (Characters, [2]) and the data set of the GIS Cup 2017 (Sigspatial, [1]). Table 1 gives statistics of these datasets.

Table 1 Information about the data sets used in the benchmarks.

Data set	Type	#Curves	Mean #vertices
Sigspatial [1]	synthetic GPS-like	20199	247.8
Characters [2]	20 handwritten characters	2858 (142.9 per character)	120.9

The aim of our evaluations is to investigate the following main questions:

1. Is our solution able to decide queries on realistic curve sets in an amount of time that is practically feasible, even when the size of the arrangement suggests infeasibility?
2. Is our combination of Lipschitz optimization and arrangement-based algorithms for value computation superior to the alternative approaches described in Section 5?

Furthermore, we aim to provide an understanding of the performance of our novel algorithms.

Decider experiments. For decision queries of the form "$d_{\text{trans-}F}(\pi, \sigma) \leq \delta$?", we generate a benchmark query set that distinguishes between how close the test distance is to the actual distance of the curves: Given a set of curves C, we sample 1000 curve pairs $\pi, \sigma \in C$ uniformly at random. Using our implementation, we determine an interval $[\delta_{\text{LB}}, \delta_{\text{UB}}]$ such that $\delta_{\text{UB}} - \delta_{\text{LB}} \leq 2 \cdot 10^{-7}$ and $d_{\text{trans-}F}(\pi, \sigma) \in [\delta_{\text{LB}}, \delta_{\text{UB}}]$. For each $\ell \in \{-10, \ldots, -1\}$, we add "$d_{\text{trans-}F}(\pi, \sigma) \leq (1 - 4^{\ell})\delta_{\text{LB}}$?" to the query set C_ℓ^{NO}, which contains only NO instances. Similarly, for each $\ell \in \{-10, \ldots, 2\}$ we add "$d_{\text{trans-}F}(\pi, \sigma) \leq (1 + 4^{\ell})\delta_{\text{UB}}$?" to the query set C_ℓ^{YES}, which contains only YES instances. We depict our timing results in Figure 4 on the Characters and Sigspatial data sets. Further experiments are given in the full version of the paper.

To give an insight for the speed-up achieved over the baseline arrangement-based algorithm that makes a black-box call to the fixed-translation Fréchet decider for each face of the arrangement $\mathcal{A}_\delta$, in Figure 5 we depict both the number of black-box calls to the fixed-translation Fréchet decider made by our implementation, as well as an estimate[7] for the arrangement size, and thus the number of black-box calls of the baseline approach.

[7] We only give an estimate for the arrangement size, since the size of the arrangement is too large to be evaluated exactly for all our benchmark queries within a day. Specifically, we estimate the number of vertices of the arrangement which closely corresponds to the number of faces by Euler's formula. We give the following estimate: We first determine a search box B for the given decision instance $\pi = (\pi_1, \ldots, \pi_n), \sigma = (\sigma_1, \ldots, \sigma_m), \delta$ as described for our algorithm. We then sample $S = 100000$ tuples $i_1, i_2 \in \{1, \ldots, n\}, j_1, j_2 \in \{1, \ldots, m\}$ and count the number I of intersections of the circles of radius δ around $\pi_{i_1} - \sigma_{j_1}$ and $\pi_{i_2} - \sigma_{j_2}$ inside B. The number $(I/S) \cdot (nm)^2$ is the estimated number of circle-circle intersections in B. Adding the number of circle-box intersections, which we can compute exactly, yields our estimate.

