The k-Fréchet distance of polygonal curves

Maike Buchin[1] and Leonie Ryvkin[2]

1 **Faculty of Computer Science, TU Dortmund,** `maike.buchin@tu-dortmund.de`

2 **Department of Mathematics, Ruhr University Bochum,** `leonie.ryvkin@rub.de`

Abstract

We introduce a new distance measure for comparing polygonal chains: the k-Fréchet distance. As the name implies it is closely related to the well-studied Fréchet distance but allows to find similarities between curves that resemble each other only piecewise. As we will explain it provides a nice transition between (weak) Fréchet distance and Hausdorff distance. We prove NP-completeness for the k-Fréchet distance of polygonal curves in different variants and furthermore APX-completeness for the optimization version of our problem, discuss algorithmic approaches and present open questions.

1 Introduction

During the past decades several methods for comparing geometrical shapes have been studied in a variety of applications, for example analysing geographic data, such as trajectories, or comparing chemical structures, e.g. protein chains or human DNA.

The Fréchet distance has been well-studied in the past decades since it has proven to be very helpful in applications such as the above mentioned geographic data analysis or computer aided design. The Hausdorff distance, another similarity measure, has also proven to be useful in applications and can be computed more efficiently than the Fréchet distance. However, it provides us with less information by taking only the overall shape of curves into consideration, not how they are traversed.

We introduce the k-Fréchet distance as a distance measure in between Hausdorff and (weak) Fréchet distance. The k-Fréchet distance allows to compare shapes consisting of several parts by cutting a curve into a number of subcurves where the subcurves resemble each other in terms of the (weak) Fréchet distance. Therefore it allows to find similarities between objects of rearranged pieces such as chemical structures or handwritten characters and symbols.

Characterizing these distance measures in the free space shows that the k-Fréchet distance bridges between the (weak) Fréchet distance and Hausdorff distance (see below for details): the weak Fréchet distance can be characterized by one component in the free space projecting surjectively onto both parameter spaces, and the Hausdorff distance can be characterized by all components of the free space projecting surjectively onto both parameters. For the k-Fréchet distance we ask that k components of the free space project surjectively onto both parameter spaces.

This paper is organized as follows: first we recall some necessary definitions and formally define the k-Fréchet distance. In Section 3 we prove NP- and APX-completeness of the k-Fréchet problem. In Section 4 we present an approximation algorithm of factor 2 as well as an exact algorithm which runs in polynomial time for fixed k. We conclude the paper with open questions.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Definitions

Recall the Fréchet distance [1], a well-known measure for curves, which is defined as follows: For curves $P, Q\colon [0,1] \to [0,1]$ the Fréchet distance is given by

$$\delta_F(P,Q) = \inf_{\sigma} \max_{t\in[0,1]} \|P(t) - Q(\sigma(t))\|,$$

where the reparametrisations $\sigma\colon [0,1] \to [0,1]$ range over all orientation-preserving homeomorphisms. A variant is the weak Fréchet distance δ_{wF}, where both curves are reparameterised by σ and τ, respectively, which range over all continuous surjective functions.

A well-known characterisation which is key to efficient algorithms for computing both weak and (strong) Fréchet distance [1] uses the free space diagram. First we recall the free space F_ε:

$$F_\varepsilon(P,Q) = \{(t_1,t_2) \in [0,1]^2\colon \|P(t_1) - Q(t_2)\| \leqslant \varepsilon\}.$$

The free space diagram puts this information into a $(n \times m)$-grid, where n and m are the number of segments in P and Q, respectively.

The Fréchet distance of two curves is at most a given value ε if there exists a monotone path through the free space connecting the bottom left to the top right corner. For the weak Fréchet distance to equal at most ε such a path need not be monotone. The Hausdorff distance δ_H can be characterised as the free space projecting surjectively onto both parameter spaces.

We define the k-Fréchet distance δ_{kF} as in between Hausdorff and weak Fréchet distance using only a constant number k of components in the free space to project onto both parameter spaces:

$$\delta_{kF}(P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where now $\sigma, \tau\colon [0,1] \to [0,1]$ range over all surjective functions which are piecewise defined, allowing at most $k-1$ jump discontinouities, such that the images of the continuous parts partition the curve. That is, we cut the curves P and Q into at most k pieces or subcurves such that two resembling subcurves have small weak Fréchet distance. For the decision version of the problem, we ask whether the weak Fréchet distance between pieces can be bounded a given value ε (note that k is a fixed upper bound of cuts here). Naturally, for a fixed distance ε, we would like to cut the curves into as few subcurves as possible (optimization version).

We will also consider the variant where we use (strong) Fréchet distance instead of weak Fréchet distance for the subcurves, that is the reparameterizations σ, τ have to be piecewise homeomorphisms. However, as the weak Fréchet distance (arguably) results in the more natural definition for the k-Fréchet distance we use it as the standard variant.

