Tight Hardness Results for Consensus Problems on Circular Strings and Time Series

Laurent Bulteau, Vincent Froese, Rolf Niedermeier

TIGHT HARDNESS RESULTS FOR CONSENSUS PROBLEMS ON CIRCULAR STRINGS AND TIME SERIES*

LAURENT BULTEAU†, VINCENT FROESE‡, AND ROLF NIEDERMEIER‡

Abstract. Consensus problems for strings and sequences appear in numerous application contexts, ranging among bioinformatics, data mining, and machine learning. Closing some gaps in the literature, we show that several fundamental problems in this context are NP- and W[1]-hard, and that the known (including some brute-force) algorithms are close to optimality assuming the Exponential Time Hypothesis. Among our main contributions is to settle the complexity status of computing a mean in dynamic time warping spaces which, as pointed out by Brill et al. [DMKD 2019], suffered from many unproven or false assumptions in the literature. We prove this problem to be NP-hard and additionally show that a recent dynamic programming algorithm is essentially optimal. In this context, we study a broad family of circular string alignment problems. This family also serves as a key for our hardness reductions, and it is of independent (practical) interest in molecular biology. In particular, we show tight hardness and running time lower bounds for Circular Consensus String; notably, the corresponding non-circular version is easily linear-time solvable.

Key words. Circular String Alignment, Time Series Averaging, Dynamic Time Warping, Fine-Grained Complexity and Reductions, Lower Bounds, Parameterized Complexity, Exponential Time Hypothesis

AMS subject classifications. 68Q17, 68T10, 92D20

1. Introduction.

Consensus problems appear in many contexts of stringology and time series analysis, including applications in bioinformatics, data mining, machine learning, and speech recognition. Roughly speaking, given a set of input sequences, the goal is to find a consensus sequence that minimizes the "distance" (according to some specified distance measure) to the input sequences. Classic problems in this context are the NP-hard Closest String [15, 25, 24, 18] (where the goal is to find a "closest string" that minimizes the maximum Hamming distance to a set of equal-length strings) or the more general Closest Substring [13, 26]. Notably, the variant of Closest String where one minimizes the sum of Hamming distances instead of the maximum distance is easily solvable in linear time.

In this work, we settle the computational complexity of prominent consensus problems on circular strings and time series. Despite their great importance in many applications, and a correspondingly rich set of heuristic solution strategies used in practice, to date, it has been unknown whether these problems are polynomial-time solvable or NP-hard. We prove their hardness, including also "tight" parameterized and fine-grained complexity results, thus justifying the massive use of heuristic solution strategies in real-world applications.

On the route to determining the complexity of exact mean computation in dynamic time warping spaces, a fundamental consensus problem in the context of time series analysis [33][1], we first study a fairly general alignment problem[2] for circular

*Submitted to the editors DATE.

†Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77454, Marne-la-Vallée, France (laurent.bulteau@u-pem.fr).

‡Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Berlin, Germany (vincent.froese@tu-berlin.de, rolf.niedermeier@tu-berlin.de).

[1] As of May 2020, according to Google Scholar the work by Petitjean et al. [33], who developed one of the most prominent heuristics for this problem, has already been cited around 460 times since 2011.

[2] Particularly from the viewpoint of applications in bioinformatics, consensus string problems can also be interpreted as alignment problems [23].

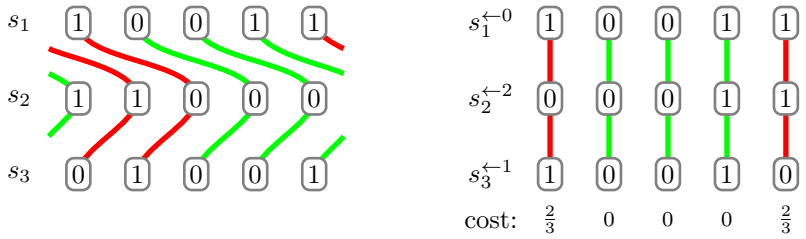

FIG. 1. *An instance of σ-MSCS with three binary input strings, and an optimal multiple circular shift $\Delta = (0, 2, 1)$, using the sum of squared distances from the mean (σ) as a cost function. Columns of Δ are indicated with dark (red) or light (green) lines, depending on their cost. For example, column 1 with values $(1, 0, 1)$ has mean $\frac{2}{3}$ and cost $\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 = \frac{2}{3}$. The overall cost is $\frac{4}{3}$.*

strings called MULTIPLE STRING CIRCULAR SHIFT (WITH COST f). Based on its analysis, we will also derive our results for two further, more specific problems. Given a set of input strings over a fixed alphabet Σ and a local cost function $f\colon \Sigma^* \to \mathbb{Q}$, the goal in MULTIPLE STRING CIRCULAR SHIFT (WITH COST f) (abbreviated by f-MSCS) is to find a cyclic shift of each input string such that the shifted strings "align well" in terms of the sum of local costs.[3]

f-MSCS
Input: A list of k strings $s_1, \ldots, s_k \in \Sigma^n$ of length n and $c \in \mathbb{Q}$.
Question: Is there a multiple circular shift $\Delta = (\delta_1, \ldots, \delta_k) \in \mathbb{N}^k$ with $\text{cost}_f(\Delta) := \sum_{i=1}^{n} f\big((s_1^{\leftarrow \delta_1}[i], \ldots, s_k^{\leftarrow \delta_k}[i])\big) \leq c$?

Here, $s^{\leftarrow \delta}$ denotes a circular shift of s by δ (see Section 2 for details). See Figure 1 for an example. We separately study the special case CIRCULAR CONSENSUS STRING for a binary alphabet, where the cost function $f\colon \{0,1\}^* \to \mathbb{N}$ is defined as $f((x_1, \ldots, x_k)) := \min\{\sum_{i=1}^{k} x_i, k - \sum_{i=1}^{k} x_i\}$. This corresponds to minimizing the sum of Hamming distances (not the maximum Hamming distance as in CLOSEST STRING). As we will show, allowing circular shifts makes consensus string problems much harder to solve.

Multiple circular string (sequence) alignment problems have been considered in different variations in bioinformatics, where circular strings naturally arise in several applications (for example, in multiple alignment of genomes, which often have a circular molecular structure) [4, 5, 14, 19, 27, 37]. Depending on the application at hand, different cost functions are used. For example, non-trivial algorithms for computing a consensus string of three and four circular strings with respect to the Hamming distance have been developed [23]. However, most of the algorithmic work so far is heuristic in nature or only considers specific special cases. A thorough analysis of the computational complexity for these problems in general so far has been missing.

After having dealt with circular string alignment problems in a quite general fashion, we then study a fundamental (consensus) problem in time series analysis. *Dynamic time warping* (see Section 2 for details) defines a distance between two time series which is used in many applications in time series analysis [21, 28, 33, 36] (notably, dynamic time warping has also been considered in the context of circular

[3] We cast all problems in this work as decision problems for easier complexity-theoretic treatment. Our hardness results correspondingly hold for the associated optimization problems.

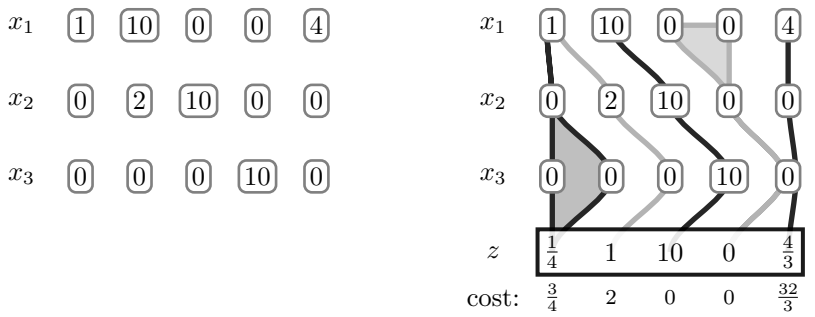

FIG. 2. *A* DTW-MEAN *instance with three input sequences and an optimal length-5 mean (z). Alignments between the mean and input sequences can progress at different speeds. This is formalized using* warping paths *(see Section 2) represented by polygons (or lines in degenerate cases) with alternating shades. Every pair of aligned elements belongs to the same polygon. The cost of each mean element is the sum of squared differences over all aligned input elements, e.g. the cost of the first element is* $(1-\frac{1}{4})^2+3\cdot(0-\frac{1}{4})^2=\frac{3}{4}$.

sequences [3, 29]). An important problem here is to compute an average of a given sample of time series under the dynamic time warping distance.

DTW-MEAN
Input: A list of k univariate rational time series $x_1,\dots,x_k$ and $c\in\mathbb{Q}$.
Question: Is there a univariate rational time series z such that $\mathcal{F}(z)=\sum_{i=1}^{k}(\mathrm{dtw}(z,x_i))^2\leq c$?

Here, dtw denotes the dynamic time warping distance (see Section 2 for details). Intuitively, dynamic time warping allows for non-linear alignments between two series. Figure 2 depicts an example. The dtw-distance of two length-n time series can be computed via standard dynamic programming in $O(n^2)$ time. Some subquadratic algorithms are known [17, 22, 16]. For two binary time series, there exists an $O(n^{1.87})$-time algorithm [1]. In general, however, a strongly subquadratic-time algorithm (that is, $O(n^{2-\varepsilon})$ time for some $\varepsilon>0$) does not exist unless the Strong Exponential Time Hypothesis fails [1, 7, 22].

Regarding the computational complexity of DTW-MEAN, although more or less implicitly assumed in many publications presenting heuristic solution strategies[4], NP-hardness still has been open (see Brill et al. [6, Section 3] for a discussion on some misconceptions and wrong statements in the literature). It is known to be solvable in $O(n^{2k+1}2^k k)$ time, where n is the maximum length of any input series [6]. Moreover, Brill et al. [6] presented a polynomial-time algorithm for the special case of binary time series which has been improved recently [34]. In practice, numerous heuristics are used [11, 31, 33, 35]. Note that DTW-MEAN is often described as closely related to multiple sequence alignment problems [2, 30, 32]. However, we are not aware of any formal proof regarding this connection. By giving a polynomial-time many-one reduction from MULTIPLE STRING CIRCULAR SHIFT (WITH COST f) to DTW-MEAN, we show that DTW-MEAN is actually connected to multiple *circular* sequence alignment problems. To the best of our knowledge, this is the first formally proven

[4]For instance, Petitjean et al. [31] write "Computational biologists have long known that averaging under time warping is a very complex problem, because it directly maps onto a multiple sequence alignment: the "Holy Grail" of computational biology." Unfortunately, the term "directly maps" has not been formally defined and only sketchy explanations are given.

result regarding this connection.

Our Results. Using plausible complexity-theoretic assumptions, we provide a fine-grained picture of the exact computational complexity (including parameterized complexity) of the problems introduced above. We present two main results.

First, we show that, for a large class of natural cost functions f, f-MSCS on binary sequences is NP-hard, W[1]-hard with respect to the number k of inputs, and not solvable in $\rho(k)\cdot n^{o(k)}$ time for any computable function ρ (unless the Exponential Time Hypothesis fails). Note that f-MSCS is easily solvable in $\rho(k)\cdot n^{O(k)}$ time (for computable functions f) since there are at most n^{k-1} cyclic shifts to try out (without loss of generality, the first string is not shifted). Our running time lower bound thus implies that the brute-force approach can only be improved up to a constant factor in the exponent. Based on this, we can also prove the same hardness for the Circular Consensus String problem. In fact, the general ideas of our reduction might also be used to develop hardness reductions for other circular string alignment problems.

As our second main contribution, we obtain the same list of hardness results as above for DTW-Mean on binary time series. We achieve this by a polynomial-time reduction from a special case of f-MSCS. Our reduction implies that, unless the Exponential Time Hypothesis fails, the known $O(n^{2k+1}2^k k)$-time algorithm [6] essentially can be improved only up to constants in the first exponent. Note that recently Buchin et al. [8] achieved the same hardness result for the problem of averaging time series under generalized (p,q)-DTW. Their reduction, however, does not yield binary input time series.

Organization. In Section 2 we fix notation and introduce basic concepts, also including the formal definition of dynamic time warping and the corresponding concept of warping paths. In Section 3, we identify a circular string problem (of independent interest in molecular biology) which forms the basis for the results in Section 5. More specifically, we prove the hardness results for Multiple String Circular Shift (with Cost f). The key ingredient here is a specially geared polynomial-time reduction from the Regular Multicolored Clique problem. Moreover, we introduce the concept of polynomially bounded grouping functions f (for which our results hold). In Section 4, providing a reduction from Multiple String Circular Shift (with Cost f), we show analogous hardness results for Circular Consensus String. Notably, the cost function corresponding to Circular Consensus String is not a polynomially bounded grouping function, making the direct application of the result for Multiple String Circular Shift (with Cost f) impossible. In Section 5 we prove analogous complexity results for DTW-Mean, again devising a polynomial-time reduction from Multiple String Circular Shift (with Cost f). In Section 6, we conclude with some open questions and directions for future research.

2. Preliminaries.

We briefly introduce our notation and formal definitions.

Circular Shifts. For a string $s = s[1]\dots s[n] \in \Sigma^n$, we denote its length n by $|s|$. For $0 \le \delta < n$, we define the *circular (left) shift by δ* as the string

$$s^{\leftarrow\delta} := s[\delta+1]\dots s[n]s[1]\dots s[\delta] \quad \text{(note that } s^{\leftarrow\delta}[i] = s[(i+\delta-1 \bmod n)+1]),$$

that is, we circularly shift the string δ times to the left. Let $s_1,\dots,s_k$ be strings of length n. A *multiple circular (left) shift* of $s_1,\dots,s_k$ is defined by a k-tuple $\Delta = (\delta_1,\dots,\delta_k) \in \{0,\dots,n-1\}^k$ and yields the strings $s_1^{\leftarrow\delta_1},\dots,s_k^{\leftarrow\delta_k}$. We define *column $i \in \{1,\dots,n\}$* of a multiple circular shift Δ as the k-tuple $(s_1^{\leftarrow\delta_1}[i],\dots,s_k^{\leftarrow\delta_k}[i])$. By *row $j \in \{1,\dots,k\}$* of column i we denote the element $s_j^{\leftarrow\delta_j}[i]$.