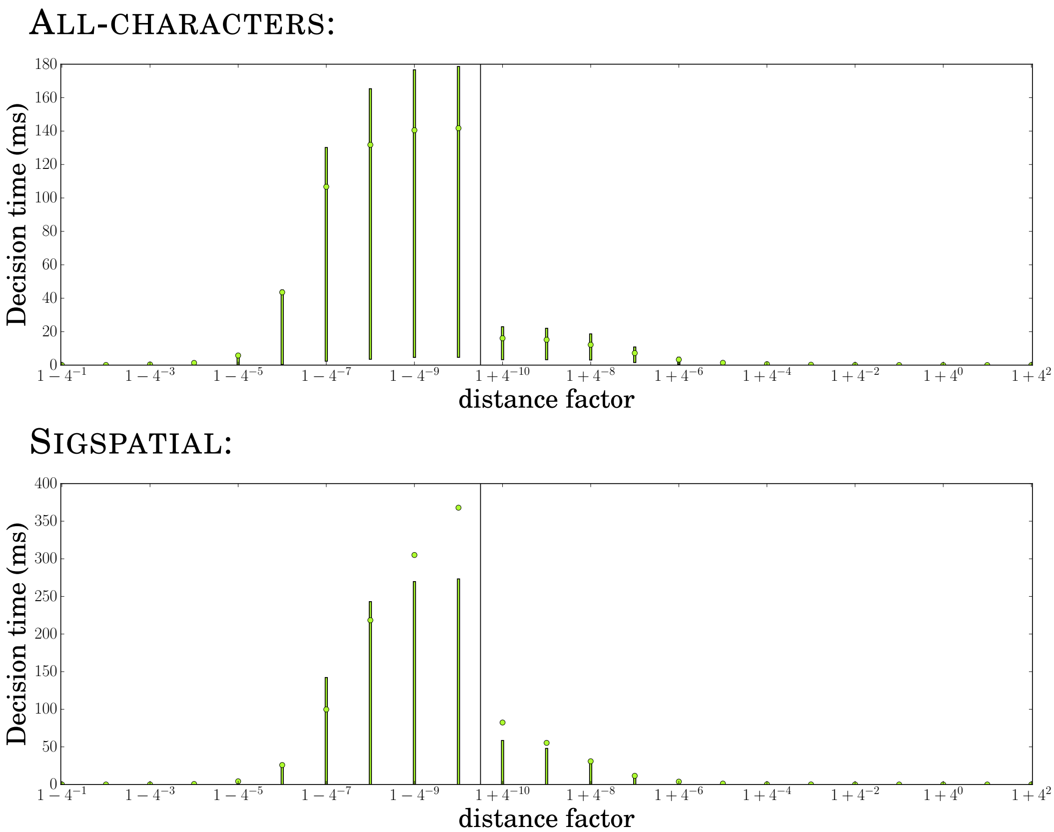


Figure 4 Running time for our decider. We plot the mean running times over 1000 NO (or YES) queries with a test distance of approximately $(1-4^{-\ell})$ (or $(1+4^{-\ell})$) times the true Fréchet distance under translation, as well as the interval between the lower and upper quartile over the queries.

We observe that on the above sets, the average decision time ranges from below 1 ms to 142 ms, deciding our Characters benchmark (involving 23, 000 queries) in 628 seconds. Our estimation suggests that a naive implementation of the baseline arrangement-based algorithm would have been worse by more than *three orders of magnitude*, as for each set, the average number of black-box calls to the fixed-translation Fréchet decider is smaller by a factor of at least 1000 than our estimation of the arrangement size.

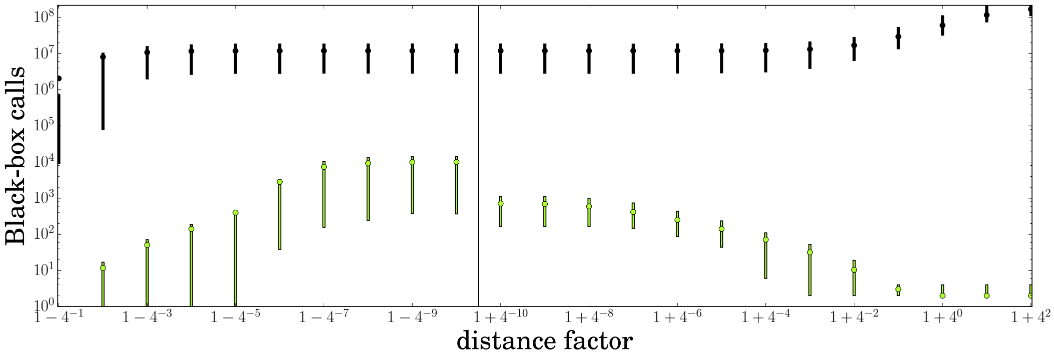
Approximate value computation experiments. We evaluate our implementation of the algorithm presented in Section 5 by computing an estimate $\tilde{\delta}$ such that $|\tilde{\delta} - d_{\text{trans-}F}(\pi, \sigma)| \leq \epsilon$ with a choice of $\epsilon = 10^{-7}$.[8] In particular, we compare the performances of the different approaches discussed in Section 5:

- **Binary Search:** Binary search using our Fréchet-under-translation decider of Section 4.
- **Lipschitz-only:** Algorithm 2 without the arrangement, i.e., without lines 18 to 23.
- **Lipschitz-meets-Fréchet (LMF):** Our implementation as detailed in Section 5.