By definition k-Fréchet distance lies in between Hausdorff and (weak) Fréchet distance

$$\delta_H(P,Q) \leq \delta_{kF}(P,Q) \leq \delta_F(P,Q).$$

Also, the k-Fréchet distance decreases as k increases, and for $k = 1$ it equals weak Fréchet distance, whereas for $k \geq n^2$ it equals Hausdorff distance.

3 NP-Completeness

▶ **Theorem 3.1.** *Deciding wether $\delta_{kF}(P,Q) \leq \varepsilon$ for fixed ε and k is NP-complete. Finding the minimum number of cuts (or components) k is NP-complete as well.*

Proof. First note that we can easily verify a given solution to our problem: we compute the free space diagram of P and Q and identify the free space components resembling mappings of the respective subcurves. Next we need to check whether the union of these components projects surjectively onto both parameter spaces. This can be done in polynomial time, therefore we have k-Fréchet $\in$ NP.

To show NP-hardness of our problem we reduce from the Minimum Common String Partition (MCSP)-Problem. Let us first recall the MCSP-Problem [7]: Given two strings A and B we want to partition them into substrings such that $A = A_1 A_2 \dots A_n$ and $B = B_1 B_2 \dots B_n$ where for all $i = 1, \dots, n$ holds that $A_i = B_j$ for some $j \in \{1, \dots, n\}$.

There are several variants of MCSP, such as k-MCSP where each letter occurs at most k times in each string (which was proven to be an NP-hard optimization problem in [5]) or MCSP^c where there are at most c elements in the alphabet of the strings (an NP-hard problem even for the decision version, presented in [4]). Both 2-MCSP and MCSP^2 have been proven to be NP-hard (therefore MCSP is NP-hard as well), the first variant is even APX-hard ([5], [7]).

Now, for an instance of MCSP we construct curves P and Q as follows: first we subdivide the unit interval into c intervals of equal size, where c denotes the number of different letters in both strings (which is bounded since we consider finite strings). We then identify every letter of the alphabet Σ with a number in $\{1, \dots, c\}$. Next we transform string A into a curve P and string B into curve Q: for every letter l in a string we form a polygonal chain consisting of five segments: the first connecting $(\frac{1}{2}, 0)$ to $(\frac{1}{2}, \frac{l}{c})$ vertically, a second connecting $(\frac{1}{2}, \frac{l}{c})$ to $(0, \frac{l}{c})$ horizontally, followed by $(0, \frac{l}{c})$ to $(1, \frac{l}{c})$, back to $(\frac{1}{2}, \frac{l}{c})$ and finally back to $(0, \frac{l}{c})$. The result is a T-shape, as shown in Figure 1.

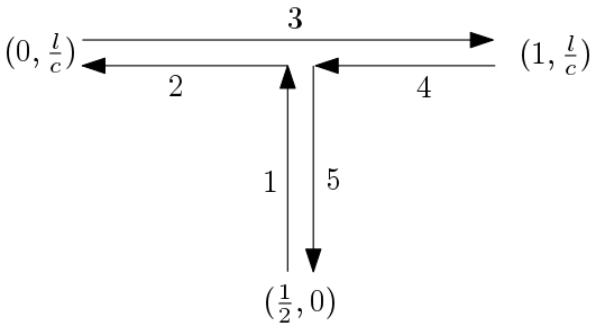

Figure 1 A T-shaped curve fragment resembling the letter l in a string

Since beginning and ending of each T-shaped curve fragment is the same, the fragments can be concatenated. In this manner we produce curves consisting of as many T-shapes as there are letters in the strings, both curves lying on top of each other. We can choose ε to be any number smaller than $1/c$, e.g. $1/2c$.

The T-shapes only differ by length of their vertical line segments, and each length corresponds to a certain letter of the respective string. By choosing $\varepsilon < 1/c$, T-shapes resembling different letters cannot be matched. Hence our construction leads to the following result for all pairs of subcurves:

$$\forall P_i, Q_j \colon P_i = Q_j \Leftrightarrow \delta_{wF}(P_i, Q_i) \leqslant \varepsilon.$$

In fact we even have $\delta_{wF}(P_i, Q_i) = 0 \leqslant \varepsilon$, since we place Q on top of P and construct identical concatenations of T-shaped subcurves for equal substrings. Because all letters start

in $(\frac{1}{2}, 0)$, we get that two substrings are equal iff the respective subcurves have weak Fréchet distance less than ε.

In the free space the following picture arises: we get components in the free space of P and Q resembling equal substrings of A and B. Hence, the longer identical substrings, the larger their resembling component in the free space. Therefore finding a minimum collection of k components projecting surjectively onto the parameter spaces equals finding a minimum number of blocks where each block of A equals a block in B. See Figure 2 for an example.