Cost Functions. A *local cost function* is a function $f\colon \Sigma^* \to \mathbb{Q}$ assigning a cost to any tuple of values. Given such a function, the *overall cost* of a circular shift Δ for k length-n strings is defined as

$$\mathrm{cost}_f(\Delta) := \sum_{i=1}^{n} f\big((s_1^{\leftarrow\delta_1}[i], \ldots, s_k^{\leftarrow\delta_k}[i])\big),$$

that is, we sum up the local costs of all columns of Δ.

An example for a local cost is the sum of squared distances from the arithmetic mean (i.e., k times the variance, here called σ), that is,

$$\sigma((x_1, \ldots, x_k)) = \sum_{i=1}^{k} \left(x_i - \frac{1}{k} \sum_{j=1}^{k} x_j \right)^2.$$

Using a well-known formula for the variance, we get the following useful formula for σ:

$$\sigma((x_1, \ldots, x_k)) = \left(\sum_{j=1}^{k} x_j^2 \right) - \frac{1}{k} \left(\sum_{j=1}^{k} x_j \right)^2.$$

For binary strings (that is, $x_j \in \{0, 1\}$ for all $1 \le j \le k$), σ does only depend on the number $w := \sum_{j=1}^{k} x_j$ of 1's and the number $k - w$ of 0's and can be written (according to the formula above) as

$$\sigma((x_1, \ldots, x_k)) = w - \frac{w^2}{k} = \frac{w(k-w)}{k}. \tag{2.1}$$

We will repeatedly use this formula later on for cost calculations in the proof for DTW-Mean (Theorem 5.1).

Dynamic Time Warping. A time series is a sequence $x = (x_1, \ldots, x_n) \in \mathbb{Q}^n$. The dynamic time warping distance between two time series is based on the concept of a warping path.

Definition 2.1. *A* warping path *of order $m \times n$ is a sequence $p = (p_1, \ldots, p_L)$, $L \in \mathbb{N}$, of index pairs $p_\ell = (i_\ell, j_\ell) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$, $1 \le \ell \le L$, such that*

(i) $p_1 = (1, 1)$,
(ii) $p_L = (m, n)$, *and*
(iii) $(i_{\ell+1} - i_\ell, j_{\ell+1} - j_\ell) \in \{(1, 0), (0, 1), (1, 1)\}$ *for each* $1 \le \ell \le L - 1$.

See Figure 2 in Section 1 for an example.

The set of all warping paths of order $m \times n$ is denoted by $\mathcal{P}_{m,n}$. A warping path $p \in \mathcal{P}_{m,n}$ defines an *alignment* between two time series $x = (x[1], \ldots, x[m])$ and $y = (y[1], \ldots, y[n])$ in the following way: Every pair $(i, j) \in p$ *aligns* element x_i with y_j. Note that every element from x can be aligned with multiple elements from y, and vice versa. The *dtw-distance* (with squared cost function) between x and y is defined as

$$\mathrm{dtw}(x, y) := \min_{p \in \mathcal{P}_{m,n}} \left(\sum_{(i,j) \in p} (x[i] - y[j])^2 \right)^{1/2}.$$

Note that also other cost functions can be considered. In this work, we only consider the most common case of squared costs.

A *mean* of time series $x_1, \ldots, x_k$ is a time series that minimizes the *Fréchet* function

$$\mathcal{F}(z) := \sum_{j=1}^{k} (\mathrm{dtw}(z, x_j))^2 = \sum_{j=1}^{k} \min_{p_j \in \mathcal{P}_{|x_j|,|z|}} \sum_{(u,v)\in p_j} (x_j[u] - z[v])^2.$$

Note that given, for each $j \in [k]$, a warping path p_j between z and x_j, the value of $z[i]$ that minimizes

$$\sum_{j=1}^{k} \sum_{(u,v)\in p_j} (x_j[u] - z[v])^2 \tag{2.2}$$

is the arithmetic mean of all values aligned to $z[i]$,

$$z[i] = \frac{\sum_{j=1}^{k} \sum_{(i,u)\in p_j} x_j[u]}{\sum_{j=1}^{k} |\{(i,u) \in p_j\}|}.$$

That is, the length of a mean together with the optimal alignments to the input time series determine the mean. The contribution of $z[i]$ to the sum (2.2) is the sum of squared distances between $z[i]$ and all values aligned to $z[i]$,

$$\sum_{j=1}^{k} \sum_{(u,i)\in p_j} (x_j[u] - z[i])^2.$$

Note that this corresponds to the cost function σ above.

We remark that for DTW-Mean, often a normalized cost $F(z) := \frac{1}{k}\mathcal{F}(z)$ is considered. Clearly, this does not affect the computational complexity of the problem, so for simplification purposes we only consider the non-normalized cost $\mathcal{F}(z)$.

Parameterized Complexity. We assume familiarity with the basic concepts from classic and parameterized complexity theory.

An instance of a parameterized problem is a pair (I, k) consisting of the classic problem instance I and a natural number k (the *parameter*). A parameterized problem is contained in the class XP if there is an algorithm solving an instance (I, k) in polynomial time if k is a constant, that is, in time $O(|I|^{f(k)})$ for some computable function f only depending on k (here $|I|$ is the size of I). A parameterized problem is *fixed-parameter tractable* (contained in the class FPT) if it is solvable in time $f(k) \cdot |I|^{O(1)}$ for some computable function f depending solely on k. The class W[1] contains all problems which are parameterized reducible to Clique parameterized by the clique size. A parameterized reduction from a problem Q to a problem P is an algorithm mapping an instance (I, k) of Q in time $f(k) \cdot |I|^{O(1)}$ to an equivalent instance (I', k') of P such that $k' \leq g(k)$ (for some computable functions f and g). It holds that FPT $\subseteq$ W[1] $\subseteq$ XP.

A parameterized problem that is W[1]-hard with respect to a parameter (such as Clique with parameter clique size) is widely believed not to be in FPT.

Exponential Time Hypothesis. Impagliazzo and Paturi [20] formulated the *Exponential Time Hypothesis* (ETH) which asserts that there exists a constant $c > 0$ such that 3-SAT cannot be solved in $O(2^{cn})$ time, where n is the number n of variables in the input formula. It is a stronger assumption than common complexity assumptions such as P$\neq$NP or FPT$\neq$W[1].

Several conditional running time lower bounds have since been shown based on the ETH, for example, Clique cannot be solved in $\rho(k) \cdot n^{o(k)}$ time for any computable function ρ unless the ETH fails [10].

3. Hardness of f-MSCS on Binary Strings. In this section, we focus on binary strings from $\{0,1\}^*$. We prove hardness for a family of local cost functions that satisfy certain properties. The functions we consider have the common property that they only depend on the number of 0's and 1's in a column, and that they aim at grouping similar values together.

DEFINITION 3.1. *A function $f\colon \{0,1\}^* \to \mathbb{Q}$ is called* order-independent *(or* symmetric*) if, for each $k \in \mathbb{N}$, there exists a function $f_k\colon \{0,\dots,k\} \to \mathbb{Q}$ such that $f((x_1,\dots,x_k)) = f_k\big(\sum_{j=1}^{k} x_j\big)$ holds for all $(x_1,\dots,x_k) \in \{0,1\}^k$.*

For an order-independent function f, we define the function $f'_k : \{1,\dots,k\} \to \mathbb{Q}$ as

$$f'_k(x) := \frac{f_k(x) - f_k(0)}{x}.$$

An order-independent function f is grouping *if $f'_k(k) < \min_{1\le x<k} f'_k(x)$ and $f'_k(2) < f'_k(1)$ holds for every $k \in \mathbb{N}$.*

For an order-independent function f, f'_k can be seen as the cost per 1-value (a column with x 1's and $k-x$ 0's has cost $f_k(x) = f_k(0) + x f'_k(x)$). It can also be seen as a discrete version of the derivative for f_k, so that if f_k is concave then f'_k is decreasing. The intuition behind a grouping function is that the cost per 1-value is minimal in columns containing only 1's, and that having two 1's in a column has less cost than having two columns with a single 1. In particular, any function f where all f_k are strictly concave is grouping. Finally, if f is grouping, then the function

$$(x_1,\dots,x_k) \mapsto f_k\left(\sum_{j=1}^{k} x_j\right) + a\sum_{j=1}^{k} x_j + b$$

is also grouping for any $a, b \in \mathbb{Q}$.

The following definitions are required to ensure that our subsequent reduction (Lemma 3.3) is computable in polynomial time.

DEFINITION 3.2. *Let f be an order-independent function. The* gap *of f_k is*

$$\varepsilon_k := \min\{f'_k(x) - f'_k(y) \mid 1 \le x, y \le k, f'_k(x) > f'_k(y)\}.$$

The range *of f_k is $\mu_k := \max_{1\le x\le k} |f'_k(x)|$.*

An order-independent function f is polynomially bounded *if it is polynomial-time computable and if, for every $k \in \mathbb{N}$, μ_k and ε_k^{-1} are upper-bounded by a polynomial in k.*

For binary strings, the function σ (see Section 2) is a polynomially bounded grouping function. Indeed, it is order-independent since $\sigma((x_1,\dots,x_k)) = \frac{w(k-w)}{k}$, where $w = \sum_{j=1}^{k} x_j$. Thus, $\sigma_k(w) = \frac{w(k-w)}{k}$ and we have $\sigma_k(0) = 0$, and $\sigma'_k(w) = \frac{k-w}{k}$, so σ'_k is strictly decreasing, which is sufficient for σ to be grouping. Finally, it is polynomially bounded, with gap $\varepsilon_k = \frac{1}{k}$ and range $\mu_k = \frac{k-1}{k} \le 1$.

We prove our hardness results with a polynomial-time reduction from a special version of the CLIQUE problem.

REGULAR MULTICOLORED CLIQUE (RMCC)
Input: A d-regular undirected graph $G = (V, E)$ where the vertices are colored with k colors such that each color class contains the same number of vertices.
Question: Does G have a size-k complete subgraph (containing $\binom{k}{2}$ edges, called a k-clique) with exactly one vertex from each color?

RMCC is known to be NP-hard, W[1]-hard with respect to k, and not solvable in $\rho(k) \cdot |V|^{o(k)}$ time for any computable function ρ unless the ETH fails [12].

The following lemma states the existence of a polynomial-time reduction from RMCC to f-MSCS which implies hardness of f-MSCS for polynomially bounded grouping functions.

LEMMA 3.3. *Let f be a polynomially bounded grouping function. Then there is a polynomial-time reduction that, given an* RMCC *instance $G = (V, E)$ with k colors, outputs binary strings $s_0, \dots, s_k$ of equal length and $c \in \mathbb{Q}$ such that the following holds:*

- *If G contains a properly colored k-clique, then there exists a multiple circular shift Δ of $s_0, \dots, s_k$ with $\mathrm{cost}_f(\Delta) = c$.*
- *If G does not contain a properly colored k-clique, then every multiple circular shift Δ of $s_0, \dots, s_k$ has $\mathrm{cost}_f(\Delta) \geq c + \varepsilon_{k+1}$.*

To prove Lemma 3.3, we first describe the reduction and then prove several claims about the structure and the costs of multiple circular shifts in the resulting f-MSCS instance.

Reduction. Consider an instance of RMCC, that is, a graph $G = (V, E)$ with a partition of V into k subsets $V_1, \dots, V_k$ of size $n := \frac{|V|}{k}$ each, such that each vertex has degree d. Let $V_j = \{v_{j,1}, \dots, v_{j,n}\}$, $m = |E|$, and $E = \{e_1, \dots, e_m\}$. We assume that $k \geq 3$ since the instance is trivially solvable otherwise.

We build an f-MSCS instance with $k+1$ binary strings. Hence, the local cost of a column of a multiple circular shift is given by the function f_{k+1}. For simplicity, we write f', gap ε, and range μ for f'_{k+1}, ε_{k+1}, and μ_{k+1}.

For each $j \in \{1, \dots, k\}$, let p_j be the length-k string such that $p_j[h] = 1$ if $h = j$, and $p_j[h] = 0$ otherwise. For each vertex $v_{j,i}$, let $q_{j,i} \in \{0,1\}^m$ be the string such that

$$q_{j,i}[h] := \begin{cases} 1, & \text{if } 1 \leq h \leq m \text{ and } v_{j,i} \in e_h \\ 0, & \text{otherwise} \end{cases}$$

and let $u_{j,i} := p_j q_{j,i}$ be the concatenation of p_j and $q_{j,i}$. Note that $u_{j,i}$ has length $m' := m + k$, contains $1 + d$ ones, and $m' - 1 - d$ zeros. Let $\boldsymbol{0} := 0^{m'}$ be the string containing m' zeros and define the numbers

$$\kappa := knd + kn + k,$$
$$\gamma := nk,$$
$$\lambda := \max\left\{ \left\lceil \kappa \left(\frac{2\mu}{\varepsilon} + 1 \right) \right\rceil, 2n(\gamma + k + 1) \right\} + 1.$$

For $1 \leq j \leq k$, we define the string

$$s_j := 1u_{j,1}(1\boldsymbol{0})^{\gamma+j} 1u_{j,2}(1\boldsymbol{0})^{\gamma+j} \dots 1u_{j,n}(1\boldsymbol{0})^{\gamma+j}(1\boldsymbol{0})^{\lambda - n(\gamma+j+1)}.$$

Note that $|1u_{j,i}| = |1\boldsymbol{0}| = m' + 1$. Thus, each string s_j has length

$$n(m'+1)(1+\gamma+j) + (m'+1)(\lambda - n(\gamma+j+1)) = \lambda(m'+1) =: \ell,$$

where $\ell \leq \text{poly}(nk)$. We further define the following length-ℓ *dummy* string

$$s_0 = 11^k 0^m (1\mathbf{0})^{\lambda-1}.$$

Finally, we define the target cost

$$\begin{aligned} c :=& \ell f_{k+1}(0) \\ &+ \lambda(k+1) f'(k+1) \\ &+ 2\left(k + \binom{k}{2}\right)(f'(2) - f'(1)) \\ &+ \kappa f'(1). \end{aligned}$$

Clearly, the strings $s_0, \ldots, s_k$ and the value c can be computed in polynomial time. Our construction is illustrated in Figure 3.

