A Reduction of the Dynamic Time Warping Distance to the Longest Increasing Subsequence Length

Yoshifumi Sakai
Graduate School of Agricultural Science, Tohoku University, Sendai, Japan
yoshifumi.sakai.c7@tohoku.ac.jp

Shunsuke Inenaga[1]
Department of Informatics, Kyushu University, Fukuoka, Japan
PRESTO, Japan Science and Technology Agency, Kawaguchi, Japan
inenaga@inf.kyushu-u.ac.jp

Abstract

The similarity between a pair of time series, i.e., sequences of indexed values in time order, is often estimated by the dynamic time warping (DTW) distance, instead of any in the well-studied family of measures including the longest common subsequence (LCS) length and the edit distance. Although it may seem as if the DTW and the LCS(-like) measures are essentially different, we reveal that the DTW distance can be represented by the longest increasing subsequence (LIS) length of a sequence of integers, which is the LCS length between the integer sequence and itself sorted. For a given pair of time series of n integers between zero and c, we propose an integer sequence that represents any substring-substring DTW distance as its band-substring LIS length. The length of the produced integer sequence is $O(c^4n^2)$ or $O(c^2n^2)$ depending on the variant of the DTW distance used, both of which can be translated to $O(n^2)$ for constant cost functions. To demonstrate that techniques developed under the LCS(-like) measures are directly applicable to analysis of time series via our reduction of DTW to LIS, we present time-efficient algorithms for DTW-related problems utilizing the semi-local sequence comparison technique developed for LCS-related problems.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases algorithms, dynamic time warping distance, longest increasing subsequence, semi-local sequence comparison

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.6

Funding *Shunsuke Inenaga*: The work of Shunsuke Inenaga was supported by JST PRESTO Grant Number JPMJPR1922.

1 Introduction

A time series is a sequence of discrete objects which are indexed in time order. Due to the recent developments in sensing technologies and semi-automated M2M communications, a vast amount of time series data has been rapidly produced in industrial, financial, medical, and scientific domains.

The most fundamental task in time series data analytics is to compare time series sequences, and to extract their similarities. The *dynamic time warping* (*DTW*) distance is a fundamental method to compute a similarity between two time series that may vary in speed. It is essentially composed of computing an optimal one-to-many alignment of two time series. Considering one-to-many mappings allows for dynamic shifts of time points, it has made DTW one of the most celebrated algorithms in all areas of algorithms. Not only is DTW widely utilized in time series data analysis [20], but also the use of DTW has been

[1] Corresponding author

extended to a wide range of other applications including image processing [21], hand writing matching [25], sign language recognition [13], music retrieval [12], robotics [15, 14], trajectory data analysis [28, 11], speech recognition [24, 18], and many others.

Consider two time series sequences A and B. For the time being, let us assume for simplicity that $|A| = |B| = n$. There is a fundamental dynamic programming algorithm that computes the DTW distance, together with an alignment achieving the distance, between A and B in $O(|A||B|) = O(n^2)$ time and space [24]. While it is possible to reduce the space-requirement of this dynamic programming method to $O(n)$ by applying Hirschberg's divide-and-conquer algorithm [8], no strongly sub-quadratic time algorithm for computing the DTW distance is known. This is supported by the conditional lower bound such that, unless the Strong Exponential Time Hypothesis (SETH) is false, there is no $O(n^{2-\epsilon})$-time algorithm for any $\epsilon > 0$ that computes the exact value of the DTW distance over sequences over 5-letter alphabets [1, 2]. Later, the same conditional lower bound was shown for 3-letter alphabets [17].

On the practical side, a number of fast heuristic algorithms for DTW have been proposed by the database community (see [29] for a survey). These algorithms typically output approximated values for the DTW distance which in many cases suffice for practical purposes, but, lack theoretical guarantees.

