Exact Mean Computation in Dynamic Time Warping Spaces

Markus Brill[1], Till Fluschnik[1], Vincent Froese[1], Brijnesh Jain[2], Rolf Niedermeier[1], and David Schultz[2]

[1]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany,
{brill,till.fluschnik,vincent.froese,rolf.niedermeier}@tu-berlin.de
[2]Distributed Artificial Intelligence Laboratory, TU Berlin, Germany,
{Brijnesh-Johannes.Jain,David.Schultz}@dai-labor.de

Abstract

Dynamic time warping constitutes a major tool for analyzing time series. In particular, computing a mean series of a given sample of series in dynamic time warping spaces (by minimizing the Fréchet function) is a challenging computational problem, so far solved by several heuristic, inexact strategies. We spot several inaccuracies in the literature on exact mean computation in dynamic time warping spaces. Our contributions comprise an exact dynamic program computing a mean (useful for benchmarking and evaluating known heuristics). Empirical evaluations reveal significant deficits of the state-of-the-art heuristics in terms of their output quality. Finally, we give an exact polynomial-time algorithm for the special case of binary time series.

Keywords: time series analysis, Fréchet function, exact dynamic programming, accuracy of heuristics

1 Introduction

Time series such as acoustic signals, electrocardiograms, and internet traffic data are time-dependent observations that vary in length and temporal dynamics. Given a sample of time series, to filter out the corresponding variations, one major direction to time series averaging applies *dynamic time warping* (dtw). Different dtw-averaging approaches have been applied to improve nearest neighbor classifiers and to formulate centroid-based clustering algorithms in dtw-spaces [1, 12, 14, 17, 21, 22, 26].

The most successful approaches pose time series averaging as an optimization problem [7, 12, 20, 23, 26]: Suppose that $\mathcal{X} = \left(x^{(1)}, \ldots, x^{(k)}\right)$ is a sample of k time series $x^{(i)}$. Then a (Fréchet) mean in dtw-spaces is any time series z that

minimizes the Fréchet function [8]

$$F(z) = \frac{1}{k} \sum_{i=1}^{k} \left(\mathrm{dtw}\left(z, x^{(i)}\right) \right)^2,$$

where $\mathrm{dtw}(x, y)$ denotes the dtw-distance between time series x and y. We refer to the problem of minimizing $F(z)$ for z taken from the set $\mathcal{T}$ of all time series of finite length as the DTW-Mean problem. A variant of the DTW-Mean problem constrains the solution set $\mathcal{T}$ to the subset $\mathcal{T}_m \subseteq \mathcal{T}$ of time series of length m. For both the constrained and unconstrained DTW-Mean problem, solutions are guaranteed to exist, but are not unique in general [23].

Polynomial-time algorithms for solving the DTW-Mean problem are unknown. NP-hardness as claimed in the literature [2, 12, 18–21] is based on wrong premises. Algorithms with exponential time complexity have been falsely claimed to provide optimal solutions [12, 19, 20]. Existing heuristics approximately solve the constrained DTW-Mean problem [7, 20, 23]. Thereby, the length m of feasible solutions is specified beforehand without any knowledge about whether the subset $\mathcal{T}_m$ contains an optimal solution of the unconstrained DTW-Mean problem. In summary, both the computational complexity of DTW-Mean and the development of nontrivial exact algorithms are to be considered widely open.

Our contributions. We discuss several problematic statements in the literature concerning the computational complexity of exact algorithms for DTW-Mean. We refute (supplying counterexamples) some false claims from the literature and clarify the known state of the art with respect to computing means in dtw-spaces (Section 3). We develop a dynamic program as an exact algorithm for the unconstrained DTW-Mean problem with rational time series. The time complexity of the proposed dynamic program is $O(n^{2k+1} \cdot k2^k)$, where k is the sample size and n is the maximum length of a sample time series (Section 5). We apply the proposed exact dynamic program on small-scaled problems as a benchmark of how well the state-of-the-art heuristics approximate a mean. Our empirical findings reveal that all state-of-the-art heuristics suffer from relatively poor worst-case solution quality in terms of minimizing the Fréchet function, and the solution quality in general may vary quite a lot (Section 6). Moreover, a further theoretical contribution is to show that in case of binary time series (both input and mean) there is an exact polynomial-time algorithm for mean computation in dtw-spaces (Section 4).

2 Preliminaries

Throughout this paper, we consider only finite time series with rational elements. A univariate *time series* of length n is a sequence $x = (x_1, \ldots, x_n) \in \mathbb{Q}^n$. We denote the set of all univariate rational time series of length n by $\mathcal{T}_n$. Furthermore, $\mathcal{T} = \bigcup_{n \in \mathbb{N}} \mathcal{T}_n$ denotes the set of all univariate rational time series of finite length. For every $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$.

The next definition is fundamental for our central computational problem.

Definition 1. *A* warping path of order $m \times n$ *is a sequence* $p = (p_1, \ldots, p_L)$ *with* $p_i \in [m] \times [n]$ *for all* $i \in [L]$ *such that*
i) $p_1 = (1, 1)$,
ii) $p_L = (m, n)$, *and*
iii) $p_{j+1} - p_j \in \{(1, 0), (0, 1), (1, 1)\}$ *for all* $j \in [L - 1]$.

Note that $\max\{m, n\} \leq L \leq m + n$. We denote the set of all warping paths of order $m \times n$ by $\mathcal{P}_{m,n}$. For two time series $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_n)$, a warping path $p = (p_1, \ldots, p_L) \in \mathcal{P}_{m,n}$ defines an alignment of x and y: each pair $p_\ell = (i_\ell, j_\ell)$ of p *aligns* element x_{i_ℓ} with y_{j_ℓ}. The *cost* $C_p(x, y)$ for aligning x and y along warping path p is defined as $C_p(x, y) = \sum_{\ell=1}^{L} (x_{i_\ell} - y_{j_\ell})^2$. The *dtw-distance* between x and y is defined as

$$\mathrm{dtw}(x, y) := \min_{p \in \mathcal{P}_{m,n}} \left\{ \sqrt{C_p(x, y)} \right\}.$$

A warping path p with $C_p(x, y) = (\mathrm{dtw}(x, y))^2$ is called an *optimal* warping path for x and y. We remark that $\mathrm{dtw}(x, y)$ for two time series x, y of length n can be computed in subquadratic time $O(n^2 \log\log\log n / \log\log n)$ [9]. The existence of a strongly subquadratic-time (that is, $O(n^{2-\epsilon})$ for some $\epsilon > 0$) algorithm is considered unlikely [5].

Our central problem DTW-Mean is defined as follows.

DTW-Mean
Input: Sample $\mathcal{X} = \left(x^{(1)}, \ldots, x^{(k)}\right)$ of k univariate rational time series.
Task: Find a univariate rational time series z that minimizes the Fréchet function $F(z)$.

For the prescribed setup, it is known that a mean always exists [13, Proposition 2.10] and that a mean of rational time series is also rational [23, Theorem 3.3].

3 Problematic Statements in the Literature

In this section we discuss wrong claims and errors present in the literature.