In the strings $s_0, \ldots, s_k$, any 1-value at a position i with $i \bmod (m'+1) = 1$ is called a *separator*, other 1-values are *coding* values. A coding value is either *vertex-coding* if it belongs to some p_j (or to the k coding values of s_0), or *edge-coding* otherwise (then it belongs to some $q_{i,j}$). There are $\lambda(k+1)$ separator values in total and κ coding values.

Given a multiple circular shift Δ, we define the *weight* w of a column as the number of 1-values it contains, that is, $0 \leq w \leq k+1$. The cost for such column is $f_{k+1}(w) = f_{k+1}(0) + wf'(w)$. Each 1-value of this column is attributed a *local cost* of $f'(w)$, so that the cost of any solution is composed of a *base cost* of $\ell f_{k+1}(0)$ and of the sum of all local costs of all 1-values. In the following we mainly focus on local costs.

It remains to be shown that there exists a multiple circular shift of $s_0, \ldots, s_k$ with cost c if G contains a properly colored k-clique, and that otherwise every multiple circular shift has cost at least $c + \varepsilon$. We proceed by analyzing the structure and costs of optimal multiple circular shifts.

Aligning Separators. Let $\Delta = (\delta_0, \ldots, \delta_k)$ be a multiple circular shift of $s_0, \ldots, s_k$. Without loss of generality, we can assume that $\delta_0 = 0$ since setting each δ_j to $(\delta_j - \delta_0) \bmod \ell$ yields a shift with the same cost. First, we show that if $\delta_j \bmod (m'+1) \neq 0$ holds for some $0 < j \leq k$, then Δ has large cost.

CLAIM 3.4. *For any multiple circular shift $\Delta = (\delta_0 = 0, \delta_1, \ldots, \delta_k)$ with $\delta_j \bmod (m'+1) \neq 0$ for some $1 < j \leq k$, it holds that $\text{cost}_f(\Delta) \geq c + \varepsilon$.*

Proof. Assume that $\delta_j \bmod (m'+1) = a \in \{1, \ldots, m'\}$ for some $0 < j \leq k$. We count the number of weight-$(k+1)$ columns: such a column cannot only contain separator values since it cannot contain a separator value in both row 0 and row j. Hence, it contains at least one coding value. Since there are κ coding values, there are at most κ weight-$(k+1)$ columns, so at most $k\kappa$ separator values have local cost $f'(k+1)$. All other separator values have local cost $f'(w)$ for some $w < k+1$, which is at least $f'(k+1)+\varepsilon$. There are at least $\lambda(k+1) - k\kappa$ such separator values. Adding the base cost of $\ell f_{k+1}(0)$, the cost of Δ is thus at least:

$$\begin{aligned} \text{cost}_f(\Delta) &\geq \ell f_{k+1}(0) + (\lambda(k+1) - k\kappa)(f'(k+1) + \varepsilon) \\ &\geq \ell f_{k+1}(0) + \lambda(k+1) f'(k+1) + \lambda k\varepsilon - k\kappa(\mu + \varepsilon). \end{aligned}$$

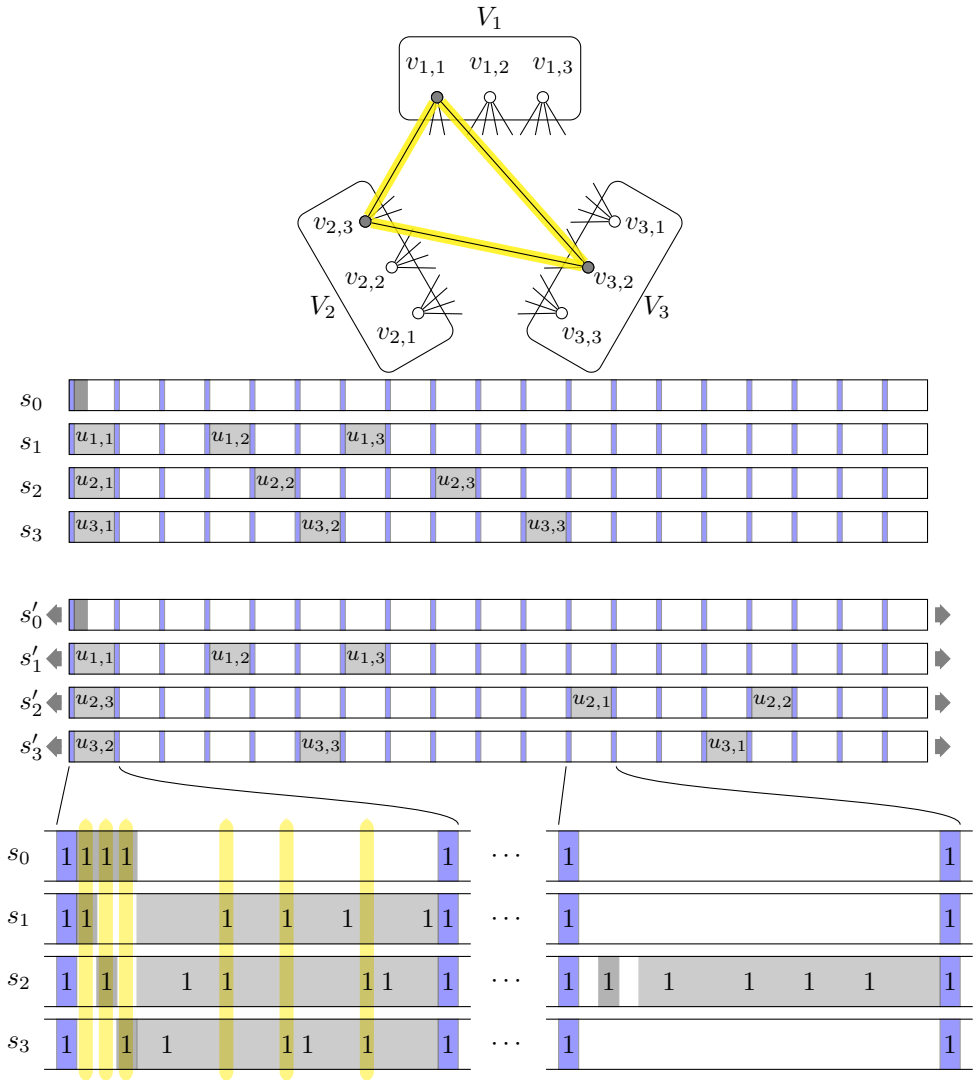

FIG. 3. *Illustration of the reduction from an instance of* RMCC *(top) with* $k = 3$. *Middle: Sequences* s_0 *to* s_3, *and their optimal circular shifts* s'_0 *to* s'_3. *Blue stripes represent the regularly-spaced separator 1-values. The (light) gray intervals contain both* 0*'s and* 1*'s according to strings* $u_{i,j}$, *and white intervals contain only 0's. The spacing between consecutive* $u_{i,j}$*'s is defined using* γ *and the overall string length depends on* λ, *both values are chosen so as to restrict the possible alignments between different* $u_{i,j}$*'s; in this example we use* $\gamma = 1$ *and* $\lambda = 19$. *Bottom: a zoom-in on blocks 1 and 12 in the shifted strings (only non-0 values are indicated, weight-2 columns are highlighted). Through vertex columns, the dummy string* s_0 *ensures that one vertex occupies block 1 in each row, and weight-2 edge-columns ensure that* $\binom{k}{2}$ *edges (as highlighted in the graph) are induced by these vertices.*

Recall that

$$c = \ell f_{k+1}(0) + \lambda(k+1)f'(k+1) + 2\left(k + \binom{k}{2}\right)(f'(2) - f'(1)) + \kappa f'(1)$$
$$\leq \ell f_{k+1}(0) + \lambda(k+1)f'(k+1) + \kappa\mu$$

since $f'(2) - f'(1) < 0$. Combining the above bounds for c and $\text{cost}_f(\Delta)$ using $\lambda \geq \kappa\left(\frac{2\mu}{\varepsilon} + 1\right) + 1$ (by definition) yields

$$\begin{aligned}\text{cost}_f(\Delta) - c &\geq \lambda k\varepsilon - k\kappa(\mu+\varepsilon) - \kappa\mu \\ &\geq 2\kappa k\mu + \kappa k\varepsilon + k\varepsilon - k\kappa(\mu+\varepsilon) - \kappa\mu \\ &\geq \varepsilon. \end{aligned}$$

∎

Cost of Circular Shifts. We assume from now on that $\delta_j \bmod (m'+1) = 0$ for all $j \in \{0, \ldots, k\}$. We now provide a precise characterization of the cost of Δ.

For $l \in \{1, \ldots, \lambda\}$, we define the *$l$-th block* consisting of the m' consecutive columns $(l-1)(m'+1)+2, \ldots, l(m'+1)$. The *block index* of column i is $i-1 \bmod (m'+1)$. For $j \in \{1, \ldots, k\}$, the substring $s_j^{\leftarrow\delta_j}[(l-1)(m'+1)+2] \ldots s_j^{\leftarrow\delta_j}[l(m'+1)]$ corresponding to the l-th block of $s_j^{\leftarrow\delta_j}$ either equals some $u_{j,i}$ or $\boldsymbol{0}$. We say that block l is *occupied* by vertex $v_{j,i} \in V_j$, if the corresponding substring of $s_j^{\leftarrow\delta_j}$ is $u_{j,i}$. Note that for each j there are n distinct blocks out of λ that are occupied by a vertex in V_j. Columns with block-index 1 to k are called *vertex-columns* and columns with block-index $k+1$ to $k+m = m'$ are *edge-columns* (they may only contain edge-coding values from some $q_{i,j}$). Let P denote the set of vertices occupying block 1.

OBSERVATION 3.5. *In block l, if the vertex-column with block-index h has weight 2, then $l = 1$, and $V_h \cap P \neq \emptyset$. No vertex-column can have weight 3 or more.*

Proof. Consider the vertex-column with block-index h. By construction, among $s_1, \ldots, s_k$ only s_h may have a 1 in this column (which is true if some vertex from V_h occupies this block). The string s_0 has a 1 in this column if it is a column in block 1. Thus, assuming that column h has weight 2 implies $l = 1$ and $V_h \cap P \neq \emptyset$. ∎

OBSERVATION 3.6. *In block l, if the edge-column with block-index $k+h$, $1 \leq h \leq m$, has weight 2, then block l is occupied by both vertices of edge $e_h \in E$. No edge-column can have weight 3 or more.*

Proof. Consider an edge-column with block-index $k+h$, $1 \leq h \leq m$. Denote by v_{j_0,i_0} and v_{j_1,i_1} the endpoints of edge e_h. For any $1 \leq j \leq k$, s_j has a 1 in this column only if block l is occupied by some vertex $v_{j,i}$, and, moreover, only if $u_{j,i}$ has a 1 in column h, i.e., $v_{j,i} = v_{j_0,i_0}$ or $v_{j,i} = v_{j_1,i_1}$, hence $j = j_1$ or $j = j_2$. So this column may not have weight 3 or more, and if it has weight 2, then block l is occupied by both endpoints of e_h. ∎

From Observations 3.5 and 3.6 it follows that no column (beside separators) can have weight 3 or more. Since the number of coding values is fixed, the cost is entirely determined by the number of weight-2 columns. In the following, we will first give an upper bound on the number of weight-2 columns and then analyze how it determines the cost.

Observation 3.5 gives a direct upper bound of at most k weight-2 vertex columns (since they all are in block 1). We now focus on weight-2 edge-columns. The following claim will help us to show an upper bound on their number.

CLAIM 3.7. *For any two rows j, j', there exists at most one block l that is occupied by vertices from both V_j and $V_{j'}$.*

Proof. First, note that if two distinct blocks l and l' are both occupied by a vertex from the same row j, then, by construction, there are two possible cases: either $|l-l'| = a(\gamma+j+1)$ or $|l-l'| = \lambda - a(\gamma+j+1)$, where $1 \leq a < n$ in both cases.

Indeed, there are n regularly-spaced substrings $u_{j,i}$ (having $\gamma + j$ blocks in between them) in s_j (consisting of λ blocks in total).

Assume towards a contradiction that two distinct blocks l and l' are each occupied by a vertex from V_j and $V_{j'}$. Then, there exists an $a \in \{1, \dots, n-1\}$ such that $|l-l'| = a(\gamma+j+1)$ or $|l-l'| = \lambda - a(\gamma+j+1)$, and there exists an $a' \in \{1, \dots, n-1\}$ such that $|l-l'| = a'(\gamma+j'+1)$ or $|l-l'| = \lambda - a'(\gamma+j'+1)$. This gives four cases to consider (in fact just three by symmetry of j and j').

If $|l-l'| = a(\gamma+j+1) = a'(\gamma+j'+1)$, then $(a-a')(\gamma+1) = a'j' - aj$. We have $a \neq a'$, as otherwise this would imply $j = j'$. So $|a'j' - aj| \geq \gamma + 1$, but this is impossible since $a < n$, $a' < n$, $j \leq k$, $j' \leq k$, that is, $|a'j' - aj| \leq kn$, whereas $\gamma + 1 = kn + 1$ (by construction).