Unlike other sequence comparison measures such as longest common subsequences (LCS) and edit distance, DTW is not a one-to-one alignment. In addition, the underlying grid graph for DTW is vertex-weighted, while those for LCS and edit distance are edge-weighted. Despite these different natures of DTW from those of LCS and edit distance, interestingly, computing LCS and weighted edit distance of two sequences of length n can be reduced to computing DTW of two sequences of length $O(n)$ [1, 17]. Thus, computing the exact DTW distance is at least as hard as for computing LCS and (weighted) edit distance. On the other hand, it is not known whether computing DTW can be reduced to computing LCS or (weighted) edit distance. These are most probably why finding an efficient algorithm for the exact DTW distance is rather challenging, and quite intriguing. Indeed, the first (weakly) sub-quadratic time algorithm for the DTW distance, which runs in $O(n^2 \log\log\log n / \log\log n)$ time, was only recently discovered [7], after 40 years from the seminal paper [24].

A few DTW algorithms whose running times depend on other parameters are also known: Hwang and Gelfand [10] showed how to compute the DTW distance in $O((s+t)n)$ time, where s and t denote the number of non-zero values in A and B, respectively. For the case where the minimum non-zero distance is one, Kuszmaul [17] proposed an algorithm for computing the DTW distance in $O(nd)$ time, where d denotes the DTW distance between A and B. Very recently, Froese et al. [6] presented a run-length-encoding (RLE) based algorithm which computes the DTW distance in $O(kn + \ell m)$ time, where $m = |A|$, $n = |B|$, and k and ℓ are respectively the RLE sizes of A and B. In the case where $k \in O(\sqrt{m})$ and $\ell \in O(\sqrt{n})$, their algorithm runs in $O(k^2\ell + \ell^2 k)$ time.

When A and B are both binary sequences, it is known that the DTW distance can be computed in $O(n^{1.87})$ time [1]. There are other DTW algorithms for binary sequences, running in $O(st)$ time [19, 9], or in $O(k\ell)$ time [5].

1.1 Our contribution

In this paper, we present a new approach for computing the DTW distance, based on a reduction to the *longest increasing subsequence* (*LIS*) problem.

We here use a standard convention that the values in A and B are rounded to integers. Also, we can easily normalize A and B so that the smallest values in A and B are 0 (e.g., when the values in A and B are all positive, then we subtract the smallest value from all

values in A and B). This way, A and B are transformed into sequences over non-negative integers $\{0, \ldots, c\}$, where c is bounded by the difference between the largest and smallest values in the original sequences. Note that the DTW distance remains unchanged before and after the transformation.

Now, our reduction works as follows: Starting from the basic vertex-weighted grid graph of size $m \times n$, we further redefine the DTW distance three times by using two more intermediate grid graphs, which gives us a reduction of the DTW distance between A and B to the LIS length of a integer sequence. The length of the integer sequence is either $O(c^4mn)$ or $O(c^2mn)$ depending of the variant of the DTW distance employed. For long sequences with large m and n, the value of c is often negligibly small and thus can be regarded as a constant in many cases. In particular, $c = 1$ always holds for binary time series such as spike trains and sensor event sequences. In these cases, the DTW distance is represented by the LIS length of an integer sequence of length $O(mn)$, or $O(n^2)$ when $m = n$.

While our reduction of DTW to LIS computes the DTW distance less time- and space-efficiently than the classical dynamic programming method, due to LIS length computation required after the reduction, our aim is at introducing the first algorithm for *semi-local sequence comparison* with DTW. The semi-local sequence comparison problem was first considered by Tiskin [26] with LCS. The task for our case is to preprocess A and B to construct a data structure supporting $O((m + n)^2)$ queries of the DTW distance of any pair of either a prefix of one of A and B and a suffix of the other, or a contiguous subsequence of one and the entire sequence of the other. While the pair of A and B, together with the dynamic programming method, can immediately be used as a naive $O(m + n)$-space data structure supporting $O(mn)$-time queries, our $O(mn)$-space data structure supports $O(\log(m+n))$-time queries. Namely, we achieve exponential speed-up for answering semi-local DTW distance queries, at the sacrifice of quadratic space usage. This gives us a non-trivial time-space trade-off for this problem. We emphasize that, despite the different nature of DTW from that of LCS or edit distance noted previously, our reduction of DTW to LIS allows us to apply Tiskin's semi-local sequence comparison technique, originally developed for LCS-related problems, directly to DTW-related problems. As such applications, we present time-efficient algorithms for the *circular DTW distance*, *square root DTW distance*, and *periodic DTW distance* problems, which can arise in time series data analysis.