3.1 NP-hardness

The DTW-Mean problem is often related to the Steiner String (STS) problem [2, 12, 18–21]. A *Steiner string* [11] (or *Steiner sequence*) for a set S of strings is a string t that minimizes $\sum_{s \in S} D(t, s)$, where D is a distance measure between two strings often assumed to fulfill the triangle inequality (e.g., the weighted edit distance). Computing a Steiner string is equivalent to solving the Multiple Sequence Alignment (MSA) problem [11]. Both, STS and MSA, are known to be NP-hard for several distance measures even on binary alphabets [4, 15, 27].

Several papers mention the NP-hardness results for MSA and STS in the context of DTW-Mean [12, 18, 21], thereby suggesting that DTW-Mean is also NP-hard. However, it is not clear (and not shown in the literature) how to reduce from MSA (or STS) to DTW-Mean since the involved distance measures are significantly different. For example, the dtw-distance lacks the metric property of the edit distance. We are not aware of any formal proof of NP-hardness for DTW-Mean. Interestingly, as we show in Section 4, DTW-Mean is solvable in polynomial time for binary time series.

3.2 Computation of Exact Solutions

Two exponential-time algorithms were proposed to exactly solve DTW-Mean. The first approach is based on the idea of multiple sequence alignment in bioinformatics [11] and the second is a brute-force method [20]. We show that neither algorithm is guaranteed to return an optimal solution for every DTW-Mean instance.

Multiple Alignment. It is claimed that DTW-Mean can be solved by averaging a (global) multiple alignment of the k input series [12, 19–21]. A multiple alignment of k time series in the context of dynamic time warping is described as computing a k-dimensional warping path in a k-dimensional matrix. Concerning the running time, it is claimed that computing a multiple alignment requires $\Theta(n^k)$ time [19, 20] ($O(n^k)$ time [21]), where n is the maximum length of an input time series. Neither the upper bound of $O(n^k)$ nor the lower bound of $\Omega(n^k)$ on the running time are formally proven. Given a multiple alignment, it is claimed that averaging the k resulting aligned time series column-wise yields a mean [19, Definition 4].

We show that this is not correct even for two time series. Note that for two time series, the multiple alignment is obtained by an optimal warping path. However, the column-wise average of two aligned time series obtained from an optimal warping path is not always an optimal solution for a DTW-Mean instance as the following example shows.

Example 1. Let $x^{(1)} = (1, 4, 2, 3)$ and $x^{(2)} = (4, 2, 4, 5)$. Using a computer, we obtained an optimal warping path $p = ((1, 1), (2, 1), (3, 2), (4, 3), (4, 4))$. The corresponding aligned time series are

$$x_p^{(1)} = (1, 4, 2, 3, 3),$$
$$x_p^{(2)} = (4, 4, 2, 4, 5).$$

The arithmetic mean of these time series is $\bar{x} = (2.5, 4, 2, 3.5, 4)$. However, for $z = (2.5, 4, 2, 4)$, we have $F(z) = 6.5 < 7 = F(\bar{x})$, which shows that $\bar{x}$ is not a mean (see Figure 1).

In the above example, the time series $\bar{x}$ is also not an optimal choice among all time series of length 5 since also $z' = (2.5, 4, 4, 2, 4)$ satisfies $F(z') = 6.5 < F(\bar{x})$. In fact, by computer-based exhaustive search we found that no warping

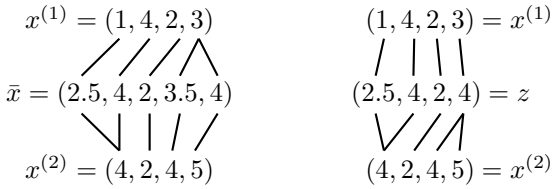


Figure 1: Illustration of Example 1. Shown are two time series $x^{(1)}$ and $x^{(2)}$, as well as the time series $\bar{x}$ obtained by the multiple alignment approach and a mean z. Lines indicate optimal alignments between the time series. Note that $\mathrm{dtw}(x^{(1)}, \bar{x})^2 = \mathrm{dtw}(x^{(2)}, \bar{x})^2 = 3.5$ whereas $\mathrm{dtw}(x^{(1)}, z)^2 = \mathrm{dtw}(x^{(2)}, z)^2 = 3.25$.

path $p \in \mathcal{P}_{4,4}$ yields a mean for $x^{(1)}$ and $x^{(2)}$ by averaging the aligned time series $x_p^{(1)}$ and $x_p^{(2)}$. We conclude that a multiple alignment as defined in [19, Definition 5] that shall produce an averaged time series that minimizes the Fréchet function does not exist in general. Example 1 implies that incremental pairwise averaging strategies such as NLAAF [10] or PSA [16] are based on a wrong mean computation for two time series.

We finish with another erroneous example from the literature [19, Figure 2].

Example 2. For the three time series

$$x^{(1)} = (1, 10, 0, 0, 4),$$
$$x^{(2)} = (0, 2, 10, 0, 0),$$
$$x^{(3)} = (0, 0, 0, 10, 0),$$

the multiple alignment is given as

$$x^{(1)'} = (1, 1, 1, 10, 0, 0, 4),$$
$$x^{(2)'} = (0, 0, 2, 10, 0, 0, 0),$$
$$x^{(3)'} = (0, 0, 0, 10, 0, 0, 0),$$

yielding the arithmetic mean $\bar{x} = (\frac{1}{3}, \frac{1}{3}, 1, 10, 0, 0, \frac{4}{3})$ with $F(\bar{x}) = 14/3 \geq 4.66$. However, for $z = (\frac{1}{4}, 1, 10, 0, \frac{4}{3})$, it holds $F(z) \leq 4.48 < F(\bar{x})$.

Brute-Force Algorithm. Another approach to solve DTW-Mean is based on a brute-force algorithm [20] working as follows: Suppose an optimal mean is of length m. Consider for each input time series a partition into m consecutive non-empty parts and align the i-th element in the mean with all elements in the i-th part of each time series. It is claimed that a mean can be found by trying out all possible partitions into m consecutive non-empty parts for each input time series.

The problem in this approach is that not all possible solutions are considered (as two elements in the mean can be aligned with the same element of an input time series). Example 1 depicts this problem.

4 Polynomial-time Solvability for Binary Data

By restricting the values in the time series (input and mean) to be binary (0 or 1), we arrive at the special case Binary DTW-Mean.

Showing that it suffices to search for a mean that is "condensed" (that is, no two consecutive values in it are the same) and that there always exists a mean of length at most $n + 1$,

Theorem 1. Binary DTW-Mean *for k input time series is solvable with $O(kn^3)$ arithmetic operations, where n is the maximum length of all input time series.*

To show polynomial-time solvability of Binary DTW-Mean, we first prove some preliminary results about the dtw-distance of binary time series and properties of a binary mean. We start with the following general definition.

Definition 2. *A time series $x = (x_1, \ldots, x_n)$ is* condensed *if no two consecutive elements are equal, that is, $x_i \neq x_{i+1}$ holds for all $i \in [n-1]$. We denote the* condensation *of a time series x by $\tilde{x}$ and define it to be the time series obtained by repeatedly removing one of two equal consecutive elements in x until the remaining series is condensed.*

The following proposition implies that a mean can always be assumed to be condensed. Note that this holds for arbitrary time series (not only for the binary case).