If $|l-l'| = a(\gamma+j+1) = \lambda - a'(\gamma+j'+1)$, then $\lambda = a(\gamma+j+1) + a'(\gamma+j'+1)$. However, $\lambda > 2n(\gamma+k+1)$ by construction, so this case also leads to a contradiction.

Finally, if $|l-l'| = \lambda - a(\gamma+j+1) = \lambda - a'(\gamma+j'+1)$, then we have $a(\gamma+j+1) = a'(\gamma+j'+1)$. This case yields, as the first case, a contradiction. ∎

Claim 3.8. *There are at most $\binom{k}{2}$ weight-2 edge-columns.*

Proof. Consider any pair j, j' such that $1 \leq j < j' \leq k$. It suffices to show that there exists at most one weight-2 edge-column with a 1 in rows j and j'. Aiming at a contradiction, assume that two such columns exist. By Observation 3.6, they must each belong to a block which is occupied by vertices both in V_j and $V_{j'}$. From Claim 3.7 it follows that both columns belong to the same block. Let v and v' be the vertices of V_j and $V_{j'}$, respectively, occupying this block. By Observation 3.6 again, both edges are equal to $\{v, v'\}$, which contradicts the fact that they are distinct. ∎

Having established an upper bound of $k + \binom{k}{2}$ for the number of weight-2 columns, the following result describes the corresponding cost.

Claim 3.9. *Let W_2 be the number of weight-2 columns. If $W_2 = k + \binom{k}{2}$, then $\mathrm{cost}_f(\Delta) = c$. If $W_2 < k + \binom{k}{2}$, then $\mathrm{cost}_f(\Delta) \geq c + \varepsilon$.*

Proof. The base cost $\ell f_{k+1}(0)$ of the solution only depends on the number ℓ of columns. Separator values are in weight-$(k+1)$ columns. Since there are λ of them, it follows that the total local cost of all separator values is $\lambda(k+1)f'(k+1)$.

The total number of coding values is κ, each coding value has a local weight of $f'(1)$ if it belongs to a weight-1 column, and $f'(2)$ otherwise (since there is no vertex- or edge-column with weight 3 or more). There are W_2 weight-2 columns, so exactly $2W_2$ coding values within weight-2 columns. Summing the base cost with the local costs of all separator and coding values, we get:

$$\begin{aligned}\mathrm{cost}_f(\Delta) = \ell f_{k+1}(0) &\\ + \lambda(k+1)f'(k+1) &\\ + 2W_2(f'(2) - f'(1)) &\\ + \kappa f'(1). &\end{aligned}$$

Thus, by definition of c, we have $\mathrm{cost}_f(\Delta) = c$ if $W_2 = k + \binom{k}{2}$. If $W_2 < k + \binom{k}{2}$, then using the fact that, by assumption, $f'(2) - f'(1) \leq -\varepsilon$, we obtain

$$\mathrm{cost}_f(\Delta) = c + 2\left(W_2 - k - \binom{k}{2}\right)(f'(2) - f'(1)) \geq c + \varepsilon.$$

∎

CLAIM 3.10. *If G does not contain a properly colored k-clique, then there are at most $k + \binom{k}{2} - 1$ weight-2 columns.*

Proof. We prove the contrapositive. Assume that there are at least $k+\binom{k}{2}$ weight-2 columns. By Claim 3.8, there are at least k weight-2 vertex-columns. By Observation 3.5, only the k vertex-columns of block 1 may have weight 2, hence for each $1 \leq j \leq k$ the column of block 1 with block-index j has weight 2. Thus, for every j, $P \cap V_j \neq \emptyset$.

By Claim 3.7, no other block than block 1 may be occupied by two vertices, hence any edge-column with weight 2 must be in block 1, and both endpoints are in P. There cannot be more than k weight-2 vertex-columns, hence there are $\binom{k}{2}$ weight-2 edge-columns, and for each of these there exists a distinct edge with both endpoints in P. Thus, P is a properly colored k-clique. ∎

Cliques and Circular Shifts with Low Cost. We are now ready to complete the proof of Lemma 3.3. First, assume that G contains a properly colored k-clique $P = \{v_{1,i_1}, \ldots, v_{k,i_k}\}$. Consider the multiple circular shift $\Delta = (\delta_0, \ldots, \delta_k)$, where $\delta_0 = 0$ and

$$\delta_j := (i_j - 1)(m' + 1)(\gamma + j + 1)$$

for $j \in \{1, \ldots, k\}$. Note that $|P| = k$, and all edge-columns in block 1 corresponding to edges induced in P have weight 2. Hence there are $\binom{k}{2}$ weight-2 edge-columns and k weight-2 vertex-columns. By Claim 3.9, $\mathrm{cost}_f(\Delta) = c$.

Now, assume that G does not contain a properly colored k-clique. Let $\Delta = (\delta_0, \ldots, \delta_k)$ be a multiple circular shift with $\delta_0 = 0$ (recall that we can assume this without loss of generality). Clearly, if $\delta_j \bmod (m'+1) \neq 0$ for some j, then $\mathrm{cost}_f(\Delta) \geq c + \varepsilon$ (by Claim 3.4). Otherwise, by Claim 3.10 there are at most $k + \binom{k}{2} - 1$ weight-2 columns. By Claim 3.9 , $\mathrm{cost}_f(\Delta) \geq c + \varepsilon$.

This completes the proof of Lemma 3.3 which directly leads to our main result of this section.

THEOREM 3.11. *Let f be a polynomially bounded grouping function. Then, f-*MSCS *on binary strings is*

(i) NP-hard,
(ii) W[1]-hard with respect to the number k of input strings, and
(iii) not solvable in $\rho(k) \cdot n^{o(k)}$ time for any computable function ρ unless the ETH fails.

Proof. The polynomial-time reduction from Lemma 3.3 yields the NP-hardness. Moreover, the number of strings in the f-MSCS instance only depends on the size of the multicolored clique. Hence, it is a parameterized reduction from RMCC parameterized by the size of the clique to f-MSCS parameterized by the number of input strings and thus yields W[1]-hardness. Lastly, the number $k' = k + 1$ of strings is linear in the size k of the clique. Thus, any $\rho(k') \cdot n^{o(k')}$-time algorithm for DTW-MEAN would imply a $\rho'(k) \cdot |V|^{o(k)}$-time algorithm for RMCC, contradicting the ETH. ∎

Note that Theorem 3.11 holds for the function σ since it is a polynomially bounded grouping function (as discussed earlier).

The assumption that f is polynomially bounded is only needed to obtain a polynomial-time reduction in Lemma 3.3. Without this assumption, we still obtain a parameterized reduction from RMCC parameterized by the clique size to f-MSCS parameterized by the number of input strings, which yields the following corollary for a larger class of functions.

COROLLARY 3.12. *Let f be a computable grouping function. Then, f-MSCS on binary strings is W[1]-hard with respect to the number k of input strings and not solvable in $\rho(k) \cdot n^{o(k)}$ time for any computable function ρ unless the ETH fails.*

4. Circular Consensus String. In this section we briefly study the CIRCULAR CONSENSUS STRING problem:

CIRCULAR CONSENSUS STRING (SCC)
Input: A list of k strings $s_1, \ldots, s_k \in \Sigma^n$ of length n and $c \in \mathbb{Q}$.
Question: Is there a string $s^* \in \Sigma^n$ and a multiple circular shift $(\delta_1, \ldots, \delta_k)$ such that $\sum_{j=1}^{k} d(s_j^{\leftarrow \delta_j}, s^*) \leq c$?

Here, d denotes the Hamming distance, that is, the number of mismatches between the positions of two strings. Although consensus string problems in general have been widely studied from a theoretical point of view [9], somewhat surprisingly this is not the case for the circular version(s). For CCS, only an $O(n^2 \log n)$-time algorithm for $k = 3$ and an $O(n^3 \log n)$-time algorithm for $k = 4$ is known [23]. However, for general k no hardness result is known. Note that without circular shifts the problem is solvable in linear time: It is optimal to set $s^*[i]$ to any element that appears a maximum number of times among the elements $s_1[i], \ldots, s_k[i]$.

For binary strings, it can easily be seen that the cost induced by column i is the minimum of the number of 0's and the number of 1's. Let f^{cs} be the polynomially bounded order-independent function with $f_k^{\mathsf{cs}}(w) = \min\{w, k - w\}$. It follows from the discussion above that CIRCULAR CONSENSUS STRING is exactly f^{cs}-MSCS. Note, however, that f^{cs} is not a grouping function since $f_k^{\mathsf{cs}\prime}(2) = f_k^{\mathsf{cs}\prime}(1) = 1$. That is, we do not immediately obtain hardness of CCS from Theorem 3.11. We can still prove hardness via a reduction using a properly chosen polynomially bounded grouping function.

THEOREM 4.1. CIRCULAR CONSENSUS STRING *on binary strings is*
(i) NP-hard,
(ii) W[1]-hard with respect to the number k of input strings, and
(iii) not solvable in $\rho(k) \cdot n^{o(k)}$ time for any computable function ρ unless the ETH fails.

Proof. As discussed above, CCS is equivalent to f^{cs}-MSCS. To prove hardness, we define a local cost function g (similar to f^{cs}) and reduce from g-MSCS to f^{cs}-MSCS.

Let g be the order-independent local cost function such that

$$g_k(w) := f_{2k-2}^{\mathsf{cs}}(w + (k-2)) = \min\{w + k - 2, k - w\}.$$

Note that the function g_k is linearly decreasing on $\{1, \ldots, k\}$ and that $g_k'(w) = \frac{2-w}{w} = \frac{2}{w} - 1$. The range of g_k is $\mu_k = 1$ and its gap is $\varepsilon_k = \frac{2}{k-1} - \frac{2}{k} > \frac{2}{k^2}$. That is, g satisfies all conditions of Theorem 3.11 and the corresponding hardness results hold for g-MSCS. See Figure 4 for an illustration.

Given an instance $\mathcal{I} = (s_1, \ldots, s_k, c)$ of g-MSCS, we define the strings $s_j := 1^{|s_1|}$ for $j = k+1, \ldots, 2k-2$. We show that $\mathcal{I}$ is a yes-instance if and only if $\mathcal{I}' := (s_1, \ldots, s_{2k-2}, c)$ is a yes-instance for f^{cs}-MSCS.

For the forward direction, consider a multiple circular shift $\Delta = (\delta_1, \ldots, \delta_k)$ of $s_1, \ldots, s_k$ such that $\mathrm{cost}_g(\Delta) \leq c$. We define the multiple circular shift $\Delta' := (\delta_1, \ldots, \delta_k, \delta_{k+1} = 0, \ldots, \delta_{2k-2} = 0)$ of $s_1, \ldots, s_{2k-2}$. Consider column i of Δ' and let w' be the number of 1's it contains. Then, $w' = w + k - 2$, where w is the number

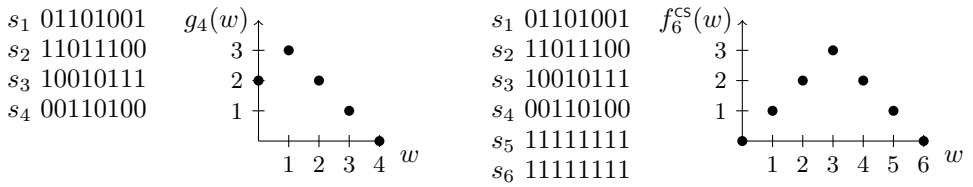

FIG. 4. *Reduction from an instance of g-*MSCS *(left) to an instance of* f^{cs}*-*MSCS*, which is equivalent to the* CIRCULAR CONSENSUS STRING *problem. Plots of the (polynomially bounded and order-independent) local cost functions for* $k = 4$ *are shown. Note that* g_4 *is obtained from* f_6^{cs} *by cropping the first two values in order to become grouping.*

of 1's in column i of Δ. The cost of column i is $f^{\mathsf{cs}}_{2k-2}(w+k-2) = g_k(w)$. Hence, column i has the same cost in both solutions. This implies $\mathrm{cost}_g(\Delta) = \mathrm{cost}_{f^{\mathsf{cs}}}(\Delta')$.

The converse direction is similar. Any multiple circular shift Δ' of $s_1, \ldots, s_{2k-2}$ can be restricted to a multiple circular shift Δ of $s_1, \ldots, s_k$ with the same cost. ∎

5. Consensus for Time Series: DTW-Mean. In this section we consider the DTW-MEAN problem.

DTW-MEAN
Input: A list of k univariate rational time series $x_1, \ldots, x_k$ and $c \in \mathbb{Q}$.
Question: Is there a univariate rational time series z such that $\mathcal{F}(z) = \sum_{i=1}^{k} (\mathrm{dtw}(z, x_i))^2 \leq c$?

We prove the following theorem, settling the complexity status of this prominent consensus problem in time series analysis.

THEOREM 5.1. DTW-MEAN *on binary time series is*
(i) NP-hard,
(ii) W[1]-hard with respect to the number k of input series, and
(iii) not solvable in $\rho(k) \cdot n^{o(k)}$ *time for any computable function* ρ *unless the ETH fails.*

The proof is based on a polynomial-time reduction from a special variant of f-MSCS for which hardness holds via Theorem 3.11 in Section 3. At this point we make crucial use of the fact that the reduction described in the proof of Lemma 3.3 actually shows that it is hard to decide whether there is a multiple circular shift of cost at most c or whether all multiple circular shifts have cost at least $c+\varepsilon$ for some ε polynomially bounded in the number of strings. This gap of ε guarantees that a no-instance of f-MSCS is reduced to a no-instance of DTW-MEAN. Being polynomially bounded is required for ε in order to obtain a polynomial-time reduction (otherwise our constructed time series are too long).