2 Preliminaries

For any sequences S and T, let $S \circ T$ denote the concatenation of S followed by T. For any sequence S, we use $|S|$ to denote the length of S and $S[i]$ with $1 \leq i \leq |S|$ to denote the ith element of S, so that $S = S[1] \circ S[2] \circ \cdots \circ S[|S|]$. We sometimes treat any sequence S as the array of its elements and use $\langle S[1], S[2], \ldots, S[|S|]\rangle$ to denote this array. A subsequence of a sequence S is obtained from S by deleting zero or more elements at any position not necessarily contiguous, i.e., $S[i_1] \circ S[i_2] \circ \cdots \circ S[i_\ell]$ with $1 \leq \ell \leq |S|$ and $1 \leq i_1 < i_2 < \cdots < i_\ell \leq |S|$. Any subsequence $S[i'] \circ S[i' + 1] \circ \cdots \circ S[i'']$ with $0 \leq i' - 1 \leq i'' \leq |S|$ is called contiguous and denoted by $S[i' : i'']$. For convenience, $S[i' : i' - 1]$ is regarded as the empty subsequence. A prefix (resp. suffix) of S is a contiguous subsequence $S[i' : i'']$ with $i' = 1$ (resp. $i'' = |S|$).

A *time series* is a nonempty finite sequence over a fixed integer interval $\{0, 1, \ldots, c\}$, which may be obtained from a sequence of real numbers by normalizing and rounding each element, as its practical approximation to be handled. For any pair of time series, the *dynamic time warping* (*DTW*) distance is defined to measure their dissimilarity. Of various variants of the DTW distance, we follow the definition from [16, 6].

▶ **Definition 1.** For any pair of time series A and B, the dynamic time warping (DTW) distance between A and B is the minimum of

$$\sqrt{\sum_{k=1}^{\ell}(A[i_k]-B[j_k])^2} \tag{1}$$

over all sequences $(i_1,j_1)\circ(i_2,j_2)\circ\cdots\circ(i_\ell,j_\ell)$ such that $(i_1,j_1)=(1,1)$, (i_k,j_k) is one of $(i_{k-1}+1,j_{k-1}+1)$, $(i_{k-1}+1,j_{k-1})$, or $(i_{k-1},j_{k-1}+1)$ for any consecutive $(i_{k-1},j_{k-1})\circ(i_k,j_k)$, and $(i_\ell,j_\ell)=(|A|,|B|)$.

For any sequence S of integers, a subsequence of S is increasing, if any element other than the last one is less than the succeeding element. As a generalization, for any integers l and u with $l\leq u$, we say that an increasing subsequence of S is $[l:u]$-*banded*, if the first element is at least l and the last element is at most u. The longest increasing subsequence (LIS) length of S is the maximum of $|S'|$ over all increasing subsequences S' of S. Any increasing subsequence of S that achieves the LIS length of S is called an LIS of S. The $[l:u]$-banded LIS length and an $[l:u]$-banded LIS are defined analogously.

The aim of this article is to propose a reduction of the DTW distance problem to the LIS length problem. This reduction is decomposed into a chain of a few sub-reductions each with respect to the problem of determining the minimum path weight on a certain grid graph. To introduce such intermediate problems, we will use the following terminology.