Since simple estimates show that the ϵ-approximate sets are clearly too costly for $\epsilon = 10^{-7}$, we drop this approach from all further consideration. We took care to implement all approaches with a similar effort of low-level optimizations.

[8] Here we use additive rather than multiplicative approximation for technical reasons. Since all computed distances are within $[1.6, 120.7]$, this also yields a multiplicative $(1+\epsilon)$-approximation with $\epsilon \leq 10^{-7}$.

ALL-CHARACTERS:



SIGSPATIAL:

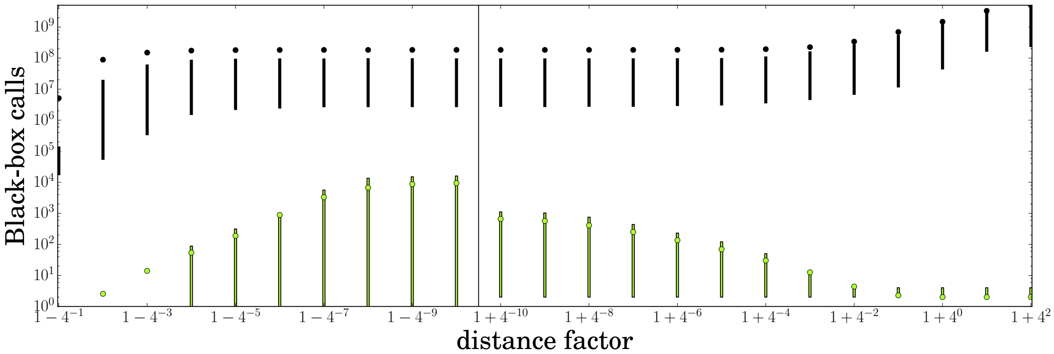


Figure 5 Number of black-box calls to the fixed-translation Fréchet decider made by our decider (below, in green), as well as an estimate of the arrangement complexity, i.e., number of calls of a naive algorithm (above, in black). We plot the mean number of calls and arrangement complexity over 1000 NO (or YES) queries with a test distance of approximately $(1 - 4^{-\ell})$ (or $(1 + 4^{-\ell})$) times the true Fréchet distance under translation, as well as the interval between the lower and upper quartile over the queries.

For our evaluation, we focus on the CHARACTERS data set which allows us to distinguish the rough shape of the curves: We subdivide the curve set into the subsets C_α for $\alpha \in \Sigma$ (where Σ is the set of 20 characters occurring in CHARACTERS). In particular for each character pair $\alpha, \beta \in \Sigma$, we create a sample of N_{samples} curve pairs (π, σ) chosen uniformly at random from $C_\alpha \times C_\beta$. For $N_{\text{samples}} = 5$, computing the value (up to $\epsilon = 10^{-7}$) for all $N_{\text{samples}} \cdot (\binom{|\Sigma|}{2} + |\Sigma|) = 1050$ sampled curve pairs gives the statistics shown in Table 2.

Since already for this example the Lipschitz-only approach is dominated by almost a factor of 30 by LMF (and by a factor of almost 8 by binary search), we perform more detailed analyses with $N_{\text{samples}} = 100$ only for LMF and binary search. The overall performance is given in Table 3. Also here LMF is more than 3 times faster than binary search. To give more insights into the relationship of their running times, we give a scatter plot of the running times of LMF and binary search on the same instances over the complete benchmark in Figure 6, showing that binary search generally outperforms LMF only on instances which are comparably easy for LMF as well. The advantage of LMF becomes particularly clear on hard instances.

We give further experiments in the full version of the paper.

Table 2 Statistics for approximate value computation for $N_{\text{samples}} = 5$. In parentheses we show the mean values averaged over a total of 1050 instances.

Approach	Time	Black-Box Calls
LMF	148,032 ms (141.0 ms per instance)	13,323,232 (12,688.8 per instance)
Binary Search	536,853 ms (511.3 ms per instance)	45,909,628 (43,723.5 per instance)
Lipschitz-only	4,204,521 ms (4,004.3 ms per instance)	820,468,224 (781,398.3 per instance)