Before proving Theorem 5.1, we introduce some definitions. A *position* i in a time series x is an integer $1 \leq i \leq |x|$, its *value* is $x[i]$. The *distance* between two positions i and i' is $|i'-i|$. A *block* in a binary time series is a maximal subseries of consecutive 0's (a *0-block*) or 1's (a *1-block*). Blocks are also represented by integers, indicating their rank in the series (a series with n blocks has blocks $1, 2, \ldots, n$). The *distance* between two blocks of rank y and y' is $|y' - y|$. Note that the notion of distance is different in the context of positions and blocks (even between size-1 blocks, as larger blocks in between increase the position distance).

Recall from Section 2 that once the length of a mean z and the alignments to

the input time series are known, then its values are determined. In the binary case, a position i that is aligned to a 0's and b 1's contributes a cost of $az[i]^2 + b(1-z[i])^2$ to $\mathcal{F}(z)$. This cost is minimal for $z[i]$ being the arithmetic mean $b/(a+b)$ of all values aligned to position i. Thus, the *cost* of position i is

$$C(i) = \min_{x \in \mathbb{Q}} (ax^2 + b(1-x)^2) = \frac{ab}{a+b},$$

where the second equality follows from Equation (2.1). The reduction in the following proof of Theorem 5.1 is from ϕ-MSCS for a polynomially bounded grouping function ϕ specifically chosen in relation to the above cost of a mean position.

Proof. We will reduce from ϕ-MSCS for a specially chosen cost function ϕ which we explain first. To this end, we briefly sketch the idea of the reduction. Given k binary strings for ϕ-MSCS, we construct $k+1$ binary time series: k time series encoding the original strings and a dummy time series containing a single 1. The construction is such that a column of a multiple circular shift with x 1's and $k-x$ 0's will correspond to a position in a time series that is aligned to $k+x$ 0's and one 1. Now, the cost of that column should be equal to the cost $(k+x)/(k+x+1)$ of that position minus $k/(k+1)$ (for technical reasons the cost should be 0 if $x = 0$). That is, we define $\phi\colon \{0,1\}^* \to [0,1]$ with $\phi((x_1, \dots, x_k)) = \phi_k(\sum_{j=1}^{k} x_j)$, where $\phi_k\colon \{0, \dots, k\} \to [0,1]$ with

$$\phi_k(x) = \frac{k+x}{k+x+1} - \frac{k}{k+1} = \frac{x}{(k+x+1)(k+1)}.$$

Note that ϕ is a polynomially bounded grouping function since

$$\phi_k'(x) = \frac{\phi_k(x) - \phi_k(0)}{x} = \frac{1}{(k+x+1)(k+1)}$$

is strictly decreasing (ϕ_k is strictly concave) with gap

$$\varepsilon_k = \phi_k'(k-1) - \phi_k'(k) = \frac{1}{(k+1)(2k)(2k+1)}$$

and range

$$\mu_k = \phi_k'(1) = \frac{1}{(k+2)(k+1)}.$$

See Figure 5 for an example of the functions ϕ_k and ϕ_k'. Hence, by Theorem 3.11 hardness holds for ϕ-MSCS. We now give the polynomial-time reduction from ϕ-MSCS, see Figure 6 for an illustration of the reduction.

Reduction. In the following, we assume to have an instance of ϕ-MSCS with $k \geq 15$ length-n binary strings $s_1, \dots, s_k \in \{A, B\}^n$ (where $A := 0$ and $B := 1$) and a target cost $0 \leq c \leq n$. We write ε for the gap ε_k of ϕ_k (note that $\varepsilon^{-1} \in O(k^3)$). The task is to decide whether there exists a multiple circular shift of cost at most c or whether all multiple circular shifts have cost at least $c+\varepsilon$ (the reduction in Lemma 3.3 implies the hardness of this decision problem).

First, we encode characters A and B via certain binary strings. To this end, we define the number $m := 1600k\lceil c + \varepsilon \rceil$ and the binary strings $t_A := (10)^m$ and $t_B := 100(10)^{m-1}$, each string having m 0-blocks (all of length one, except for the first 0-block of t_B which has length two). The first 0-block of t_A and t_B is called a *coding* block (respectively, an A-coding or a B-coding block).

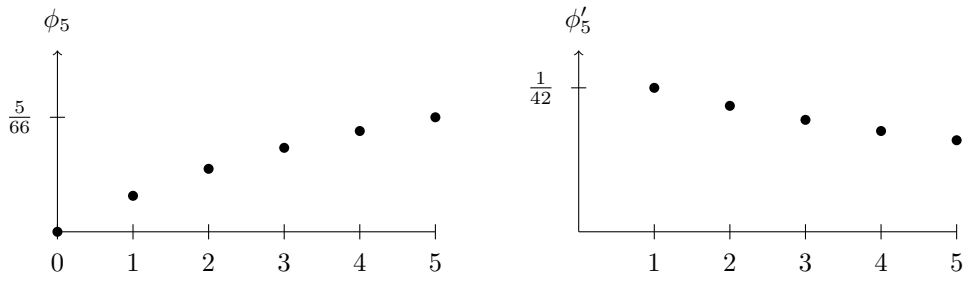

FIG. 5. *Left: The function* $\phi_5(x) = \frac{x}{6(6+x)}$. *Right: The function* $\phi'_5(x) = \frac{1}{6(6+x)}$.

Now, for each $1 \leq j \leq k$, let s'_j be the string obtained by concatenating the strings $t_{s_j[i]}$ for $1 \leq i \leq |s_j|$. The final time series x_j is obtained by concatenating r copies of s'_j, where

$$r \coloneqq \left\lceil \frac{1}{\varepsilon}(3mnk + 2(c+\varepsilon)) \right\rceil + 1.$$

Note that each x_j contains $2mnr$ blocks and $|x_j| \leq \text{poly}(nk)$. We also define the *extra* series $x_{k+1} = (1)$ and set the target cost to

$$c' \coloneqq r\left(c + \frac{mnk}{k+1}\right) + 3mnk.$$

For the correctness of this reduction we need to prove the following:

(i) If $(s_1, \ldots, s_k, c)$ is a yes-instance, that is, there exists a multiple circular shift Δ with $\text{cost}_\phi(\Delta) \leq c$, then there exists a time series z with $\mathcal{F}(z) \leq c'$.

(ii) If $(s_1, \ldots, s_k, c)$ is a no-instance, that is, $\text{cost}_\phi(\Delta) \geq c+\varepsilon$ holds for every multiple circular shift Δ, then $\mathcal{F}(z) > c'$ holds for every time series z.

(i) Yes-instance of ϕ-MSCS. Consider a multiple circular shift $\Delta = (\delta_1, \ldots, \delta_k)$ of $s_1, \ldots, s_k$ with $\text{cost}_\phi(\Delta) \leq c$. Without loss of generality, we assume that $0 \leq \delta_j < n$ holds for every $1 \leq j \leq k$.

We construct a time series z of length $2mn(r-1)+2$ (also see Figure 6) such that $\mathcal{F}(z) \leq c'$. To this end, we describe the alignments between z and $x_1, \ldots, x_{k+1}$ (recall that this determines the values and costs of positions in z). For each $j = 1, \ldots, k$, the first position is aligned with the leftmost $2\delta_j m$ blocks of each x_j (or with the first block if $\delta_j = 0$) and the last position is aligned with the rightmost $2(n-\delta_j)m$ blocks of x_j. For each $1 < i < 2mn(r-1)+2$, position i is aligned with the $(i-1+2\delta_j m)$-th block of x_j. These positions are called *regular* positions, whereas the first and last position are called *extreme*. Clearly, all positions of z are also aligned with the single 1 in x_{k+1}.

Given the above alignments, the sum of costs of all positions in z is clearly an upper bound for $\mathcal{F}(z)$. The following two claims give an upper bound for this sum.

CLAIM 5.2. *The total cost of regular positions is at most* $(r-1)(c + \frac{mnk}{k+1})$.

Proof. Due to the alternation of 1- and 0-blocks in each x_j and the fact that $i + (2\delta_j m) \equiv i \pmod 2$, it follows that the i-th regular position (which is $z[i+1]$) is mapped only to 1's if i is odd (*odd* position) or only to 0's (and the single 1 in x_{k+1}) if i is even (*even* positions). Thus, odd positions have cost $\frac{(k+1)\cdot 0}{k+1} = 0$, and even positions have a cost depending on the size of the 0-blocks to which they are mapped.

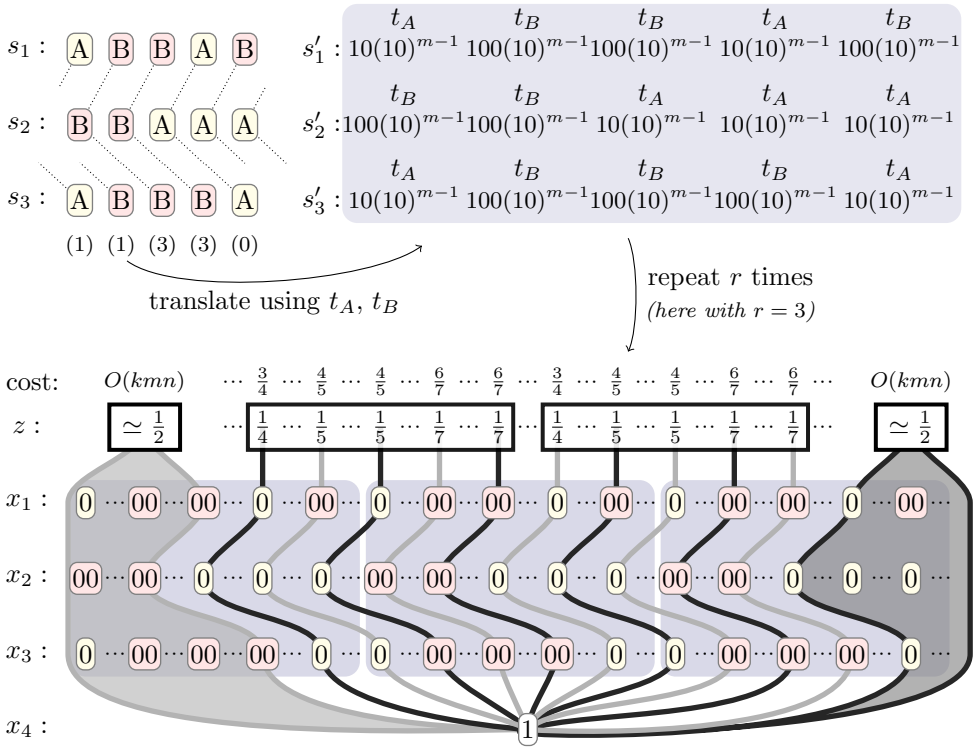

FIG. 6. *Top left: Illustration of the reduction to* DTW-MEAN *from an instance of* ϕ-MSCS *with* $k = 3$ *and* $n = 5$. *An optimal circular shift* $\Delta = (3, 2, 4)$ *is indicated by dotted lines, and the number of* B*'s in a shifted column is below each column. The total cost is* $\text{cost}_\phi(\Delta) = 2\phi_3(1)+2\phi_3(3)+\phi_3(0)$. *Top right: The intermediate strings* s'_1, s'_2, s'_3 *encoding the original strings* s_1, s_2, s_3. *Bottom: The resulting instance* $x_1, \ldots, x_4$ *of* DTW-MEAN *(only coding blocks are shown) and an alignment to a time series* z *mimicking the circular shift* Δ. *The values of* z *are shown along with the cost of each position (positions that are only aligned to non-coding blocks are ignored, and contribute a* background cost *of either* 0 *or* $\frac{3}{4}$*). Note that the cost function* ϕ *is chosen so that the cost of a position aligned to* k *coding blocks equals the cost of the corresponding column of the original circular shift (plus the background cost* $\frac{3}{4}$*). For example, a position aligned to two* A*-coding blocks and one* B*-coding block has cost* $\frac{4}{5} = \frac{3}{4} + \phi_3(1)$, *where* $\phi_3(1)$ *is the cost of a column with two* A*'s and one* B *in* ϕ-MSCS. *The value of* m *is chosen large enough to yield a large cost for misalignments, such as two consecutive coding blocks of the same series aligned together. The value of* r *is chosen such that only a periodic pattern ensures low cost of* z*, even though it requires to pay a high (but bounded) misalignment cost for the first and last positions.*

Consider an even position i such that $i \bmod 2m \neq 2$. The i-th regular position is not aligned with any coding block in any x_j. Thus, it is aligned to k 0's and a single 1, and has cost $C(i+1) = \frac{k}{k+1}$. There are $(m-1)n(r-1)$ such positions, which thus contribute a total cost of

$$(r-1)\frac{(m-1)nk}{k+1}.$$

For an even position i with $i \bmod 2m = 2$, the i-th regular position is aligned with a coding block in each x_j (except for the single 1 in x_{k+1}). Let $i = 2mi' + 2$. Then, $z[i+1]$ is aligned to coding blocks corresponding to column $i' \bmod n$ of Δ. If this column contains a A's and $k-a$ B's, then $z[i+1]$ is aligned to $a + 2(k-a)$ 0's

and a single 1 and has cost

$$C(i+1) = \frac{a+2(k-a)}{a+2(k-a)+1} = \frac{2k-a}{2k-a+1} = \phi_k(k-a) + \frac{k}{k+1}.$$

Note that $\phi_k(k-a)$ is the cost of column $i' \bmod n$ of Δ. Hence, the overall cost of the $(r-1)n$ regular positions i with $i \bmod 2m = 2$ is

$$(r-1)\,\mathrm{cost}_\phi(\Delta) + (r-1)\frac{nk}{k+1} \le (r-1)\left(c + \frac{nk}{k+1}\right).$$

Overall, the regular positions have a total cost of at most

$$(r-1)\frac{(m-1)nk}{k+1} + (r-1)\left(c + \frac{nk}{k+1}\right) = (r-1)\left(c + \frac{mnk}{k+1}\right). \qquad \square$$

CLAIM 5.3. *The total cost of extreme positions is at most* $2knm + 2$.