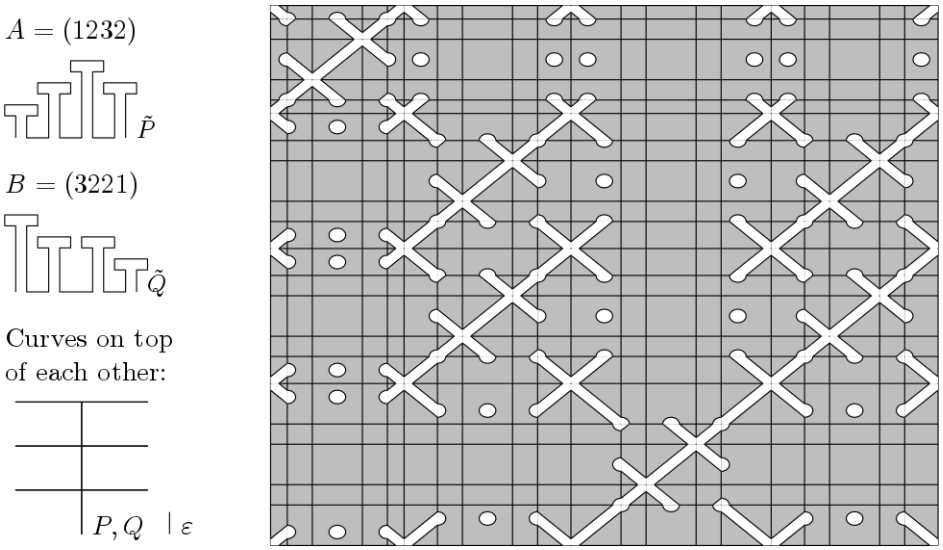

Figure 2 The curves and resulting free space diagram corresponding to strings A and B.

Note that the antenna-like symbol in the bottom left corner consists of the actual curves P and Q that resemble the input strings A and B. To make the curves more visible we added illustrations $\tilde{P}$, $\tilde{Q}$ above.

For the decision problem we need to identify a collection of components of size at most k, which is a constant and part of the input in this case, that projects surjectively onto the parameter spaces. As discussed this corresponds to the problem of finding a common string partition consisting of at most k substrings per string. The latter has been proven to be NP-hard [7] and the result transfers to our problem.

When we use (strong) Fréchet distance instead of weak Fréchet distance for the subcurves, our reduction still works. In fact, even a simpler construction suffices for this variant: we construct the curves as before but omit the horizontal line segments, i.e., we only get (one-dimensional) "I"-shaped subcurves. Again, we get that two letters are equal iff the corresponding subcurves have Fréchet distance 0. Observe that this simplified reduction does not work for the weak Fréchet distance which allows to backtrack. Therefore we use the T-shape for the weak Fréchet, which makes backtracking ineffective, in the sense that it is impossible to map a subcurve of P representing a letter of A to a subcurve of Q representing a letter of B with reversed orientation. ◀

▶ Remark. Our reduction from MCSP also works for the variants MCSP^c and k-MCSP, which translate to curves with at most c different heights of T-shapes (so basically the same curves as above) and curves with no more than k identical copies of T-shapes of each height.

▶ **Theorem 3.2.** *The optimization version of the k-Fréchet-problem is APX-complete.*

Proof. The reduction above provides a one-to-one relationship between MCSP and k-Fréchet, as any selection of components can be directly translated into common substrings. Therefore a selection of components (say, of quality $k + c$ with c being a constant and k the size of an optimal solution) corresponds to a division of the input strings into the exact same number of substrings, thus proving that our reduction is a strict AP-reduction. Since k-MCSP is APX-hard (in the optimization version), so is the k-Fréchet-Problem. APX-completeness follows from the existence of a 2-approximation algorithm which we present in the Section 4. ◀

▶ **Lemma 3.3.** *For curves in one dimension, that is $P, Q\colon [0,1] \to \mathbb{R}$, k-Fréchet distance equals Hausdorff and weak Fréchet distance, whereas strong k-Fréchet distance remains NP-complete.*

Proof. Equality of k-Fréchet distance, Hausdorff and weak Fréchet distance can be shown using the Mountain Climbing Theorem, which states that it is possible for two climbers to proceed from foot to top of a mountain while always remaining on equal height. Of course, the climbers have to start at the same time but on different sides of the same mountain [2], [6]. For the strong Fréchet distance as underlying distance measure the simplified reduction mentioned above (I-shapes) shows NP-hardness for one-dimensional curves. ◀

4 Algorithmic approaches

First, we observe that a brute force approach results in a runtime exponential in k, and then present an efficient 2-approximation algorithm (regarding the size of the found selection of components) using a greedy approach.