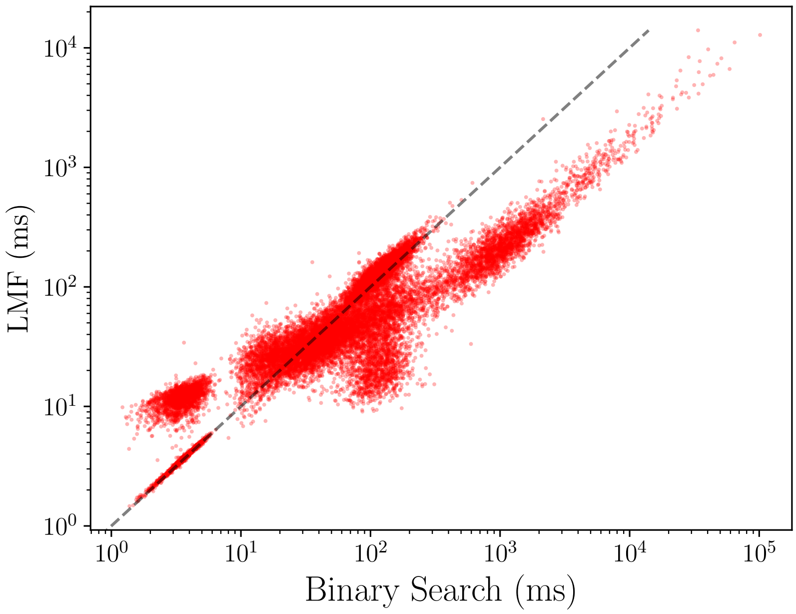


Figure 6 Running times of LMF and binary search on set of randomly sampled CHARACTERS curves.

Table 3 Statistics for approximate value computation for $N_{\text{samples}} = 100$. In parentheses, we give average values over the total of 21,000 curve pairs.

Algorithm	Time		Black-Box Calls
LMF	2,938,512 ms (140.0 ms per instance)		260,128,449 (12,387.1 per instance)
	- Preprocessing	71,728 ms	
	- Black-box calls (Lipschitz)	400,189 ms	
	- Arrangement estimation	166,479 ms	
	- Arrangement algorithm	2,250,493 ms	
	* Construction	1,537,500 ms	
	* Black-box calls	545,442 ms	
Binary Search	10,555,630 ms (502.7 ms per instance)		875,424,988 (41,686.9 per instance)

7 Conclusion

We engineer the first practical implementation for the discrete Fréchet distance under translation in the plane. While previous algorithmic solution for the problem solve it via expensive discrete methods, we introduce a new method from continuous optimization to achieve significant speed-ups on realistic inputs. This is analogous to the success of integer programming solvers which, while optimizing a discrete problem, choose to work over the reals to gain access to linear programming relaxations, cutting planes methods, and more. A novelty here is that we successfully apply such methods to obtain drastic speed-ups for a *polynomial-time problem*.

We leave as open problems to determine whether there are reasonable analogues of further ideas from integer programming, such as cutting plane methods or preconditioning, that could help to get further improved algorithms for our problem. More generally, we believe that this gives an exciting direction for algorithm engineering in general that should find wider applications. A particular direction in this vein is the use of our methods to compute rotation- or scaling-invariant versions of the Fréchet distance. Intuitively, by introducing additional dimensions in our search space, our methods can in principle also be used to optimize over such additional degrees of freedom. However, the Lipschitz condition changes significantly, and it remains subject of future work to determine the applicability of our methods in these settings.

References

1. ACM SIGSPATIAL GIS Cup 2017 Data Set. `https://www.martinwerner.de/datasets/san-francisco-shortest-path.html`. Accessed: 2018-12-03.
2. Character Trajectories Data Set. `https://archive.ics.uci.edu/ml/datasets/Character+Trajectories`. Accessed: 2018-12-03.
3. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014. `doi:10.1137/130920526`.
4. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1–2):78–99, 1995.
5. Helmut Alt, Christian Knauer, and Carola Wenk. Matching polygonal curves with respect to the Fréchet distance. In *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'01)*, pages 63–74, 2001.
6. Julian Baldus and Karl Bringmann. A fast implementation of near neighbors queries for Fréchet distance (GIS Cup). In *Proc. of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2017)*, pages 99:1–99:4. ACM, 2017. `doi:10.1145/3139958.3140062`.
7. Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. A faster algorithm for the discrete Fréchet distance under translation. *ArXiv preprint*, 2015. `arXiv:1501.03724`.
8. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Ann. IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 661–670, 2014.
9. Karl Bringmann, Marvin Künnemann, and André Nusser. Fréchet distance under translation: Conditional hardness and an algorithm via offline dynamic grid reachability. In *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 2902–2921, 2019. `doi:10.1137/1.9781611975482.180`.
10. Karl Bringmann, Marvin Künnemann, and André Nusser. Walking the dog fast in practice: Algorithm engineering of the Fréchet distance. In *Proc. 35th International Symposium on Computational Geometry (SoCG 2019)*, pages 17:1–17:21, 2019. `doi:10.4230/LIPIcs.SoCG.2019.17`.