Proof. Since $0 \le \delta_j < n$ for $j \in \{1, \dots, k\}$, an extreme position $i \in \{1, |z|\}$ is aligned to at most $2nm$ consecutive blocks in each x_j, thus accounting for at most nm 1's in x_j. Moreover, position i is aligned with the additional 1 in x_{k+1}. Thus, position i is aligned with at most $nmk + 1$ 1's, which implies $C(i) \le nmk + 1$ (this bound is achieved if $z[i] = 0$). $\square$

Combining Claims 5.2 and 5.3, we obtain

$$\mathcal{F}(z) \le \sum_{i=1}^{|z|} C(i) \le (r-1)\left(c + \frac{mnk}{k+1}\right) + 2knm + 2 \le c'.$$

Hence, $(x_1, \dots, x_{k+1}, c')$ is a yes-instance of DTW-MEAN.

(ii) No-instance of ϕ-MSCS. We assume that every multiple circular shift Δ has $\mathrm{cost}_\phi(\Delta) \ge c + \varepsilon$. Consider a fixed mean time series z (minimizing $\mathcal{F}$) together with optimal warping paths between z and $x_1, \dots, x_{k+1}$. We show that $\mathcal{F}(z) > c'$. We will do this hierarchically, starting with a lower bound on the cost of an individual position of z. We then proceed to derive lower bounds for the cost of certain intervals of positions until finally obtaining the desired lower bound on $\mathcal{F}(z)$. Before doing so, we start with some preliminary observations about the structure of a mean.

Structural Observations. We say that position i of x_j is *matched* to position i' of z if (i, i') is in the warping path between x_j and z. Clearly, the single position in x_{k+1} is matched to every position of z. We write $\#_1(i)$ and $\#_0(i)$ respectively for the number of positions with value 1 (resp. 0) among $x_1, \dots, x_k$ (ignoring the extra sequence x_{k+1}) which are matched to position i of z. Clearly, the cost of i is

$$C(i) = \frac{\#_0(i)(\#_1(i)+1)}{\#_0(i)+\#_1(i)+1}.$$

We will use the following monotonicity property of the cost.

LEMMA 5.4. *For any* $a \ge a' \ge 0$ *and* $b \ge b' \ge 1$, *it holds* $\frac{ab}{a+b} \ge \frac{a'b'}{a'+b'}$.

Proof. It suffices to see that the partial derivatives

$$\frac{\partial}{\partial a}\frac{ab}{a+b} = \frac{b^2}{(a+b)^2} \quad \text{and} \quad \frac{\partial}{\partial b}\frac{ab}{a+b} = \frac{a^2}{(a+b)^2}$$

are non-negative for $a \ge 0$ and $b \ge 1$. $\square$

We define some further notation for the remaining part of the proof. The *range* of a position i of z in x_j is the set of positions of x_j that are matched to i. The range is a subinterval of $\{1, 2, ..., |x_j|\}$ and by construction of x_j, the corresponding subseries cannot have three consecutive 0's or two consecutive 1's. More precisely, its values alternate between 0 and 1, except for the (rare) occasions where it includes a B-coding block. The number of blocks of x_j intersecting the range of i in x_j is denoted $r_j(i)$. The number of B-coding blocks contained in the range of i in x_j is denoted $r_j^B(i)$ and we define $r^B(i) \coloneqq \sum_{j=1}^{k} r_j^B(i)$.

A position i of z is called *0-simple* (resp. *1-simple*) if $\#_1(i) = 0$ (resp. $\#_0(i) = 0$). It is *simple* if it is 0- or 1-simple, and it is *bad* otherwise. Clearly, the cost of a 1-simple position is 0. For a 0-simple position i, we have $\#_1(i) = 0$ and $k \leq \#_0(i) \leq 2k$ (more precisely, $\#_0(i) = k + r^B(i)$ and $r^B(i) \leq k$). Thus, $\frac{k}{k+1} \leq C(i) \leq \frac{2k}{2k+1}$. Since we assumed $k \geq 15$, the cost of a 0-simple position is always contained in $[0.9, 1]$.

We continue with several structural observations regarding a mean.

Observation 5.5. *There exists a mean without consecutive 1-simple positions or consecutive 0-simple positions. Such a mean is called* non-redundant*.*

Proof. Any two consecutive 1-simple (or 0-simple) positions of a non-redundant mean z have consecutive or intersecting ranges in each x_j with the same value (1 or 0). Hence, they can be merged into one single 1-simple (or 0-simple) position.

Since the warping of the other positions in z remains unchanged, we focus on the cost of the merged position. For 1-simple positions, the cost remains unchanged (both solutions yield a cost of 0 for the 1-simple positions). For 0-simple positions, the cost of the two 0-simple positions in the original solution is at least 0.9 each. However, the cost of the merged 0-simple position is at most 1, which is a contradiction to z being a mean. ∎

In the following, we assume z to be a non-redundant mean.

We say a block b of some input x_j is *matched* (*fully matched*) to a position i in z if some position (all positions) in b is (are) matched to i. That is, a matched block intersects the range of i and a fully matched block is included in the range of i. Note that the distinction is only relevant for B-coding blocks, as all other blocks have size 1. Moreover, the number of B-coding blocks that are fully matched to a position i equals $r^B(i)$.

Observation 5.6. *For a non-redundant mean z, any B-coding block of some x_j that is not fully matched to a position in z is matched to at least one bad position in z.*

If any block is matched to two consecutive positions in z, then at least one of them is bad.

Proof. Consider a B-coding block b and all positions of z to which it is matched. There are at least two of them, which cannot all be 0-simple (since z is non-redundant). Also, none of them can be 1-simple (since b is a 0-block). Thus, at least one of them is bad.

We prove the contrapositive of the second statement: If two simple positions have a common block matched to them, then they are both a-simple, $a \in \{0, 1\}$, and cannot be consecutive in a non-redundant mean. ∎

We now introduce an *assignment* relation between a block b of some input series and a position i of z. We say that b is *assigned* to the position i if i is the leftmost simple position to which b is fully matched (if any), or (if no such simple position

exists) if i is the leftmost bad position to which b is matched. Note that any size-1 block is fully matched to at least one position (simple or bad), and every size-2 block is either fully matched to a simple position or matched to a bad position (by Observation 5.6). Thus, every block is assigned to exactly one position.

For a position i of z, we introduce the following quantities:

$$\begin{aligned}
\diamondsuit_0(i) &\coloneqq \text{number of 0-blocks in } x_1, \dots, x_k \text{ matched to } i,\\
\diamondsuit_1(i) &\coloneqq \text{number of 1-blocks in } x_1, \dots, x_k \text{ matched to } i,\\
q(i) &\coloneqq \text{number of } B\text{-coding blocks assigned to } i,\\
g(i) &\coloneqq \begin{cases} 0, & \text{if } i \text{ is simple}\\ \max\{1, r_1(i)-1, \dots, r_k(i)-1\}, & \text{if } i \text{ is bad} \end{cases}.
\end{aligned}$$

We quickly observe the following:

(I) If i is simple, then $q(i) = r^B(i)$.
Proof. By definition, every B-coding block assigned to i is fully matched to i, that is, $q(i) \le r^B(i)$. Furthermore, a B-coding block can be fully matched to only one position. Thus, if i is simple, then all B-coding blocks fully matched to it are also assigned to it, that is, $q(i) \ge r^B(i)$.

(II) If i is 0-simple, then $\diamondsuit_0(i) = k$, $\diamondsuit_1(i) = 0$, and $C(i) = \frac{k+q(i)}{k+q(i)+1}$.
Proof. A 0-simple position has exactly one 0-block in each $x_1, \dots, x_k$ matched to it. The cost follows from (I).

(III) If i is 1-simple, then $\diamondsuit_0(i) = 0$, $\diamondsuit_1(i) = k$, and $C(i) = 0$.
Proof. A 1-simple position has exactly one 1-block in each $x_1, \dots, x_k$ matched to it.

(IV) If i is bad, then $g(i) \le 2\min\{\diamondsuit_1(i), \diamondsuit_0(i)\}$.
Proof. For a bad position i, we have

$$\min\{\diamondsuit_1(i), \diamondsuit_0(i)\} \ge \max\left\{1, \frac{1}{2}\max_{j=1,\dots,k}(r_j(i)-1)\right\}.$$

(V) If i is bad, then $C(i) \ge \frac{1}{2}\min\{\diamondsuit_1(i)+1, \diamondsuit_0(i)\}$.
Proof. Let $\eta \coloneqq \min\{\diamondsuit_1(i)+1, \diamondsuit_0(i)\} \ge 1$. Then i is aligned to at least η 0's and at least η 1's. Thus, by Lemma 5.4, $C(i) \ge \frac{\eta^2}{2\eta} = \frac{\eta}{2}$.

(VI) For every position i, it holds that $|\diamondsuit_0(i) - \diamondsuit_1(i)| \le k$ and $\diamondsuit_0(i) + \diamondsuit_1(i) \ge k$.
Proof. For each x_j, $1 \le j \le k$, the difference between the number of 0-blocks matched to i and the number of 1-blocks matched to i is at most one. Clearly, there is at least one 0- or 1-block matched to i in each x_j.

Cost of a Single Position. We consider a fixed position i of z. For simplification, we write $\diamondsuit_0 \coloneqq \diamondsuit_0(i)$, $\diamondsuit_1 \coloneqq \diamondsuit_1(i)$, $q \coloneqq q(i)$, $g \coloneqq g(i)$, and $C \coloneqq C(i)$. The goal is to provide a lower bound for C that can be decomposed into the following elements:

- a *background cost* $C_{\text{back}}(i) \coloneqq \frac{\diamondsuit_0}{k+1}$,
- a *coding cost* $C_{\text{code}}(i) \coloneqq \phi_k(q)$ reflecting the extra cost induced by a matched coding block,
- a *gap cost* of $C_{\text{gap}}(i) \coloneqq 0.01g$, which is 0 if i is simple, and which increases with the number of blocks matched to i if i is bad.

CLAIM 5.7. *The cost C of position i is at least $LB(\diamondsuit_0, q, g, k)$, defined as follows:*

$$\begin{aligned}
LB(\diamondsuit_0, q, g, k) &\coloneqq C_{\text{back}}(i) + C_{\text{code}}(i) + C_{\text{gap}}(i)\\
&= \frac{\diamondsuit_0}{k+1} + \phi_k(q) + 0.01g.
\end{aligned}$$

Proof. In the following, we write LB for $LB(\diamondsuit_0, q, g, k)$. We prove $C \geq LB$ by case distinction.

For a 0-simple position, by (II), we have $\diamondsuit_0 = k$ and $g = 0$. Thus,

$$LB = \frac{k}{k+1} + \phi_k(q) + 0 = \frac{k+q}{k+q+1} = C.$$

For a 1-simple position, by (III), we have $\diamondsuit_0 = 0$, $q = 0$, and $g = 0$. Thus,

$$LB = 0 + \phi_k(0) + 0 = 0 = C.$$

For a bad position, we have $\diamondsuit_0 \geq 1$ and $\diamondsuit_1 \geq 1$. First, note that

$$LB - 0.01g = \frac{\diamondsuit_0}{k+1} + \frac{q}{(k+q+1)} \cdot \frac{1}{(k+1)} \leq \frac{\diamondsuit_0}{k} + \frac{1}{k+1} \leq 2\frac{\diamondsuit_0}{k},$$

that is, $LB \leq 2\frac{\diamondsuit_0}{k} + 0.01g$.

We now use this upper bound for the following three sub-cases. First, if $\diamondsuit_0 \leq \diamondsuit_1$, then we have

$$\begin{aligned} C &\geq \frac{1}{2}\diamondsuit_0 \quad \text{(by (V))} \\ &\geq \frac{2}{k}\diamondsuit_0 + 0.02\diamondsuit_0 \quad \text{(since } k \geq 15\text{)} \\ &\geq LB \quad \text{(by (IV)).} \end{aligned}$$

Second, if $6 \leq \diamondsuit_1 < \diamondsuit_0$, we have

$$\begin{aligned} C &\geq \frac{1}{2}\diamondsuit_1 \quad \text{(by (V))} \\ &\geq 2 + \left(\frac{2}{k} + 0.02\right)\diamondsuit_1 \quad \text{(since } k \geq 15 \text{ and } \diamondsuit_1 \geq 6\text{)} \\ &= 2 + \frac{2\diamondsuit_1}{k} + 0.02\diamondsuit_1 \\ &\geq \frac{2\diamondsuit_0}{k} + 0.02\diamondsuit_1 \quad \text{(since } \diamondsuit_0 \leq \diamondsuit_1 + k \text{ by (VI))} \\ &\geq LB \quad \text{(by (IV)).} \end{aligned}$$

Finally, if $\diamondsuit_1 \leq 5 < \diamondsuit_0$ (note that $\diamondsuit_1 < \diamondsuit_0 \leq 5$ is not possible since, by (VI), we have $\diamondsuit_1 + \diamondsuit_0 \geq k$ and we assumed $k \geq 15$), then $k - 5 \leq \diamondsuit_0 \leq k + 5$ (by (VI)), and $g \leq 10$ (by (IV)). We have

$$C = \frac{(\diamondsuit_1 + 1)\#_0(i)}{\diamondsuit_1 + 1 + \#_0(i)} \geq \frac{2(k-5)}{k-3} \geq 1.66$$

using Lemma 5.4, with $\diamondsuit_1 \geq 1$, $\#_0(i) \geq \diamondsuit_0 \geq k - 5$, and $k \geq 15$. On the other side, we have

$$\begin{aligned} LB &= \frac{\diamondsuit_0}{k+1} + \phi_k(q) + 0.01g \\ &\leq \frac{5}{k+1} + \frac{k}{k+1} + \phi_k(q) + 0.1 \quad \text{(since } \diamondsuit_0 \leq k+5 \text{ and } g \leq 10\text{)} \\ &\leq \frac{5}{k} + 1.1 \quad \text{(since } \frac{k}{k+1} + \phi_k(q) \leq 1\text{)} \\ &\leq 1.44 \quad \text{(since } k \geq 15\text{)} \\ &\leq C. \end{aligned}$$

∎

Cost of (Ir)regular Intervals. We aim at computing lower bounds on the cost of intervals of positions. Two positions i, i' of the mean z at distance $|i - i'| = \ell$ form an *irregular pair* if for some x_j a block b is matched to i and a block b' is matched to i' such that either $|b - b'| \leq \ell - \frac{m}{2k}$ or $|b - b'| \geq \ell + \frac{m}{2k}$. An interval I of positions in z is called *regular* if it does not contain any irregular pair (otherwise it is called *irregular*). The background, coding, and gap cost of I is the sum of the respective costs of its positions. The structure of regular intervals allows us to bound the coding cost from below using the minimum cost of the original ϕ-MSCS instance. Irregular intervals contain bad positions, which allow us to derive a lower bound on their gap cost.