For any pair (m,n) of positive integers, let a grid graph be $m\times n$, if it consists of all index pairs (i,j) with $1\leq i\leq m$ and $1\leq j\leq n$ as vertices. For any positive integer d, a grid graph is called *d-valued*, if each vertex is weighted by an integer between 0 and $d-1$, including the boundary. For example, a 2-valued (or *binary*) grid graph has vertices each weighted by either zero or one. For convenience, we do not explicitly specify the set of edges of any $m\times n$ grid graph. A path on a grid graph is a sequence of vertices in the graph. For any vertex in a path, we sometimes say that the vertex appears in the path, or that a path contains a vertex. Any contiguous subsequence of a path is called a subpath. For any vertices (i,j) and (i',j'), we say that (i,j) is *followed vertically*, *horizontally*, or *diagonally* by (i',j'), if (i',j') is $(i+1,j)$, $(i,j+1)$, or $(i+1,j+1)$, respectively. Any path on a *vhd-path* (resp. *vh-path*) grid graph is a sequence of vertices such that any vertex in the sequence other than the last one is followed vertically, horizontally, or diagonally (resp. not diagonally but vertically or horizontally) by the succeeding vertex. Any vertex in such a path other than the last vertex is called vertical, horizontal, or diagonal, if it is followed vertically, horizontally, or diagonally by the succeeding vertex in the path, respectively. The weight of a path on any of vhd-path or vh-path graphs is defined as the sum of the weight of (i,j) over all vertices (i,j) in the path. The minimum path weight from (i',j') to (i'',j'') is the minimum of the weight of P over all paths P from (i',j') to (i'',j''). A path from (i',j') to (i'',j'') is called optimal, if this path achieves the minimum path weight from (i',j') to (i'',j'').

3 Reduction

Let A and B be arbitrary two time series both over a fixed integer interval $\{0,1,\ldots,c\}$. We design a sequence of $O(|A||B|)$ integers, called the DTW distance sequence, that can be used to compute the DTW distance between any contiguous subsequence of A and any contiguous subsequence of B by calculating the $[l:u]$-banded LIS length of a certain contiguous subsequence of the DTW distance sequence for certain l and u.

Before proceeding, it is useful to redefine the DTW distance using the following grid graph based on an immediate lemma claimed subsequently.

▶ **Definition 2.** Let the DTW distance graph for A and B be the $|A| \times |B|$ (c^2+1)-valued vhd-path grid graph with each vertex (i, j) weighted by $(A[i] - B[j])^2$.

▶ **Lemma 3.** *The square of the DTW distance between any subsequences $A[i' : i'']$ and $B[j' : j'']$ is equal to the minimum path weight on the DTW distance graph for A and B from (i', j') to (i'', j'').*

To define the DTW distance sequence for A and B, we further redefine the DTW distance three times in a manner similar to the above, by introducing two more intermediate grid graphs one by one. An outline is as follows. The first intermediate grid graph is obtained from the DTW distance graph by transforming into a binary one, which we will call the binary DTW distance graph. To define this graph, we introduce a general technique that allows us to transform an $m \times n$ $(d+1)$-valued vhd-path grid graph into a $dm \times dn$ binary vhd-path grid graph that represents the minimum path weight on the original graph from any vertex to any. The number of zero-weighted vertices of the resulting graph is $O(dmn)$. The second intermediate one is obtained from the binary DTW distance graph by transforming into a vh-path one, which we will call the binary vh-path DTW distance graph. To define this graph, we also introduce a general technique that allows us to transform an $m \times n$ binary vhd-path grid graph into an $(m + (m-1)(n-1)) \times (n + (m-1)(m-1))$ binary vh-path grid graph that represents the minimum path weight on the original graph from any vertex to any. The number of zero-weighted vertices in the resulting graph is $O(mn)$. The DTW distance sequence is obtained from this second intermediate graph. To defined this sequence, we introduce a general technique that allows us to transform an $m \times n$ binary vh-path grid graph into a sequence of integers that represents the minimum path weight on the original graph from any vertex to any. The length of this sequence is at most the number of zero-weighted vertices in the original graph.

Due to the above chain of transformation, we can define the DTW distance sequence as an integer sequence of length $O(mn)$, if c is treated as a constant, or $O(c^4mn)$, otherwise, that represents the DTW distance between any contiguous subsequence of A and any contiguous subsequence of B. The chain is conceptual and we do not need to construct each intermediate graph explicitly. Instead, we can construct the DTW distance sequence directly from A and B in time linear in the length of the DTW distance sequence.

We present below each transformation technique mentioned above by a separate subsection, and the final subsection defines the DTW sequence.

3.1 A binary vhd-path grid graph representing a multi-valued one

Let G be an arbitrary $m \times n$ $(d+1)$-valued vhd-path grid graph, with each vertex (i, j) weighted by $w(i, j)$. This section introduces a $dm \times dn$ binary vhd-path grid graph, denoted $\bar{G}$, that represents the minimum path weight on G between any pair of vertices. To obtain such $\bar{G}$, we replace each vertex (i, j) in G by a $d \times d$ binary vhd-path grid graph, which we will call the (i, j)-block, such that the minimum path weight on this block from the upper-left corner vertex $(1, 1)$ to any of the other corner vertices $(1, d)$, $(d, 1)$, and (d, d) is equal to $w(i, j)$, the weight of (i, j) on G.