Proposition 1. *Let x be a time series and let $\tilde{x}$ denote its condensation. Then, for every time series y, it holds that* $\mathrm{dtw}(\tilde{x}, y) \leq \mathrm{dtw}(x, y)$.

Proof. Let y have length m and assume that $x = (x_1, \ldots, x_n)$ is not condensed. Then, $x_i = x_{i+1}$ holds for some $i \in [n-1]$. Let $p = ((i_1, j_1), \ldots, (i_L, j_L))$ be an optimal warping path for x and y. Now, consider the time series $x' = (x_1, \ldots, x_i, x_{i+2}, \ldots, x_n)$ that is obtained from x by deleting the element x_{i+1}. We construct a warping path p' for x' and y such that $C_{p'}(x', y) = C_p(x, y)$. To this end, let $p_a = (i_a, j_a)$, $2 \leq a \leq L$, be the first index pair in p where $i_a = i+1$ (hence, $i_{a-1} = i$). Now, we consider two cases.

If $j_a = j_{a-1} + 1$, then we define the order-$((n-1) \times m)$ warping path

$$p' := ((i_1, j_1), \ldots, (i_{a-1}, j_{a-1}), (i_a - 1, j_a), \ldots, (i_L - 1, j_L))$$

of length L. Thus, each element of y that was aligned to x_{i+1} in p is now aligned to x_i instead. To check that p' is a valid warping path, note first that $(i_1, j_1) = (1, 1)$ and $(i_L - 1, j_L) = (n - 1, m)$ holds since p is a warping path. Also, it holds

$$\begin{aligned} \forall 1 \leq \ell \leq a-2\colon (i_{\ell+1}, j_{\ell+1}) - (i_\ell, j_\ell) &\in \{(1,0), (0,1), (1,1)\}, \\ (i_a - 1, j_a) - (i_{a-1}, j_{a-1}) &= (0, 1), \\ \forall a \leq \ell \leq L-1\colon (i_{\ell+1} - 1, j_{\ell+1}) - (i_\ell - 1, j_\ell) &\in \{(1,0), (0,1), (1,1)\}. \end{aligned}$$

The cost of p' is

$$\begin{aligned} C_{p'}(x', y) &= \sum_{\ell=1}^{a-1} (x'_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a}^{L} (x'_{(i_\ell-1)} - y_{j_\ell})^2 \\ &= \sum_{\ell=1}^{a-1} (x_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a}^{L} (x_{i_\ell} - y_{j_\ell})^2 = C_p(x, y). \end{aligned}$$

Otherwise, if $j_a = j_{a-1}$, then we define the warping path

$$p' := ((i_1, j_1), \ldots, (i_{a-1}, j_{a-1}), (i_{a+1} - 1, j_{a+1}), \ldots, (i_L - 1, j_L))$$

of length $L-1$. Again, $(i_1, j_1) = (1, 1)$ and $(i_L - 1, j_L) = (n-1, m)$ holds since p is a warping path. Clearly, also

$$\begin{gathered} \forall 1 \leq \ell \leq a-2 \colon (i_{\ell+1}, j_{\ell+1}) - (i_\ell, j_\ell) \in \{(1,0), (0,1), (1,1)\} \text{ and} \\ \forall a+1 \leq \ell \leq L-1 \colon (i_{\ell+1} - 1, j_{\ell+1}) - (i_\ell - 1, j_\ell) \in \{(1,0), (0,1), (1,1)\} \end{gathered}$$

holds. Finally, we have $(i_{a+1} - 1, j_{a+1}) - (i_{a-1}, j_{a-1}) \in \{(1,0), (0,1), (1,1)\}$ since $i_{a+1} - 1 - i_{a-1} = i_{a+1} - i_a$ holds and also $j_{a+1} - j_{a-1} = j_{a+1} - j_a$ holds. Thus, p' is a valid warping path and its cost is

$$\begin{aligned} C_{p'}(x', y) &= \sum_{\ell=1}^{a-1} (x'_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a+1}^{L} (x'_{(i_\ell-1)} - y_{j_\ell})^2 \\ &= \sum_{\ell=1}^{a-1} (x_{i_\ell} - y_{j_\ell})^2 + \sum_{\ell=a+1}^{L} (x_{i_\ell} - y_{j_\ell})^2 \\ &= C_p(x, y) - (x_{i_a} - y_{j_a})^2. \end{aligned}$$

Since in both cases above, the cost does not increase, we obtain

$$\mathrm{dtw}(x', y) \leq C_{p'}(x', y) \leq C_p(x, y) = \mathrm{dtw}(x, y).$$

Repeating this argument until x' is condensed finishes the proof. □

Proposition 1 implies that we can assume a mean to be condensed. Next, we want to prove an upper bound on the length of a binary mean. To this end, we analyze the dtw-distances of binary time series. Note that a binary condensed time series is fully determined by its first element and its length. We use this property to give a closed expression for the dtw-distance of two condensed binary time series.

Lemma 1. *Let $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_m) \in \{0,1\}^*$ be two condensed binary time series with $n \geq m$. Then, it holds*

$$\mathrm{dtw}(x, y)^2 = \begin{cases} \lceil (n-m)/2 \rceil, & x_1 = y_1 \\ 2, & x_1 \neq y_1 \wedge n = m \\ 1 + \lfloor (n-m)/2 \rfloor, & x_1 \neq y_1 \wedge n > m \end{cases}.$$

Proof. We prove the statement by first giving a warping path that has the claimed cost and second proving that every warping path has at least the claimed cost.

"$\leq$": We show that there exists a warping path p between x and y that has the claimed cost. The warping path p is defined as follows:

If $x_1 = y_1$, then we have $x_i = y_i$ for all $i \in [m]$ (since x and y are condensed and binary) and we set

$$p := ((1,1),(2,2),\ldots,(m,m),(m+1,m),\ldots,(n,m)).$$

This warping path has cost $C_p(x,y) = \sum_{i=m+1}^{n} |x_i - y_m| = \lceil (n-m)/2 \rceil$.