We first introduce some notation: a position i of z is *j-coding* if there is a coding block b in x_j such that b is assigned to i; it is *coding* if it is j-coding for some j, otherwise it is *non-coding*. A non-coding position i is *free* if all positions at distance at most $\frac{m}{2k} + 2$ from i are non-coding (see Figure 7). We first make the following technical claim before proving the main bound on the coding cost of regular intervals (Claim 5.9).

Claim 5.8. *In a regular interval, if two positions $i < i'$ are at distance at most $2\alpha m - \frac{m}{2k}$ for some α, then, for any j, there are at most α j-coding positions in $[i, i']$.*

Conversely, if i and i' are at distance at least $2\alpha m + \frac{m}{2k} + 1$, then, for any j, there are at least α j-coding positions in $[i, i']$.

Proof. Fix $j \in \{1, \ldots, k\}$ and consider the first block b in x_j matched to i and the last block b' in x_j matched to i' (then $b' \geq b$). Note that all j-coding positions in $[i, i']$ have been assigned a distinct coding block in $[b, b']$. Since i and i' are not an irregular pair, it holds that

$$b' - b < i' - i + \frac{m}{2k} \leq 2\alpha m.$$

That is, $b' < b + 2\alpha m$, and thus x_j contains at most α coding blocks in $[b, b']$. These coding blocks are assigned to positions in $[i, i']$. Hence, $[i, i']$ contains at most α j-coding positions.

For the other direction, consider again blocks b and b' as above. In this case there is a slight difference: If block b or b' is coding, then it might be assigned to a coding position outside of the interval $[i, i']$, which then would not count in the lower bound. Thus, we consider only blocks strictly between b and b', among which all coding blocks are assigned to a coding position in $[i, i']$. Since $i' - i \geq 2\alpha m + \frac{m}{2k} + 1$, we have $b' - b > i' - i + \frac{m}{2k} \geq 2\alpha m + 1$, so there are at least $2\alpha m$ blocks strictly between b and b', including at least α coding blocks. These are assigned to at least α j-coding positions in $[i, i']$. ∎

Claim 5.9. *The coding cost of a length-ℓ regular interval I is at least*

$$C_{\text{code}}(I) \geq \left(\frac{\ell + 1}{2mn} - 2\right)(c + \varepsilon).$$

Proof. Let I be a regular length-ℓ interval. Assume that $\ell \geq 3mn$ (otherwise the stated lower bound is negative which is trivial).

The first part of the proof consists of splitting interval I into consecutive length-$2m$ segments, each one containing exactly one j-coding position for each j. To this end, a few positions need to be cropped from both ends of I. In other words, we need to find a good starting point (a free position) close to the left end of I.

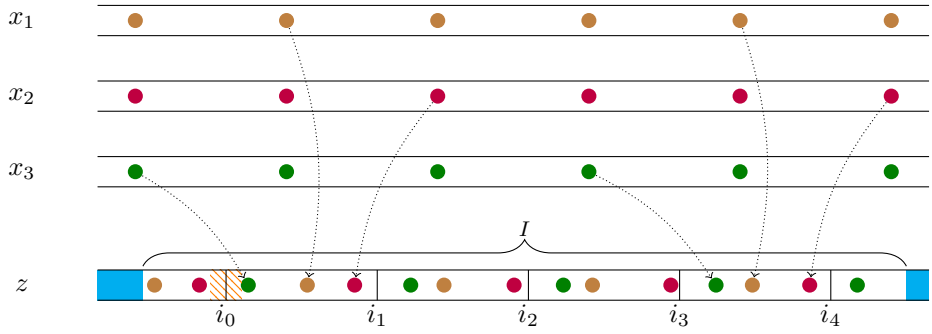

FIG. 7. *Decomposition of a regular interval (I) into segments delimited at positions i_l. Coding blocks in x_j are indicated with colored bullets, as well as their assigned positions in z (dotted arcs). Position i_0 is free (no coding position within the striped area), and each segment $[i_l, i_{l+1}]$ contains exactly one j-coding position from each x_j. These coding positions correspond to columns of a multiple circular shift of the ϕ-MSCS input strings. The coding cost of n consecutive segments can be bounded from below by the minimum cost of the ϕ-MSCS instance.*

Consider the first position i of I and position $i' := i + 2m - \frac{m}{2k}$ (in I). By Claim 5.8, for any j, there is at most one j-coding position in $[i, i']$ (and so at most k such coding positions in total). Accordingly, the coding positions split the interval $[i, i']$ into at most $(k+1)$ disjoint intervals of non-coding positions, with total size at least $(i' - i + 1) - k = 2m - \frac{m}{2k} - k + 1$. Hence, there is one interval of non-coding positions in $[i, i']$ with size at least

$$\frac{2m - \frac{m}{2k} - k + 1}{k+1} \geq \frac{2m - \frac{m}{2k}}{k+1} - 1 \geq \frac{m}{k} + 5.$$

Note that the second inequality above is equivalent to

$$\frac{m(2 - \frac{1}{2k})}{k+1} \geq \frac{m}{k} + 6 \quad \Leftrightarrow \quad m \geq \frac{6k(k+1)}{k - \frac{3}{2}},$$

which is true since $m \geq 1600k$ and $k \geq 15$. Hence, this interval contains a free position, denoted i_0, with $i + \frac{m}{2k} + 2 \leq i_0 \leq i + 2m - \frac{m}{2k} - 2$.

Let $\lambda := \lfloor \frac{\ell}{2m} \rfloor - 2$ and $i_l := i_0 + 2lm$ for $0 < l \leq \lambda$. Note that $\lambda \geq n$ and that every i_l is in I (following the assumption on ℓ).

We fix some input series x_j, $1 \leq j \leq k$. Intuitively, positions i_l are the cutting points of our segments within interval I (see Figure 7). We now aim at showing that there is exactly one j-coding position in each segment $[i_{l-1}, i_l - 1]$. First, consider positions $h = i_0 - \frac{m}{2k} - 2$ and $i_l - 1$. By Claim 5.8, since $(i_l - 1) - h = 2lm + \frac{m}{2k} + 1$, interval $[h, i_l - 1]$ contains at least l j-coding positions. Since i_0 is free, these coding positions cannot be before i_0 (as $i_0 - h = \frac{m}{2k} + 2$), so they are in $[i_0, i_l - 1]$. Consider now positions $h' = i_0 + \frac{m}{2k} + 1$ and $i_l - 1$. By Claim 5.8, since $(i_l - 1) - h' = 2lm - \frac{m}{2k}$, there are at most l j-coding positions in $[h', i_l - 1]$, and therefore at most l j-coding positions in $[i_0, i_l - 1]$. That is, there are exactly l j-coding positions in $[i_0, i_l - 1]$.

Overall, there is exactly one j-coding position in $[i_{l-1}, i_l - 1]$ for every $0 < l \leq \lambda$ and every j. We write $C_{l,j}$ for the corresponding coding block in x_j.

Let q_l be the number of B-coding blocks among $C_{l,1}, \ldots, C_{l,k}$. Then,

$$\sum_{h=i_{l-1}}^{i_l - 1} q(h) = q_l,$$

and thus

$$\sum_{h=i_{l-1}}^{i_l-1} \phi_k(q(h)) = \sum_{h=i_{l-1}}^{i_l-1} \frac{q(h)}{(k+q(h)+1)(k+1)}$$
$$\geq \sum_{h=i_{l-1}}^{i_l-1} \frac{q(h)}{(k+q_l+1)(k+1)} = \phi_k(q_l).$$

Let δ_j be such that $C_{1,j}$ is the δ_j-th coding block of x_j. Then, $C_{l,j}$ is the (δ_j+l-1)-th coding block of x_j, for every $0 < l \leq \lambda$. Note that the coding blocks $C_{l,1}, \ldots, C_{l,k}$ correspond to the $((l-2) \bmod n+1)$-th column in the multiple circular shift $\Delta = (\delta_1 \bmod n, \ldots, \delta_k \bmod n)$ of $s_1, \ldots, s_k$. Thus, q_l is the number of B's in this column and $\phi_k(q_l)$ is the corresponding cost of this column.

That is, for any integer a with $0 < a \leq \lambda - n$, the sum $\sum_{l=a}^{a+n-1} \phi_k(q_l)$ corresponds to the cost of some multiple circular shift of $s_1, \ldots, s_k$. Since, by assumption, every multiple circular shift of $s_1, \ldots, s_k$ has cost at least $c+\varepsilon$, we have

$$\sum_{l=a}^{a+n-1} \phi_k(q_l) \geq c+\varepsilon.$$

We can now compute the lower bound on the coding cost of interval I. To this end, we first extract $\left\lfloor \frac{\lambda}{n} \right\rfloor \geq 1$ length-$2mn$ subintervals of I, each consisting of n segments of the form $[i_{l-1}, i_l - 1]$. It follows

$$C_{\text{code}}(I) = \sum_{h \in I} \phi_k(q(h)) \geq \sum_{h=i_0}^{i_\lambda-1} \phi_k(q(h))$$
$$\geq \sum_{a=0}^{\lfloor \frac{\lambda}{n} \rfloor - 1} \sum_{l=an}^{an+n-1} \sum_{h=i_{l-1}}^{i_l-1} \phi_k(q(h))$$
$$\geq \sum_{a=0}^{\lfloor \frac{\lambda}{n} \rfloor - 1} \sum_{l=an}^{an+n-1} \phi_k(q_l)$$
$$\geq \sum_{a=0}^{\lfloor \frac{\lambda}{n} \rfloor - 1} c+\varepsilon$$
$$= \left\lfloor \frac{\lambda}{n} \right\rfloor (c+\varepsilon)$$
$$\geq \left(\frac{\lambda}{n} - 1 \right) (c+\varepsilon)$$
$$= \left(\frac{\lfloor \frac{\ell}{2m} \rfloor - 2}{n} - 1 \right) (c+\varepsilon)$$
$$\geq \left(\frac{\frac{\ell}{2m} - 3}{n} - 1 \right) (c+\varepsilon)$$
$$\geq \left(\frac{\ell+1}{2mn} - 2 \right) (c+\varepsilon),$$

where we assume $n \geq 4$ for the last inequality. ∎

Next, we prove a lower bound on the gap cost of an interval I. In the following, the total number of matches between blocks of $x_1, \dots, x_k$ and positions in I is denoted $W(I) = \sum_{i \in I}(\diamondsuit_0(i) + \diamondsuit_1(i))$.

Claim 5.10. *For any interval I of length ℓ, the gap cost $C_{\text{gap}}(I)$ fulfills*

$$C_{\text{gap}}(I) \geq \frac{1}{100}\left(\frac{W(I)}{k} - \ell\right).$$

Moreover, if I is irregular, then

$$C_{\text{gap}}(I) \geq \frac{m}{400k}.$$

Proof. For the first lower bound, it suffices to note that for any position i (simple or bad), it holds

$$g(i) \geq \max_{1 \leq j \leq k} r_j(i) - 1 \geq \frac{W(i)}{k} - 1,$$

where $W(i) \coloneqq \diamondsuit_0(i) + \diamondsuit_1(i)$.

For the second lower bound, consider an irregular pair $i < i'$ in I and an integer j such that a block b in x_j is matched to i and a block $b' > b$ in x_j is matched to i' where $|(b' - b) - (i' - i)| > \frac{m}{2k}$.

If $b' - b > i' - i + \frac{m}{2k}$, then

$$\sum_{h=i}^{i'} g(h) \geq \sum_{h=i}^{i'} (r_j(h) - 1) = \sum_{h=i}^{i'} r_j(h) - (i' - i + 1) \geq b' - b - (i' - i) > \frac{m}{2k} > \frac{m}{4k}.$$

If $b' - b < i' - i - \frac{m}{2k}$, then there are at least $\frac{m}{2k}$ pairs of consecutive positions having the same block in x_j matched to them, and for every such pair at least one of the two positions is bad (by Observation 5.6). Since any bad position may be counted in at most two such pairs, the interval has at least $\frac{m}{4k}$ bad positions. Hence, using $g(h) \geq 1$ for bad positions, we obtain $\sum_{h=i}^{i'} g(h) \geq \frac{m}{4k}$. ∎

Cost of a Mean. To obtain a lower bound for $\mathcal{F}(z)$, we now partition the positions $[1, |z|]$ into minimal irregular intervals (from left). To this end, let $\alpha_1 \coloneqq 1$ and let β_1 be the position such that the interval $[\alpha_1, \beta_1]$ is irregular (if such a position does not exist, then $\beta_1 \coloneqq |z|$) and $[\alpha_1, \beta_1 - 1]$ is regular. If $\beta_1 < |z|$, then we continue analogously and define $\alpha_2 \coloneqq \beta_1 + 1$ and β_2 to be the position such that $[\alpha_2, \beta_2]$ is irregular and $[\alpha_2, \beta_2 - 1]$ is regular. This procedure is repeated until we obtain a partition

$$[\alpha_1 \coloneqq 1, \beta_1], [\alpha_2 \coloneqq \beta_1 + 1, \beta_2], \dots, [\alpha_L \coloneqq \beta_{L-1} + 1, \beta_L \coloneqq |z|]$$

of $[1, |z|]$ into $L \geq 1$ intervals of which the first $L - 1$ are irregular and the last is possibly regular. The following lower bounds hold.