▶ **Definition 4.** Let $\bar{G}$ denote the $dm \times dn$ binary vhd-path grid graph such that the weight of any of vertices $(d(i-1)+1, d(j-1)+k)$, $(d(i-1)+k, d(j-1)+1)$, and $(d(i-1)+k, d(j-1)+k)$ with $1 \le i \le m$, $1 \le j \le n$, and $1 \le k \le d - w(i, j)$ is zero and the weight of any other vertex is one (see Figure 1).

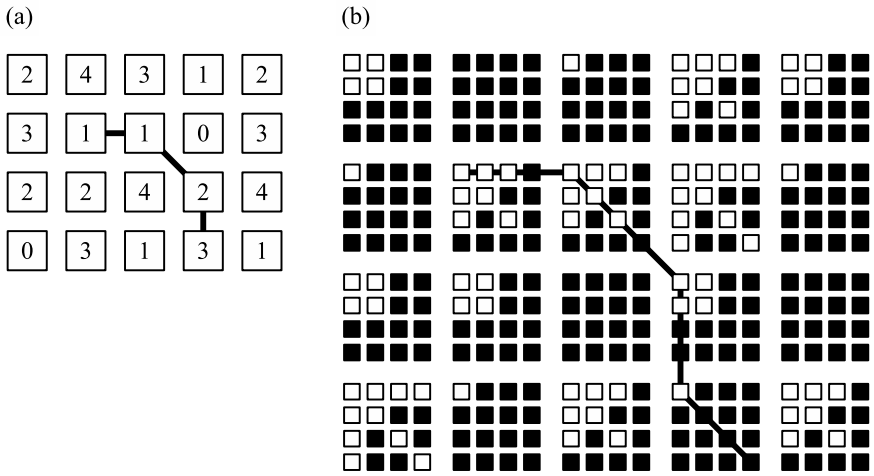


Figure 1 (a) A 4×5 5-valued vhd-path grid graph G and (b) the 16×20 binary vhd-path grid graph $\bar{G}$ for this G are presented, where each vertex in $\bar{G}$ that is weighted by zero (resp. one) is indicated by an open (resp. a solid) square. The path indicated by a polygonal line in (a) is represented by the path indicated analogously in (b).

For any vertex (i, j) of G, let us refer to the subgraph of $\bar{G}$ consisting of all vertices (g, h) with $d(i-1)+1 \leq g \leq di$ and $d(j-1)+1 \leq h \leq dj$ as the (i, j)-block. Let ${}^{\star}(i, j)$, $(i, j)^{\star}$, ${}_{\star}(i, j)$, and $(i, j)_{\star}$ respectively denote the upper-left, upper-right, lower-left, and lower-right corner vertices of the (i, j)-block, so that, for example, $(i, j)_{\star}$ represents the vertex (di, dj) in $\bar{G}$. Graph $\bar{G}$ is designed so as to immediately have the following two properties.

▶ **Property 5.** *The weight of the shortest path from ${}^{\star}(i, j)$ to any of $(i, j)^{\star}$, ${}_{\star}(i, j)$, and $(i, j)_{\star}$ is equal to $w(i, j)$.*

▶ **Property 6.** *The weight of any path from a zero-weighted vertex in the (i, j)-block to any vertex on the rightmost column or lowermost row of the block is at least $w(i, j)$.*

Graph $\bar{G}$ inherently represents G in the sense as follows.