▶ Remark. The k-Fréchet distance can be computed in $\mathcal{O}(k \cdot n^{2k})$ time.

The brute force approach simply checks for all selections of k components of the free space whether their joint projections cover both parameter spaces. That means we have to check at most $\binom{n^2}{k}$ possible combinations of components, which results in a runtime of $\mathcal{O}(k \cdot n^{2k})$ for fixed k which is only feasible for very small k.

To approximately decide the k-Fréchet distance, we greedily choose a set of components that cover both parameter spaces.

For this we compute the free space F_ε, project its components onto the two parameter spaces and interpret the projections as intervals. We store the intervals in sorted lists. For each parameter space we then use a scan to greedily select the smallest number of intervals that cover it. The worst case that might result is the following: the intervals we select correspond to different components in the free space for the two parameter spaces, so that the union of our selections is of size $s_1 + s_2$ where s_1 and s_2 are the number of selected components for the respective parameter spaces. A different selection of size $s = \max(s_1, s_2)$ might cover both parameter spaces but is not detected by the greedy scan. So the size of our selection indicates whether $\delta_{kF} \leqslant \varepsilon$: if it is smaller or equal to k the answer is positive, i.e. $\delta_{kF} \leqslant \varepsilon$, and if it is larger than $2k$ the answer is definitely negative. For any number of selected components in between k and $2k$ we cannot rule out that there might be a smaller selection of at most k components that covers the parameter spaces.

Computing the free space takes quadratic time. Sorting the lists adds another logarithmic factor while the scan takes linear time in the number of intervals. All in all we get a runtime of $\mathcal{O}(n^2 \log n)$.

▶ **Theorem 4.1.** *The algorithm described above runs in $\mathcal{O}(n^2 \log n)$ time and finds a selection of components that covers both parameter spaces iff one exists. A found selection contains at worst twice the minimum number of components needed.*

We conjecture that the approximation factor 2 is probably not tight, as we were not able to construct an example where it is. Showing that it is or that a better approximation factor holds remains open.

5 Conclusion and further work

We introduced a new distance measure, the k-Fréchet distance, which lies in between Hausdorff and (weak) Fréchet distance. We showed NP-hardness of deciding two variants of this distance between polygonal curves. Finding the minimum number of subcurves is even APX-hard. Furthermore, we presented a polynomial time algorithm that works for small fixed k as well as an efficient 2-approximation algorithm. We close this section with some open questions.

Since the MCSP-Problem has been studied intensively, we hope to transfer more of the interesting results to our k-Fréchet-Problem: It was shown in [3] that MCSP is fixed-parameter tractable, therefore we plan to investigate whether our problem is as well. We defined the k-Fréchet distance as subdivision of curves where we match subcurves in terms of the weak Fréchet distance, but it would be interesting to just reparameterize the curves allowing jump discontinuities. As a result it is possible to match a subcurve of one to several subcurves of the other original curve. Our reduction does not work in that case and we believe it is an even harder problem. Furthermore it would be interesting to see if there are special cases for our problem to which we can find algorithms with polynomial runtimes since the curves we construct are neither monotone, nor κ-straight, κ-bounded or c-packed. Finally it remains open to improve the approximation factor achieved by our greedy approach.

Acknowledgments. We are grateful to Simon Pflips for proofreading and interesting, helpful discussions.

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995.

2 Kevin Buchin, Maike Buchin, Christian Knauer, Günther Rote, and Carola Wenk. How difficult is it to walk the dog? In *Proc. 23rd European Workshop on Computational Geometry*, pages 170–173, 2007.

3 Laurent Bulteau and Christian Komusiewicz. Minimum common string partition parameterized by partition size is fixed-parameter tractable. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 102–121. ACM, New York, 2014.

4 Peter Damaschke. Minimum common string partition parameterized. In *Algorithms in bioinformatics*, volume 5251 of *Lecture Notes in Comput. Sci.*, pages 87–98. Springer, Berlin, 2008.

5 Avraham Goldstein, Petr Kolman, and Jie Zheng. Minimum common string partition problem: hardness and approximations. *Electron. J. Combin.*, 12:Research Paper 50, 18, 2005.

6 Jacob E. Goodman, János Pach, and Chee-K. Yap. Mountain climbing, ladder moving, and the ring-width of a polygon. *Amer. Math. Monthly*, 96(6):494–510, 1989.

7 Haitao Jiang, Binhai Zhu, Daming Zhu, and Hong Zhu. Minimum common string partition revisited. *J. Comb. Optim.*, 23(4):519–527, 2012.

8 Leonie Lange. Algorithms for comparing monotone curves and terrains. Master's thesis, Ruhr-Universität Bochum, Faculty of Mathematics, 2017.