Claim 5.11. *For $1 \leq l < L$, it holds that*

$$C_{\text{gap}}([\alpha_l, \beta_l]) + C_{\text{code}}([\alpha_l, \beta_l]) \geq (c + \varepsilon)\frac{W([\alpha_l, \beta_l])}{2knm}.$$

For the coding and gap costs of $[\alpha_L, \beta_L]$, it holds that

$$C_{\text{gap}}([\alpha_L, \beta_L]) + C_{\text{code}}([\alpha_L, \beta_L]) \geq (c + \varepsilon)\frac{W([\alpha_L, \beta_L])}{2knm} - 2(c + \varepsilon).$$

Proof. Consider an interval $[\alpha_l, \beta_l]$, $1 \le l \le L$, and let ℓ be its length. Let $W := W([\alpha_l, \beta_l])$. Since $[\alpha_l, \beta_l - 1]$ is a regular interval of length $\ell - 1$, by Claim 5.9, we have the following lower bound on the coding cost:

$$C_{\text{code}}([\alpha_l, \beta_l]) \ge C_{\text{code}}([\alpha_l, \beta_l - 1]) \ge \left(\frac{\ell}{2mn} - 2\right)(c + \varepsilon). \tag{5.1}$$

For $l < L$, we combine both bounds on the gap cost of Claim 5.10 (by averaging their values):

$$C_{\text{gap}}([\alpha_l, \beta_l]) \ge \frac{1}{200}\left(\frac{W}{k} - \ell\right) + \frac{m}{800k}.$$

Using $m \ge 1600k(c + \varepsilon)$ (by definition) and $m \ge 100\frac{c+\varepsilon}{n}$, we obtain

$$C_{\text{gap}}([\alpha_l, \beta_l]) \ge \frac{c + \varepsilon}{2nm}\left(\frac{W}{k} - \ell\right) + 2(c + \varepsilon). \tag{5.2}$$

The sum of Inequations 5.1 and 5.2 yields the claimed lower bound.

For interval $[\alpha_L, \beta_L]$, we use the general lower bound from Claim 5.10, which yields

$$C_{\text{gap}}([\alpha_L, \beta_L]) \ge \frac{1}{100}\left(\frac{W}{k} - \ell\right) \ge \frac{c + \varepsilon}{2mn}\left(\frac{W}{k} - \ell\right). \tag{5.3}$$

The sum of Inequations 5.1 and 5.3 yields the claimed lower bound. ∎

Finally, to finish the proof of Theorem 5.1, we show that the mean z has high cost, that is, $(x_1, \ldots, x_{k+1}, c')$ is a no-instance of DTW-Mean.

Claim 5.12. $\mathcal{F}(z) > c'$.

Proof. Using Claim 5.7 on each position of $I := [1, |z|]$, we obtain the following lower bound

$$\mathcal{F}(z) = \sum_{i=1}^{|z|} C(i) \ge C_{\text{code}}(I) + C_{\text{gap}}(I) + C_{\text{back}}(I).$$

For the coding and gap cost of I, we use Claim 5.11 together with the fact that all $2knmr$ blocks of $x_1, \ldots, x_k$ are involved in at least one match with a position of z, which yields $W(I) = \sum_{l=1}^{L} W([\alpha_l, \beta_l]) \ge 2knmr$. Thus,

$$C_{\text{code}}(I) + C_{\text{gap}}(I) \ge (c + \varepsilon)r - 2(c + \varepsilon).$$

The overall background cost is $C_{\text{back}}(I) = \sum_{i=1}^{|z|} \frac{\diamondsuit_0(i)}{k+1}$. Since overall there are $knmr$ 0-blocks in $x_1, \ldots, x_k$, and each of those is matched to at least one position of z, we have $\sum_{i=1}^{|z|} \diamondsuit_0(i) \ge knmr$ and thus

$$C_{\text{back}}(I) \ge \frac{nmrk}{k+1}.$$

Combining the two bounds above yields

$$C_{\text{code}}(I) + C_{\text{gap}}(I) + C_{\text{back}}(I) \ge \frac{nmrk}{k+1} + (c + \varepsilon)r - 2(c + \varepsilon).$$

Since $\varepsilon r > 3mnk + 2(c + \varepsilon)$, we get

$$\mathcal{F}(z) > \frac{nmrk}{k+1} + rc + 3mnk = c'.$$

∎

Since the above reduction is a polynomial-time reduction from ϕ-MSCS where the resulting number of time series is linear in the number of strings in the ϕ-MSCS instance, Theorem 5.1 now follows from Theorem 3.11. ∎

Closing this section, we remark that Buchin et al. [8, Theorem 7] recently obtained the same hardness results as in Theorem 5.1 for the problem of computing an average series z that minimizes

$$\mathcal{F}_p^q(z) := \sum_{j=1}^{k} \left(\min_{p_j \in \mathcal{P}_{|x_j|,|z|}} \sum_{(u,v)\in p_j} |x_j[u] - z[v]|^p \right)^{q/p}$$

for all integers $p, q \geq 1$. Their reduction, however, builds time series containing three different values. Hence, Theorem 5.1 yields a stronger hardness on binary inputs for $p = q = 2$. Note that if also the mean is restricted to be a binary time series, then the problem is solvable in polynomial time [6, 34].

6. Conclusion. Shedding light on the computational complexity of prominent consensus problems in stringology and time series analysis, we proved several tight computational hardness results for circular string alignment problems and time series averaging in dynamic time warping spaces. Notably, we have shown that the computational complexity of consensus string problems can drastically change (that is, they become hard) when considering *circular* strings instead of classic strings. Our results imply that these problems with a rich set of applications are intractable in the worst case (even on binary data). Hence, it is unlikely to find algorithms which significantly improve the worst-case running times of the best known algorithms. This now partly justifies the use of heuristics as has been done for a long time in many real-world applications.

We conclude with some open questions and directions for future work.

- We conjecture that the idea of the reduction for f-MSCS can be used to prove the same hardness result for most non-linear (polynomially bounded) order-independent cost functions (note that f-MSCS is trivially solvable if f_k is linear since every shift has the same cost). Proving a complexity dichotomy with respect to the cost function is a worthwhile goal.
- From an algorithmic point of view, it would be nice to improve the constant in the exponent of the running time for DTW-Mean, that is, to find algorithms running in $O(n^{\alpha k})$ time for small α. In particular, we ask to find an $O(n^k)$-time algorithm for DTW-Mean.
- What about the parameter maximum sequence length n? Are the considered problems polynomial-time solvable if n is a constant? Are they even fixed-parameter tractable with respect to n?
- Finally, can the hardness result for averaging time series with respect to (p, q)-DTW by Buchin et al. [8, Theorem 7] be strengthened to binary inputs?

Acknowledgments. We are very grateful to two anonymous reviewers of *SIAM Journal on Discrete Mathematics* whose very detailed and constructive feedback helped to improve the presentation of the paper significantly.

References.

[1] A. Abboud, A. Backurs, and V. V. Williams, *Tight hardness results for LCS and other sequence similarity measures*, in Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15), IEEE, 2015, pp. 59–78.

[2] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, *Time-series clustering – A decade review*, Information Systems, 53 (2015), pp. 16–38.
[3] N. Arica, *Cyclic sequence comparison using dynamic warping*, in Proceedings of the International Conference on Image and Video Retrieval (CIVR '05), vol. 3568 of LNCS, Springer, 2005, pp. 328–335.
[4] L. A. K. Ayad and S. P. Pissis, *MARS: improving multiple circular sequence alignment using refined sequences*, BMC Genomics, 18 (2017), p. 86.
[5] C. Barton, C. S. Iliopoulos, R. Kundu, S. P. Pissis, A. Retha, and F. Vayani, *Accurate and efficient methods to improve multiple circular sequence alignment*, in Proceedings of the 14th International Symposium on Experimental Algorithms (SEA '15), vol. 9125 of LNCS, Springer, 2015, pp. 247–258.
[6] M. Brill, T. Fluschnik, V. Froese, B. Jain, R. Niedermeier, and D. Schultz, *Exact mean computation in dynamic time warping spaces*, Data Mining and Knowledge Discovery, 33 (2019), pp. 252–291.
[7] K. Bringmann and M. Künnemann, *Quadratic conditional lower bounds for string problems and dynamic time warping*, in Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15), IEEE, 2015, pp. 79–97.
[8] K. Buchin, A. Driemel, and M. Struijs, *On the hardness of computing an average curve*, CoRR, abs/1902.08053 (2019). Accepted for publication at the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT '20).
[9] L. Bulteau, F. Hüffner, C. Komusiewicz, and R. Niedermeier, *Multivariate algorithmics for NP-hard string problems*, Bulletin of the EATCS, 114 (2014).
[10] J. Chen, B. Chor, M. Fellows, X. Huang, D. W. Juedes, I. A. Kanj, and G. Xia, *Tight lower bounds for certain parameterized NP-hard problems*, Information and Computation, 201 (2005), pp. 216–231.
[11] M. Cuturi and M. Blondel, *Soft-DTW: a differentiable loss function for time-series*, in Proceedings of the 34th International Conference on Machine Learning (ICML '17), vol. 70 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 894–903.
[12] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015.
[13] M. R. Fellows, J. Gramm, and R. Niedermeier, *On the parameterized intractability of motif search problems*, Combinatorica, 26 (2006), pp. 141–167.
[14] F. Fernandes, L. Pereira, and A. T. Freitas, *CSA: An efficient algorithm to improve circular DNA multiple alignment*, BMC Bioinformatics, 10 (2009), p. 230.
[15] M. Frances and A. Litman, *On covering problems of codes*, Theory of Computing Systems, 30 (1997), pp. 113–119.
[16] V. Froese, B. Jain, M. Rymar, and M. Weller, *Fast exact dynamic time warping on run-length encoded time series*, CoRR, abs/1903.03003 (2020).
[17] O. Gold and M. Sharir, *Dynamic time warping and geometric edit distance: Breaking the quadratic barrier*, ACM Transactions on Algorithms, 14 (2018), pp. 50:1–50:17.
[18] J. Gramm, R. Niedermeier, and P. Rossmanith, *Fixed-parameter algorithms for Closest String and related problems*, Algorithmica, 37 (2003), pp. 25–42.

[19] R. Grossi, C. S. Iliopoulos, R. Mercas, N. Pisanti, S. P. Pissis, A. Retha, and F. Vayani, *Circular sequence comparison: algorithms and applications*, Algorithms for Molecular Biology, 11 (2016).
[20] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, Journal of Computer and System Sciences, 62 (2001), pp. 367–375.
[21] E. Keogh and C. A. Ratanamahatana, *Exact indexing of dynamic time warping*, Knowledge and Information Systems, 7 (2005), pp. 358–386.
[22] W. Kuszmaul, *Dynamic time warping in strongly subquadratic time: Algorithms for the low-distance regime and approximate evaluation*, in Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP '19), 2019, pp. 80:1–80:15.
[23] T. Lee, J. C. Na, H. Park, K. Park, and J. S. Sim, *Finding consensus and optimal alignment of circular strings*, Theoretical Computer Science, 468 (2013), pp. 92–101.
[24] M. Li, B. Ma, and L. Wang, *Finding similar regions in many sequences*, Journal of Computer and System Sciences, 65 (2002), pp. 73–96.
[25] M. Li, B. Ma, and L. Wang, *On the closest string and substring problems*, Journal of the ACM, 49 (2002), pp. 157–171.
[26] D. Marx, *Closest substring problems with small distances*, SIAM Journal on Computing, 38 (2008), pp. 1382–1410.
[27] R. A. Mollineda, E. Vidal, and F. Casacuberta, *Cyclic sequence alignments: Approximate versus optimal techniques*, International Journal of Pattern Recognition and Artificial Intelligence, 16 (2002), pp. 291–299.
[28] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson, *Time-series averaging using constrained dynamic time warping with tolerance*, Pattern Recognition, 74 (2018), pp. 77–89.
[29] V. Palazón-González and A. Marzal, *On the dynamic time warping of cyclic sequences for shape retrieval*, Image and Vision Computing, 30 (2012), pp. 978–990.
[30] J. Paparrizos and L. Gravano, *Fast and accurate time-series clustering*, ACM Transactions on Database Systems, 42 (2017), pp. 8:1–8:49.
[31] 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 (2016), pp. 1–26.
[32] F. Petitjean and P. Gançarski, *Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment*, Theoretical Computer Science, 414 (2012), pp. 76–91.
[33] F. Petitjean, A. Ketterlin, and P. Gançarski, *A global averaging method for dynamic time warping, with applications to clustering*, Pattern Recognition, 44 (2011), pp. 678–693.
[34] N. Schaar, V. Froese, and R. Niedermeier, *Faster binary mean computation under dynamic time warping*, in Proceedings of the 31th Annual Symposium on Combinatorial Pattern Matching (CPM '20), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Accepted for publication (arXiv version available).
[35] D. Schultz and B. Jain, *Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces*, Pattern Recognition, 74 (2018), pp. 340–358.
[36] 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 (2016), pp. 63–69.

[37] S. Will and P. F. Stadler, *A common framework for linear and cyclic multiple sequence alignment problems*, in Proceedings of the 14th International Workshop on Algorithms in Bioinformatics (WABI '14), vol. 8701 of LNCS, Springer, 2014, pp. 135–147.