11 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four Soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.

12 Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. 20th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA'09)*, pages 645–654, 2009.

13 Kevin Buchin, Yago Diez, Tom van Diggelen, and Wouter Meulemans. Efficient trajectory queries under the Fréchet distance (GIS Cup). In *Proc. of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2017)*, pages 101:1–101:4. ACM, 2017. doi:10.1145/3139958.3140064.

14 Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based clustering of trajectories. In *Proc. of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2019)*, pages 496–499. ACM, 2019. doi:10.1145/3347146.3359111.

15 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2887–2901. SIAM, 2019. doi:10.1137/1.9781611975482.179.

16 Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. FRESH: Fréchet similarity with hashing. In *Proc. of the 16th International Symposium on Algorithms and Data Structures (WADS 2019)*, volume 11646 of *LNCS*, pages 254–268. Springer, 2019. doi:10.1007/978-3-030-24766-9_19.

17 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, July 2012. doi:10.1007/s00454-012-9402-z.

18 Fabian Dütsch and Jan Vahrenhold. A filter-and-refinement-algorithm for range queries based on the Fréchet distance (GIS Cup). In *Proc. of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2017)*, pages 100:1–100:4. ACM, 2017. doi:10.1145/3139958.3140063.

19 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

20 Efim A. Galperin. The cubic algorithm. *Journal of Mathematical Analysis and Applications*, 112(2):635–640, 1985. doi:10.1016/0022-247X(85)90268-9.

21 Efim A. Galperin. Two alternatives for the cubic algorithm. *Journal of Mathematical Analysis and Applications*, 126(1):229–237, 1987. doi:10.1016/0022-247X(87)90088-6.

22 Efim A. Galperin. Precision, complexity, and computational schemes of the cubic algorithm. *Journal of Optimization Theory and Applications*, 57(2):223–238, 1988. doi:10.1007/BF00938537.

23 Efim A. Galperin. The fast cubic algorithm. *Computers & Mathematics with Applications*, 25(10):147–160, 1993. doi:10.1016/0898-1221(93)90289-8.

24 E. Gourdin, P. Hansen, and B. Jaumard. Global optimization of multivariate lipschitz functions: Survey and computational comparison, 1994.

25 Pierre Hansen and Brigitte Jaumard. Lipschitz optimization. In Reiner Horst and Panos M. Pardalos, editors, *Handbook of Global Optimization*, pages 407–493. Springer US, Boston, MA, 1995. doi:10.1007/978-1-4615-2025-2_9.

26 Reiner Horst. A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Applications*, 51:271–291, 1986. doi:10.1007/BF00939825.

27 Reiner Horst and Hoang Tuy. On the convergence of global methods in multiextremal optimization. *Journal of Optimization Theory and Applications*, 54:253–271, 1987. doi:10.1007/BF00939434.

28 Reiner Horst and Hoang Tuy. *Global Optimization – Deterministic Approaches*. Springer Berlin Heidelberg, 3rd edition, 1996.

29 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure–structure alignment with discrete Fréchet distance. *J. Bioinformatics and Computational Biology*, 6(01):51–64, 2008.

30 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983. `doi:10.1145/2157.322410`.

31 Axel Mosig and Michael Clausen. Approximately matching polygonal curves with respect to the fréchet distance. *Computational Geometry*, 30(2):113–127, 2005. Special Issue on the 19th European Workshop on Computational Geometry. `doi:10.1016/j.comgeo.2004.05.004`.

32 S.A. Piyavskii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972. `doi:10.1016/0041-5553(72)90115-2`.

33 Ron Wein, Eric Berberich, Efi Fogel, Dan Halperin, Michael Hemmer, Oren Salzman, and Baruch Zukerman. 2D arrangements. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.2 edition, 2020. URL: `https://doc.cgal.org/5.0.2/Manual/packages.html#PkgArrangementOnSurface2`.

34 Carola Wenk. *Shape matching in higher dimensions*. PhD thesis, Freie Universität Berlin, 2002. PhD Thesis.

35 Martin Werner and Dev Oliver. ACM SIGSPATIAL GIS cup 2017: Range queries under Fréchet distance. *SIGSPATIAL Special*, 10(1):24–27, 2018.

36 L.A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.