▶ **Lemma 7.** *For any vertices (i', j') and (i'', j'') of G with $i' \leq i''$ and $j' \leq j''$, the minimum path weight on G from (i', j') to (i'', j'') is equal to the minimum path weight on $\bar{G}$ from ${}^{\star}(i', j)$ to $(i'', j'')_{\star}$.*

Proof. For any path P on G from (i', j') to (i'', j''), Property 5 immediately allows us to define a path Q on $\bar{G}$ from ${}^{\star}(i', j')$ to $(i'', j'')_{\star}$ the weight of which is equal to the weight of P. This Q is defined as the concatenation $Q_1 \circ Q_2 \circ \cdots \circ Q_{|P|}$ such that Q_k with $k \leq |P|-1$ is the shortest path from ${}^{\star}P[k]$ to respectively $P[k]^{\star}$, ${}_{\star}P[k]$, or $P[k]_{\star}$, if $P[k]$ is followed horizontally, vertically, or diagonally by $P[k+1]$, and $Q_{|P|}$ is the shortest path from ${}^{\star}(i'', j'')$ to $(i'', j'')_{\star}$. (For example, the path in Figure 1(b) is such Q for the path in Figure 1(a).)

For any path Q on $\bar{G}$ from ${}^{\star}(i', j')$ to $(i'', j'')_{\star}$, Property 6 allows us to define a path on G from (i', j') to (i'', j'') whose weight is at most the weight of Q as follows. Let P be the path on G consisting of all vertices (i, j) such that the (i, j)-block of $\bar{G}$ contains at least a vertex in Q. Let Q_k with $1 \leq k \leq |P|$ denote the subpath of Q consisting of all vertices in the $P[k]$-block, so that $Q = Q_1 \circ Q_2 \circ \cdots \circ Q_{|P|}$. Let K be the sequence of all indices k with $1 \leq k \leq |P|$ such that Q_k has at last a zero-weighted vertex in ascending order, so that $K[1] = 1$. For any index k with $1 \leq k \leq |K|$, let $Q'_k = Q_{K[k]+1} \circ Q_{K[k]+1} \circ \cdots \circ Q_{\min(K[k+1]-1, |P|)}$, so that

$$Q = Q[K[1]] \circ Q'_1 \circ Q[K[2]] \circ Q'_2 \circ \cdots \circ Q[K[|K|]] \circ Q'_{|K|}.$$

Furthermore, let P'_k be the path obtained from an arbitrary shortest path on G from $P[K[k]]$ to either $P[K[k+1]]$, if $k < |K|$, or $P[|P|]$, otherwise, by deleting $P[K[k]]$ and also deleting $P[K[k+1]]$, if $k < |K|$. Define P' as the following path from (i', j') to (i'', j''):

$$P' = P[K[1]] \circ P'_1 \circ P[K[2]] \circ P'_2 \circ \cdots \circ P[K[|K|]] \circ P'_{|K|}.$$

We show that the weight of Q is at least the weight of P'. For any index k with $1 \leq k \leq |K|$, the weight of $Q_{K[k]}$ is at least the weight of $P[K[k]]$ due to Property 6. Hence, it suffices to show that, for any index k with $1 \leq k \leq |K|$, the weight of P'_k is at most the weight of Q'_k. Suppose that $k < |K|$, let $P[k] = (i_k, j_k)$, and let $P[k+1] = (i_{k+1}, j_{k+1})$. Since P'_k is obtained from a shortest path on G from $P[k]$ to $P[k+1]$ by deleting $P[k]$ and $P[k+1]$, the length $|P'_k|$ of P'_k is equal to $\max(i_{k+1} - i_k - 1, j_{k+1} - j_k - 1)$. Therefore, the weight of P'_k is at most $d|P'_k| \leq d\max(i_{k+1} - i_k - 1, j_{k+1} - j_k - 1)$. On the other hand, since Q'_k has no zero-weighted vertex, the weight of Q'_k is at least $d\max(i_{k+1} - i_k - 1, j_{k+1} - j_k - 1)$. By an analogous argument, we can verify that the weight of $P'_{|K|}$ is at most the weight of $Q'_{|K|}$. ◀

3.2 A binary vh-path grid graph representing a vhd-path one

Let G be an arbitrary $m \times n$ binary vhd-path grid graph. This section introduces an $(m + (m-1)(n-1)) \times (n + (m-1)(n-1))$ binary vh-path grid graph, denoted $\tilde{G}$, that represents the minimum path weight on G between any pair of vertices. The key idea to design such $\tilde{G}$ from G is to insert $(m-1)(n-1)$ rows and $(m-1)(n-1)$ columns of vertices, each having exactly one zero-weighted vertex called the checkpoint vertex, so that, for any path P on G, the corresponding path Q on $\tilde{G}$ contains a unique checkpoint vertex corresponding to each diagonal vertex on P. Since the number of zero-weighted vertices in Q is equal to those in P plus the number of diagonal vertices on P, the weight of P can be determined as the weight of Q minus the number of the inserted rows and columns through which Q passes.