If $x_1 \neq y_1$, then we have $x_i = y_{i-1}$ for all $2 \leq i \leq m$. Thus, for $n = m$, the warping path

$$p := ((1,1),(2,1),(3,2),\ldots,(n,m-1),(n,m))$$

has cost $C_p(x,y) = |x_1 - y_1| + |x_n - y_m| = 2$, Finally, for $n > m$, the warping path $p := ((1,1),(2,1),(3,2),\ldots,(m+1,m),(m+2,m),\ldots,(n,m))$ yields cost

$$\begin{aligned} C_p(x,y) &= |x_1 - y_1| + \sum_{i=2}^{m+1} |x_i - y_{i-1}| + \sum_{i=m+2}^{n} |x_i - y_m| \\ &= 1 + \sum_{i=m+2}^{n} |x_i - y_m| = 1 + \lceil (n-m-1)/2 \rceil = 1 + \lfloor (n-m)/2 \rfloor. \end{aligned}$$

"$\geq$": We show that every warping path has at least the cost claimed above. Consider an optimal warping path $p = (p_1,\ldots,p_L)$ for x and y and note that there are at least $n-m$ different indices $\ell \in [L-1]$ such that $p_{\ell+1} - p_\ell = (1,0)$ since $n \geq m$. For every such pairs $p_{\ell+1} = (i_{\ell+1}, j_{\ell+1})$, $p_\ell = (i_\ell, j_\ell)$ with $j_{\ell+1} = j_\ell$, we have

$$|x_{i_{\ell+1}} - y_{j_{\ell+1}}| + |x_{i_\ell} - y_{j_\ell}| = |x_{i_{\ell+1}} - y_{j_\ell}| + |x_{i_\ell} - y_{j_\ell}| = 1,$$

since $x_{j_{\ell+1}} \neq x_{j_\ell}$ (recall that x is condensed and binary). Hence, at least for every second such index ℓ (starting from the first one) a cost of 1 is induced. Hence, $\mathrm{dtw}(x,y)^2 \geq \lceil (n-m)/2 \rceil$. If $x_1 = y_1$, then this lower bound matches the claimed cost.

If $x_1 \neq y_1$ and $n = m$, then also $x_n \neq y_m$ and hence the cost is at least 2.

If $x_1 \neq y_1$ and $n > m$, then we can assume that $p_2 = (2,1)$. To see this, note that for $p_2 \neq (2,1)$ the cost is at least $1 + \lceil (n-m)/2 \rceil$ by the above argument. If $p_2 = (2,1)$, then the subpath $(p_2, p_3, \ldots, p_L)$ of p is an optimal warping path between $(x_2,\ldots,x_n)$ and y, where $x_2 = y_1$ and $n - 1 \geq m$. As we have already shown above, this path has cost $\lceil (n-1-m)/2 \rceil$. Hence, in this case p has cost $1 + \lceil (n-1-m)/2 \rceil = 1 + \lfloor (n-m)/2 \rfloor$. Thus, we can assume that $p_2 = (2,1)$ in which case the cost of p matches the claimed cost of the lemma. This finishes the proof. $\square$

Note that according to Lemma 1, for a fixed condensed binary time series y of length m, the value $\mathrm{dtw}(x,y)^2$ is monotonically increasing in the length of x for all condensed binary time series x of length $n \geq m+1$. We use this property later in the proof of Lemma 3 where we derive an upper bound on the length of a binary mean. In order to prove Lemma 3, we also need the following lemma concerning the dtw-distances between condensed and non-condensed time series.

Lemma 2. *Let $x = (x_1, \ldots, x_n)$ be a condensed binary time series and let $y = (y_1, \ldots, y_m) \in \{0,1\}^*$ with $n \geq m$. Then, for the condensation $\tilde{y}$ of y it holds $\mathrm{dtw}(x,y)^2 = \mathrm{dtw}(x,\tilde{y})^2$.*

Proof. Assume that y is not condensed. Then, y consists of $\ell \in [m]$ *blocks*, where a block is a maximal subsequence of consecutive 0's or consecutive 1's in y. Let $m_1, \ldots, m_\ell$ denote the lengths of these blocks where $m_1 + \ldots + m_\ell = m$. Note also that $\tilde{y}$ has length ℓ with $\ell < m \leq n$. We define a warping path p between x and y such that $C_p(x,y) = \mathrm{dtw}(x,\tilde{y})^2$. Note that, by Lemma 1, we have

$$\mathrm{dtw}(x,\tilde{y})^2 = \begin{cases} \lceil (n-\ell)/2 \rceil, & x_1 = \tilde{y}_1 \\ 1 + \lfloor (n-\ell)/2 \rfloor, & x_1 \neq \tilde{y}_1 \end{cases}.$$

If $x_1 = y_1$, then we set $p := ((1,1), \ldots, (1,m_1), (2, m_1+1), \ldots, (2, m_1 + m_2), \ldots, (\ell, m), (\ell+1, m), \ldots, (n,m))$ and obtain cost $C_p(x,y) = \sum_{i=\ell+1}^{n} |x_i - y_m| = \lceil (n-\ell)/2 \rceil$.

If $x_1 \neq y_1$, then we set $p := ((1,1), (2,1), \ldots, (2,m_1), (3, m_1+1), \ldots, (3, m_1 + m_2), \ldots, (\ell+1, m), (\ell+2, m), \ldots, (n,m))$ and obtain cost

$$C_p(x,y) = 1 + \sum_{i=\ell+2}^{n} |x_i - y_m| = 1 + \lfloor (n-\ell)/2 \rfloor. \quad \square$$

We now have all ingredients to show that there always exists a binary mean of length at most one larger than the maximum length of any input time series.

Lemma 3. *For binary input time series $x^{(1)}, \ldots, x^{(k)} \in \{0,1\}^*$ of maximum length n, there exists a binary mean $z \in \{0,1\}^*$ of length at most $n+1$.*

Proof. Assume that $z = (z_1, \ldots, z_m) \in \{0,1\}^*$ is a mean of length $m > n+1$. By Proposition 1, we can assume that z is condensed, that is, $z_i \neq z_{i+1}$ for all $i \in [m-1]$. We claim that $z' := (z_1, \ldots, z_{n+1})$ is also a mean. We prove this claim by showing that $\mathrm{dtw}(z', x^{(i)})^2 \leq \mathrm{dtw}(z, x^{(i)})^2$ holds for all $i \in [k]$. By Lemmas 1 and 2, we have $\mathrm{dtw}(z', x^{(i)})^2 = \mathrm{dtw}(z', \tilde{x}^{(i)})^2 \leq \mathrm{dtw}(z, \tilde{x}^{(i)})^2 = \mathrm{dtw}(z, x^{(i)})^2$, where the inequality follows from Lemma 1 since z' is of length $n+1 < m$ and the dtw-distance is monotonically increasing. $\square$

Having established that a binary mean is always condensed and of bounded length, we now show that it can be found in polynomial time.

Proof of Theorem 1. By Proposition 1 and Lemma 3, we can assume the desired mean z to be a condensed series of length at most $n+1$. Thus, there are at

most $2n + 2$ many possible candidates for z. For each candidate z, we can compute the value $F(z)$ with $O(kn^2)$ arithmetic operations and select the one with the smallest value. This yields an overall number of arithmetic operations in $O(kn^3)$. □

5 An Exact Algorithm Solving DTW-Mean

We develop a nontrivial exponential-time algorithm solving DTW-Mean exactly. The key is to observe a certain structure of a mean and the corresponding alignments to the input time series. To this end, we define redundant elements in a mean. Note that this concept was already used by Jain and Schultz [13] in order to prove the existence of a mean of bounded length [13, Theorem 2.7] (though [13, Definition 3.20] is slightly different).

Definition 3. *Let $x^{(1)}, \ldots, x^{(k)}$ and $z = (z_1, \ldots, z_m)$ be time series and let $p^{(j)}$, $j \in [k]$ denote an optimal warping path between $x^{(j)}$ and z. We call an element z_i of z* redundant *if in every time series $x^{(j)}$ there exists an element that is aligned with z_i and with another element of z by $p^{(j)}$.*

The next lemma states that there always exists a mean without redundant elements. The proof is implicitly contained in the proof of [13, Theorem 2.7]. For the sake of clarity, however, we give an explicit proof here.

Lemma 4. *There exist a mean z for time series $x^{(1)}, \ldots, x^{(k)}$ and optimal warping paths $p^{(j)}$ between z and $x^{(j)}$ for each $j \in [k]$ such that z contains no redundant element.*

Proof. Let z be a mean of $x^{(1)}, \ldots, x^{(k)}$ with optimal warping paths $p^{(j)}$, $j \in [k]$, such that the element z_i is redundant. Note that by [13, Proposition 2.10] a mean z always exists. We show that there also exists a mean z' and optimal warping paths $p^{(j)'}$ such that no element in z' is redundant.

Assume first that there exists a $j \in [k]$ such that the element z_i is aligned by $p^{(j)}$ with at least one element $x_\ell^{(j)}$ in $x^{(j)}$ that is not aligned with any other element in z. Then, $p^{(j)}$ is of the form

$$p^{(j)} = (p_1, \ldots, (i-1, \ell_{t-1}), (i, \ell_t), \ldots, (i, \ell_{t+\alpha}), (i+1, \ell_{t+\alpha+1}), \ldots, p_L)$$

for some $\ell_t \leq \ell \leq \ell_{t+\alpha}$ and $\alpha \geq 1$. Since z_i is redundant, it follows that $\ell_{t-1} = \ell_t$ or $\ell_{t+\alpha} = \ell_{t+\alpha+1}$ holds. If $\ell_{t-1} = \ell_t$, then we remove the pair (i, ℓ_t) from $p^{(j)}$. Also, if $\ell_{t+\alpha} = \ell_{t+\alpha+1}$, then we remove the pair $(i, \ell_{t+\alpha})$ from $p^{(j)}$. Note that this yields a warping path $p^{(j)'}$ between z and $x^{(j)}$ since even if we removed both pairs (i, ℓ_t) and $(i, \ell_{t+\alpha})$, then we know by assumption that there still exists the pair (i, ℓ) with $\ell_t < \ell < \ell_{t+\alpha}$ in $p^{(j)'}$ since z_i is aligned with $x_\ell^{(j)}$ which is not aligned with another element in z. Since we only removed pairs from $p^{(j)}$, it holds $C_{p^{(j)'}}(z, x^{(j)}) \leq C_{p^{(j)}}(z, x^{(j)})$. Moreover, z_i is not redundant anymore.

Now, assume that for all $j \in [k]$, z_i is aligned only with elements in $x^{(j)}$ which are also aligned with another element of z by $p^{(j)}$ (that is, z_i is redundant

Figure 2: Illustration of the dynamic program computing a mean $(z_1, \ldots, z_q)$ for the subseries $(x_1^{(1)}, \ldots, x_{i_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{i_k}^{(k)})$ (white elements are not considered at the current iteration). The light gray elements are aligned to the last mean element z_q. The remaining mean elements $(z_1, \ldots, z_{q-1})$ form an optimal mean for the dark gray subseries.

according to [13, Definition 3.20]). Let z' denote the time series obtained by deleting the element z_i from z. Jain and Schultz [13, Proof of Theorem 2.7] showed that in this case $\mathrm{dtw}(z', x^{(j)}) \leq \mathrm{dtw}(z, x^{(j)})$ holds for all $j \in [k]$.

Under both assumptions above, we reduced the number of redundant elements. Hence, we can repeat the above arguments until we obtain a mean z' without redundant elements. □

Lemma 4 allows us to devise a dynamic program computing a mean without redundant elements. We compute a mean by testing all possibilities to align the last mean element to elements from the input time series while recursively adding an optimal solution for the remaining non-aligned elements in the input time series (see Figure 2). The fact that we assume the mean not to contain redundant elements is crucial for this recursive approach, which is described in the proof of the following theorem.

Theorem 2. DTW-Mean *for k input time series is solvable with $O(n^{2k+1}2^k k)$ arithmetical operations, where n is the maximum length of all input time series.*

Proof. Assume for simplicity that all time series have length n (the general case can be solved analogously). We find a mean using a dynamic programming approach. Let C be a k-dimensional table, where for all $(i_1, \ldots, i_k) \in [n]^k$, we define

$$C[i_1, \ldots, i_k] = \min_{z \in \mathcal{T}} \left(\frac{1}{k} \sum_{j=1}^{k} \left(\mathrm{dtw}(z, (x_1^{(j)}, \ldots, x_{i_j}^{(j)})) \right)^2 \right),$$

that is, $C[i_1, \ldots, i_k]$ is the value $F(z)$ of the Fréchet function of a mean z for the subseries $(x_1^{(1)}, \ldots, x_{i_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{i_k}^{(k)})$. Clearly, $C[n, \ldots, n]$ yields the optimal value $F(z)$ of the input instance.

For $i_1 = i_2 = \ldots = i_k = 1$ it is clear that there exists a mean z containing just one element and each optimal warping path between z and $(x_1^{(j)})$ trivially equals $((1,1))$. Since it can be shown that, given optimal warping paths, each element of a mean equals the arithmetic mean of all elements in all time series

Algorithm 1: Exact Dynamic Program (EDP) solving DTW-MEAN

Input: Time series $x^{(1)}, \ldots, x^{(k)}$ of length n.
Output: Mean z and $F(z)$.

```
Initialize C                          // k-dimensional DP table storing F-values
Initialize Z                          // k-dimensional table storing means
foreach (i_1, ..., i_k) ∈ [n]^k do    // fill tables iteratively
    C[i_1, ..., i_k] := ∞
    Z[i_1, ..., i_k] := ()
    foreach (ℓ_1, ..., ℓ_k) ∈ [i_1] × ... × [i_k] do   // compute C[i_1, ..., i_k]
        μ := (Σ_{j=1}^{k} Σ_{t=ℓ_j}^{i_j} x_t^(j)) / Σ_{j=1}^{k} (i_j − ℓ_j + 1)
        σ := (1/k) Σ_{j=1}^{k} Σ_{t=ℓ_j}^{i_j} (x_t^(j) − μ)^2
        c* := ∞
        if ℓ_1 = ℓ_2 = ... = ℓ_k = 1 then
            c* := 0
        else          // compute c*(ℓ_1, ..., ℓ_k) based on table look-ups
            foreach (ℓ'_1, ..., ℓ'_k) ∈ {ℓ_1 − 1, ℓ_1} × ... × {ℓ_k − 1, ℓ_k} do
                if ∀j ∈ [k] : ℓ'_j ≥ 1 and ∃j ∈ [k] : ℓ'_j < ℓ_j then
                    if C[ℓ'_1, ..., ℓ'_k] < c* then
                        c* := C[ℓ'_1, ..., ℓ'_k]
                        Z[i_1, ..., i_k] := Z[ℓ'_1, ..., ℓ'_k]
        if c* + σ < C[i_1, ..., i_k] then   // update mean and F-value
            C[i_1, ..., i_k] := c* + σ
            append(Z[i_1, ..., i_k], μ)
return (Z[n, ..., n], C[n, ..., n])
```

that are aligned to this element [23, Theorem 3.3], we initialize

$$C[1,\ldots,1] = \frac{1}{k}\sum_{j=1}^{k}(x_1^{(j)} - \mu)^2, \qquad \mu = 1/k\sum_{j=1}^{k} x_1^{(j)}.$$

Note that the corresponding mean is $z = (\mu)$.