▶ **Definition 8.** Let $\tilde{G}$ denote the $(m+(m-1)(n-1)) \times (n+(m-1)(n-1))$ binary vh-path grid graph such that the weight of each vertex is as follows. For any vertex (i, j) of G, the weight of $(n(i-1)+1, m(j-1)+1)$ on $\tilde{G}$ is set to the weight of (i, j) on G. In addition, if $i < m$ and $j < n$, then the weight of $(ni - j + 1, mj - i + 1)$ is set to zero. The weight of any other vertex is set to one. (See Figure 2.)

For any vertex (i, j) of G, we call $(n(i-1)+1, m(j-1)+1)$ on $\tilde{G}$ the (i, j)-originated vertex. Furthermore, we call $(ni - j + 1, mj - i + 1)$ the (i, j)-checkpoint vertex, if $i < m$ and $j < n$. (For example, Figure 2(b) uses a square to indicate the (i, j)-originated vertex and an open circle to indicate the (i, j)-checkpoint vertex for each vertex (i, j) in G.) The checkpoint vertices are chosen carefully so that, if a vh-path on $\tilde{G}$ passes through a pair of distinct checkpoint vertices, say the (i, j)- and (i', j')-checkpoint vertices, then neither $i = i'$ nor $j = j'$. This condition is satisfied because the (i, j)-checkpoint vertex with $j \geq 2$ is chosen from the upper row of the row of vertices where the $(i, j-1)$-checkpoint vertex is located, and the (i, j)-checkpoint vertex with $i \geq 2$ is chosen in an analogous manner. Consequently, $\tilde{G}$ has the following property.

▶ **Property 9.** *For any vertices (i, j) and (i', j') of G with $i \leq i'$ and $j \leq j'$, $\tilde{G}$ has a vh-path from the (i', j')-originated vertex to the (i'', j'')-originated vertex that passes through $\min(i'' - i', j'' - j')$ checkpoint vertices, but has no such path passing through more than $\min(i'' - i', j'' - j')$ checkpoint vertices.*

Graph $\tilde{G}$ inherently represents G in the sense as follows.

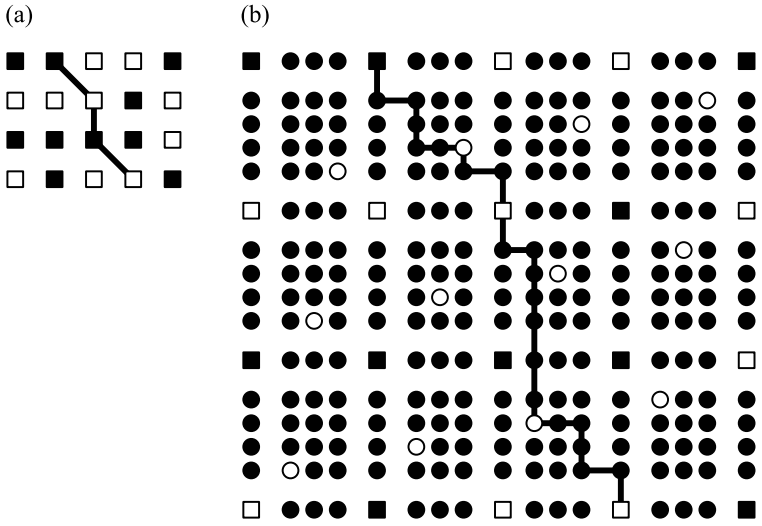


Figure 2 (a) A 4×5 binary vhd-path grid graph G and (b) the 16×17 binary vh-path grid graph $\tilde{G}$ for this G are presented, where each zero-weighted (resp. one-weighted) vertex (i, j) in G and the (i, j)-originated vertex in $\tilde{G}$ are indicated by open (resp. solid) squares, and each inserted vertices weighted by zero (resp. one) in $\tilde{G}$ is indicated by an open (resp. a solid) circle. The path indicated by a polygonal line in (b) represents the path indicated analogously in (a).