For the case that $i_j > 1$ holds for at least one $j \in [k]$, consider a mean z for $(x_1^{(1)}, \ldots, x_{i_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{i_k}^{(k)})$. By Lemma 4, we can assume that there exist optimal warping paths $p^{(j)}$ between z and $(x_1^{(j)}, \ldots, x_{i_j}^{(j)})$ such that z contains no redundant elements. Let z_q be the last element of z. Then, for each $j \in [k]$, z_q is aligned by $p^{(j)}$ with some elements $x_{\ell_j}^{(j)}, \ldots, x_{i_j}^{(j)}$ for $\ell_j \in [i_j]$. By [23, Theorem 3.3], z_q is the arithmetic mean of all elements in all time series with which it is aligned. Hence, the contribution of z_q to the overall cost $F(z)$ can be determined by the formulas below. Now, consider the case that there exists another element z_{q-1} in z. Clearly, for each $j \in [k]$, z_{q-1} is aligned only with elements of indices up to ℓ_j since otherwise the warping path conditions are violated. Hence, the remaining cost can be obtained recursively from a mean

of the time series $(x_1^{(1)}, \ldots, x_{\ell_1}^{(1)}), \ldots, (x_1^{(k)}, \ldots, x_{\ell_k}^{(k)})$. Recall, however, that we assumed z not to contain any redundant element. It follows that z_{q-1} cannot be aligned with $x_{\ell_j}^{(j)}$ for all $j \in [k]$ since z_q is already aligned with each $x_{\ell_j}^{(j)}$. That is, we add the minimum value $C[\ell'_1, \ldots, \ell'_k]$ over all $\ell'_j \in \{\max(1, \ell_j - 1), \ell_j\}$, where $\ell'_j = \ell_j - 1$ holds for at least one $j \in [k]$. Hence, the following recursion holds: $C[i_1, \ldots, i_k] = \min\{c^*(\ell_1, \ldots, \ell_k) + \sigma(\ell_1, \ldots, \ell_k) \mid \ell_1 \in [i_1], \ldots, \ell_k \in [i_k]\}$, where

$$\sigma(\ell_1, \ldots, \ell_k) = \frac{1}{k} \sum_{j=1}^{k} \sum_{t=\ell_j}^{i_j} (x_t^{(j)} - \mu)^2,$$

$\mu = (\sum_{j=1}^{k} \sum_{t=\ell_j}^{i_j} x_t^{(j)}) / (\sum_{j=1}^{k} (i_j - \ell_j + 1))$, and we define $c^*(\ell_1, \ldots, \ell_k) = \min\{C[\ell'_1, \ldots, \ell'_k] \mid \ell'_j \in \{\max(1, \ell_j - 1), \ell_j\}, \sum_{j=1}^{k} (\ell_j - \ell'_j) > 0\}$ if $\ell_j > 1$ holds for some $j \in [k]$. We set $c^*(1, \ldots, 1) := 0$.

The minimum is computed over all possible choices $\ell_j \in [i_j]$, $j \in [k]$. For each choice, the arithmetic mean μ corresponds to the element of the mean and $\sigma(\ell_1, \ldots, \ell_k)$ equals the induced cost for aligning this element with $x_{\ell_j}^{(j)}, \ldots, x_{i_j}^{(j)}$ for each $j \in [k]$. The value $c^*(\ell_1, \ldots, \ell_k)$ recursively yields the optimal cost of a mean for all remaining subseries $(x_1^{(j)}, \ldots, x_{\ell'_j}^{(j)})$ over all $\ell'_j \in \{\max(1, \ell_j - 1), \ell_j\}$ such that $\sum_{j=1}^{k} (\ell_j - \ell'_j) > 0$, which implies that $\ell'_j = \ell_j - 1$ holds for at least one $j \in [k]$. This condition guarantees that we only find alignments which do not yield redundant elements in the mean (which we can assume by Lemma 4). Note that $\ell_j = 1$ implies that $\ell'_j = 1$ (since index 0 does not exist in $x^{(j)}$).

The dynamic programming table C can be filled iteratively along the dimensions starting from $C[1, \ldots, 1]$. The overall number of entries is n^k. For each table entry, the minimum of a set containing $O(n^k)$ elements is computed. Computing an element requires the computation of $\sigma(\ell_1, \ldots, \ell_k)$ which can be done with $O(kn)$ arithmetical operations plus the computation of $c^*(\ell_1, \ldots, \ell_k)$ which is the minimum of a set of size at most 2^k whose elements can be obtained by constant-time table look-ups. Thus, the table can be filled using $O(n^k \cdot n^k \cdot 2^k \cdot kn)$ arithmetical operations. A mean can be obtained by storing the values μ for which the minimum in the above recursion is attained (Algorithm 1 contains the pseudocode). □

We close with some remarks on the above result.

- The dynamic program also allows to compute all non-redundant means by storing all possible values for which the minimum in the recursion is attained.
- It is possible to incorporate a maximum mean length q into the dynamic program (by adding another dimension to the table C) such that it outputs only optimal solutions of length upper-bounded by the specified maximum. The running time increases by a factor of q.
- The algorithm can easily be extended to multivariate time series with elements in $\mathbb{Q}^d$ and a cost function $C_p(x, y) := \sum_{\ell=1}^{L} \|x_{i_\ell} - y_{j_\ell}\|_2^2$ (with a running time increase by a factor of d).

6 Experiments

The goal of this section is to assess the performance of state-of-the-art heuristics in terms of minimizing the Fréchet function. To this end, we use Algorithm 1 to obtain exact solutions.

6.1 Test Data

We selected 10 datasets from the UCR time series classification and clustering repository [6]. The time series within each dataset are of identical length and range from 24 to 150 (see Table 1).

UCR	#	L
ItalyPowerDemand	1096	24
SyntheticControl	600	60
SonyAIBORobotSurface1	621	70
ProximalPhalanxTW	605	80
ProximalPhalanxOutlineCorrect	891	80
MedicalImages	1141	99
TwoPatterns	5000	128
FaceAll	2250	131
ECG5000	5000	140
GunPoint	200	150

Table 1: Ten UCR time series data sets. The column marked with # contains the numbers of time series in the data set and the column L denotes the lengths of the time series.

6.2 Tested Algorithms

The following table lists the algorithms which we compared in our experiments:

Algorithm	Acr.	Ref.
exact dynamic programming	EDP	Alg. 1
multiple alignment	MAL	[12, 19]
DTW barycenter averaging	DBA	[20]
soft-dtw	SDTW	[7]
batch subgradient	BSG	[7, 23]
stochastic subgradient	SSG	[23]

We implemented MAL (for two time series) and EDP in Java. For SDTW and BSG we used the python implementation provided by Blondel [3]. For DBA and SSG we used the implementation by Schultz and Jain [24]. EDP and MAL required no parameter optimization. For the other algorithms, the following settings were used: We optimized the algorithms on every sample using

different parameter configurations and reported the best result. Every configuration was composed of an initialization method and an optional parameter selection. As initialization we used (i) the arithmetic mean of the sample, (ii) a random time series from the sample, and (iii) a random time series drawn from a normal distribution with zero mean and standard deviation one. DBA and BSG required no additional parameter selection. For SDTW, we selected the best smoothing parameter $\gamma \in \{0.001, 0.01, 0.1, 1\}$ and for SSG the best initial step size $\eta_0 \in \{0.25, 0.2, 0.15, 0.1\}$. The step size of SSG was linearly decreased from η_0 to $\eta_0/10$ within the first 100 epochs and then remained constant for the remaining 100 epochs (one epoch is a full pass through the sample). Thus, for DBA and BSG, we picked the best result from three parameter configurations for each sample, and for SDTW and SSG we picked the best result from twelve parameter configurations. The length of the mean was set beforehand to the length of the sample time series (here, all time series of a sample have identical length). For every sample and every parameter configuration, all algorithms terminated after 200 epochs at the latest. The tolerance parameter of SDTW and BSG was set to $\varepsilon = 10^{-6}$.

6.3 Experimental Setup

Due to the restrictive running time of EDP, our experiments are limited to small sample sizes and time series lengths. We conducted two types of experiments: As a basic case, we compared the algorithms on $k = 2$ time series of full length (E1). Additionally, we compared all algorithms on $k = 2, \ldots, 5$ time series of length 8 in order to investigate the performance for larger sample sizes (E2).

In experiment E1, we randomly selected 100 different pairs of time series from every dataset. Thus, we ran each algorithm on a total of $1,000$ pairs. For E2, we randomly selected 100 different samples consisting of k time series from every dataset for every $k \in \{2, \ldots, 5\}$. Then, we randomly selected a consecutive subseries of length 24 from every time series and sampled it down to a series of length 8 by picking every third element. We ran each algorithm (except MAL) on a total of $1,000$ samples of size k.

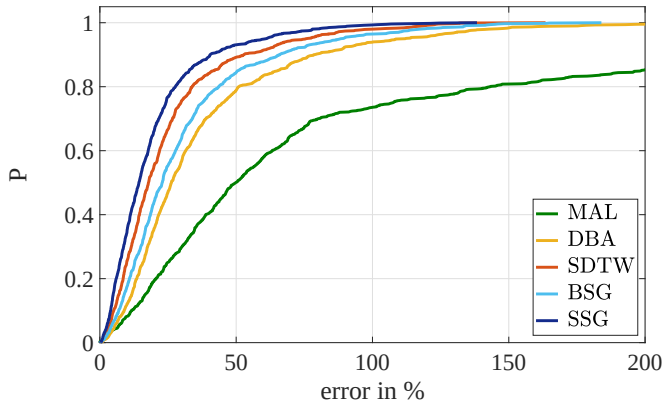
As quality measure for algorithm A we used the error percentage $E_A = 100 \cdot (F_A - F_*)/F_*$, where F_A is the value of the Fréchet function of the best solution obtained by A on a given sample and F_* is the optimal value (that is, the value of the Fréchet function of a mean obtained by EDP).

6.4 Results and Discussion

We present and discuss the results of both experiments separately.

Experiment E1. Figure 3 summarizes the results. We made the following observations:

First, the results of MAL clearly show that multiple alignment does not always yield optimal solutions (see Section 3.2). Moreover, with an average error of 114.2% and a maximum error of 1039.9%, MAL achieved the worst



	avg	std	max	eq
MAL	114.2	175.1	1093.9	3
DBA	37.1	34.7	274.3	0
SDTW	24.7	23.2	163.5	0
BSG	30.6	27.3	184.0	1
SSG	19.5	18.8	138.3	0

Figure 3: Results of experiment E1. Quality measure is the error percentage. **Top:** Error probabilities of all heuristics (a steeper curve indicates better quality) relative to the exact algorithm EDP. The curves are truncated at an error of 200% for the sake of presentation. A point (E_A, P_A) on the curve of algorithm A states that solutions F_A obtained by A deviate by at most E_A percent from the optimal solution F_* with probability P_A (estimated from all $1,000$ results of E1). **Bottom:** Average error percentage (avg), standard deviation (std), and maximum error percentage (max) of the heuristics. The last column shows how often an optimal solution was found by the heuristic.

results on average and was by far the least robust method. These findings may partly explain why algorithms based on pairwise mean computation such as NLAAF [10] and PSA [16] are not competitive [20, 25].

Second, the average error percentage of the other algorithms range from 19.5% (SSG) to 37.1% (DBA), while the maximum error percentage ranged from 138.3% (SSG) to 274.3% (DBA). These results suggest that state-of-the-art heuristics are far from being optimal on average and can fail significantly in the worst cases even on small problem instances. One potential cause of error is that all algorithms arbitrarily pre-specify the length of a solution without any guarantee that a mean of this length exists (in Figure 4 we illustrate a mean of length strictly smaller than the length of the input series). Thus, for reliably estimating a mean (and the variance) of a dataset, which belongs to the most important and fundamental tasks in data analysis, there is a need to devise better algorithms that substantially improve the state-of-the-art.

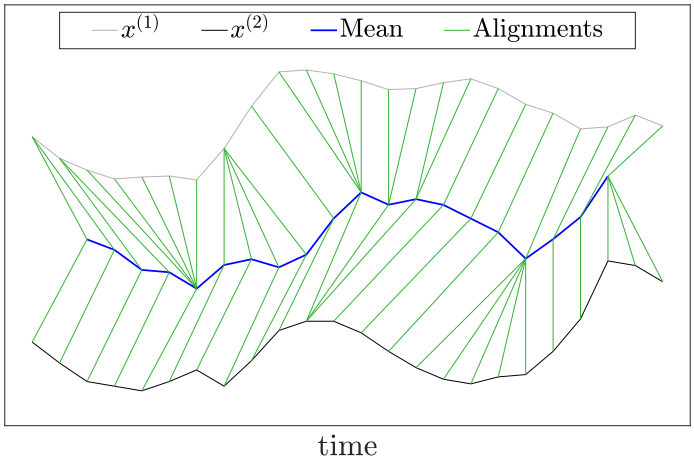


Figure 4: Mean of two time series from the ItalyPowerDemand dataset. The time series have been shifted along both axes for clearer presentation of the alignments. Observe that the mean is shorter than the sample time series since for example multiple points of the broad valley of $x^{(2)}$ are aligned with a single point in the mean.

Experiment E2. Figure 5 summarizes the results. Compared to the results of experiment E1, both plots show a similar (but slightly improved) pattern. The upper plot in Figure 5 shows that the heuristics on average still deviate by at least 10% from the exact solution in all cases. All heuristics slightly improve on average with increasing sample size. A similar statement can be made for the standard deviation, which is still at a high level. In contrast, there is no visible trend with respect to the maximum error percentage (lower plot in Figure 5).

7 Conclusion

We conclude with some challenges for future research. The main complexity open question (as discussed in Section 3) is whether DTW-Mean is NP-hard. From an algorithmic point of view it is interesting to investigate whether one can extend the polynomial-time solvability of the binary case of DTW-Mean to larger "alphabet" sizes; already the case of alphabet size three is open. Our exact dynamic programming algorithm for DTW-Mean does not yield fixed-parameter tractability with respect to the parameter number k of input time series; again, this remains open and a positive answer would most likely mean significantly increased practical impact. Finally, we wonder whether there are other practically relevant restrictions of DTW-Mean that make the problem more tractable. On an applied side, our experimental results strongly motivate the search for further improved, more "robust" heuristics.

Acknowledgements. This work is supported by the Deutsche Forschungsgemeinschaft under grants JA 2109/4-1 and NI 369/13-2, and by a Feodor Lynen return

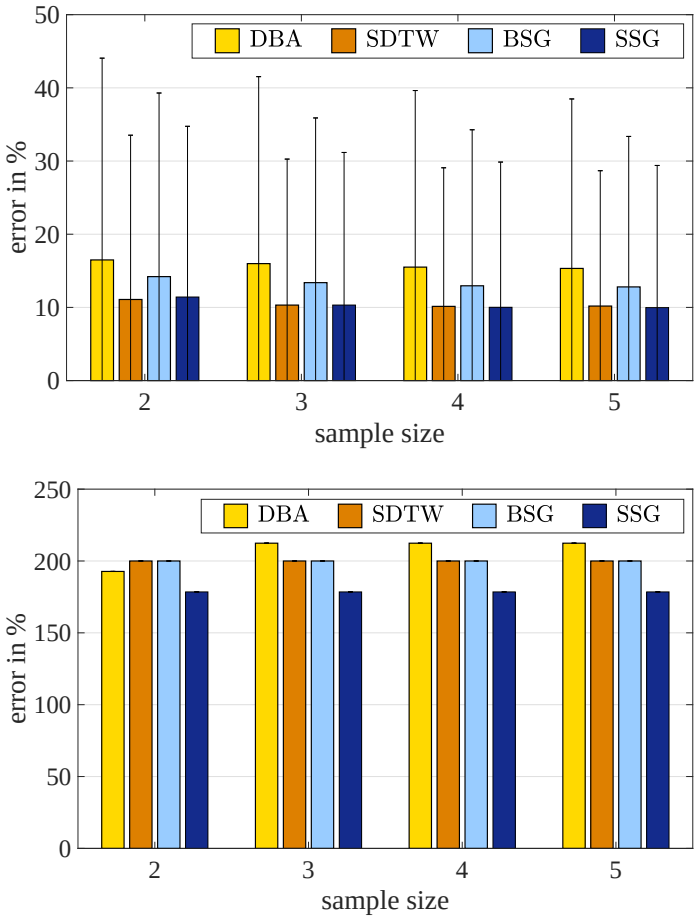


Figure 5: Results of experiment E2. Shown are the error percentages as a function of the sample size. **Top:** Average error percentage and standard deviation. **Bottom:** Maximum error percentage.

fellowship of the Alexander von Humboldt Foundation. The work on the theoretical part of this paper started at the research retreat of the Algorithmics and Computational Complexity group, TU Berlin, held at Boiensdorf, Baltic Sea, April 2017, with MB, TF, VF, and RN participating.

References

[1] W. Abdulla, D. Chow, and G. Sin. Cross-words reference template for DTW-based speech recognition systems. In *Conference on Convergent Technologies for Asia-Pacific Region*, 2003.

[2] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering – A decade review. *Information Systems*, 53:16–38, 2015.

[3] M. Blondel. Python implementation of soft-DTW. `https://github.com/mblondel/soft-dtw`, 2017.

[4] P. Bonizzoni and G. Della Vedova. The complexity of multiple sequence alignment with SP-score that is a metric. *Theoret. Comput. Sci.*, 259(1–2): 63–79, 2001.

[5] K. Bringmann and M. Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. of 56th FOCS*, pages 79–97. IEEE, 2015.

[6] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The UCR time series classification archive, 2015. URL `www.cs.ucr.edu/~eamonn/time_series_data/`.

[7] M. Cuturi and M. Blondel. Soft-DTW: a differentiable loss function for time-series. In *Proc. of 34th ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 894–903. PMLR, 2017.

[8] M. Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l'institut Henri Poincaré*, pages 215–310, 1948.

[9] O. Gold and M. Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. In *Proc. of 44th ICALP*, volume 80 of *LIPIcs*, pages 25:1–25:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[10] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Trans. Biomed. Eng.*, 43(4):348–356, 1996.

[11] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[12] V. Hautamaki, P. Nykanen, and P. Franti. Time-series clustering by approximate prototypes. In *Proc. of 19th ICPR*, pages 1–4. IEEE, 2008.

[13] B. Jain and D. Schultz. A reduction theorem for the sample mean in dynamic time warping spaces. *CoRR*, abs/1610.04460, 2016.

[14] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson. Time-series averaging using constrained dynamic time warping with tolerance. *Pattern Recognition*, 2017. in press.

[15] F. Nicolas and E. Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *J. Discrete Algorithms*, 3(2):390–415, 2005.

[16] V. Niennattrakul and C. A. Ratanamahatana. Shape averaging under time warping. In *Proc. of 6th ECTI-CON*, volume 02, pages 626–629, 2009.

[17] T. Oates, L. Firoiu, and P. Cohen. Clustering time series with hidden markov models and dynamic time warping. In *IJCAI Workshop on Neural, Symbolic and Reinforcement Learning methods for Sequence Learning*, 1999.

[18] J. Paparrizos and L. Gravano. k-shape: Efficient and accurate clustering of time series. In *Proc. of 2015 ACM SIGMOD*, pages 1855–1870. ACM, 2015.

[19] F. Petitjean and P. Gançarski. Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment. *Theor. Comput. Sci.*, 414(1):76–91, 2012.

[20] F. Petitjean, A. Ketterlin, and P. Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[21] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowledge and Information Systems*, 47(1):1–26, 2016.

[22] L. Rabiner and J. Wilpon. Considerations in applying clustering techniques to speaker-independent word recognition. *The Journal of the Acoustical Society of America*, 66(3):663–673, 1979.

[23] D. Schultz and B. Jain. Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. *Pattern Recognition*, 74 (Supplement C):340–358, 2018.

[24] D. Schultz and B. J. Jain. Sample mean algorithms for averaging in dynamic time warping spaces, 2016. URL `https://doi.org/10.5281/zenodo.216233`.

[25] S. Soheily-Khah, A. Douzal-Chouakria, and E. Gaussier. Progressive and iterative approaches for time series averaging. In *Proc. of 1st International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD '15)*, 2015.

[26] S. Soheily-Khah, A. Douzal-Chouakria, and E. Gaussier. Generalized k-means-based clustering for temporal data under weighted and kernel time warp. *Pattern Recognition Letters*, 75:63–69, 2016.

[27] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J. Comput. Biol.*, 1(4):337–348, 1994.