▶ **Lemma 10.** *For any vertices (i', j') and (i'', j'') of G with $i' \leq i''$ and $j' \leq j''$, the minimum path weight on G from (i', j') to (i'', j'') is equal to the minimum path weight on $\tilde{G}$ from the (i', j')-originated vertex to the (i'', j'')-originated vertex minus $(i''-i')(n-1)+(j''-j')(m-1)$.*

Proof. For any path P on G from (i', j') to (i'', j''), the weight of an arbitrary path Q on $\tilde{G}$ from the (i', j')-originated vertex to the (i'', j'')-originated vertex that contains the (i, j)-originated vertices for all vertices (i, j) in P and the (i, j)-checkpoint vertices for all diagonal vertices (i, j) on P is at most the weight of P plus $(i''-i')(n-1)+(j''-j')(m-1)$. This can be verified because the length of P is equal to $i''-i'+j''-j'+1$ minus the number of diagonal vertices on P and the length of Q is equal to $(i''-i'+j''-j'+1)+(i''-i')(n-1)+(j''-j')(m-1)$.

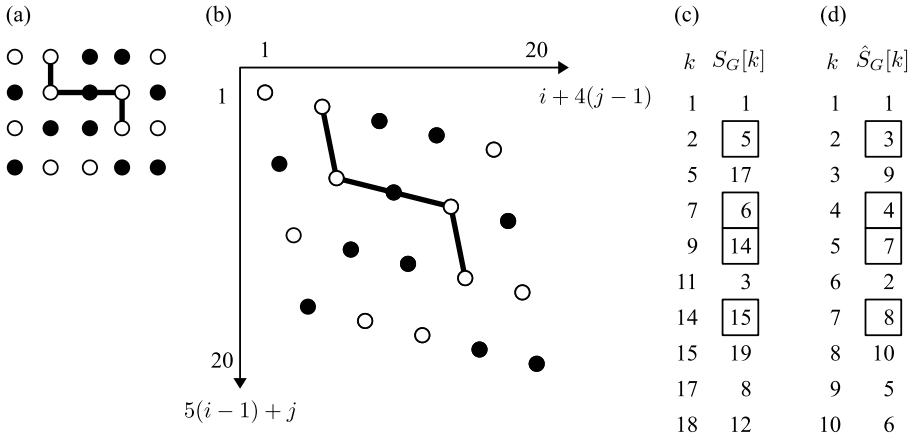
On the other hand , for any path Q on $\tilde{G}$ from the (i', j')-originated vertex to the (i'', j'')-originated vertex, Property 9 allows us to define a path P on G from (i', j') to (i'', j'') whose weight is at most the weight of Q minus $(i''-i')(n-1)+(j''-j')(m-1)$. Let K be the sequence of all indices k with $1 \leq k \leq |Q|$ such that $Q[k]$ is the (i_k, j_k)-originated vertex for some vertex, which we denote (i_k, j_k), in G in ascending order, so that $K[1] = 1$ and $K[|K|] = |Q|$. For any index k with $1 \leq k \leq |K|-1$, let $Q_k = Q[K[k]+1 : K[k+1]-1]$, so that

$$Q = Q[K[1]] \circ Q_1 \circ Q[K[2]] \circ Q_2 \circ \cdots \circ Q[K[|K|-1]] \circ Q_{|K|-1} \circ Q[K[|K|]].$$

Furthermore, let P_k be the path obtained from an arbitrary optimal path on G from (i_k, j_k) to (i_{k+1}, j_{k+1}) by deleting the first and last vertices. We define P as

$$P = (i_1, j_1) \circ P_1 \circ (i_2, j_2) \circ P_2 \circ \cdots \circ (i_{|K|-1}, j_{|K|-1}) \circ P_{|K|-1} \circ (i_{|K|}, j_{||K|}).$$

(For example, the path in Figure 2(a) satisfies the conditions of P, if we consider the path in Figure 2(b) as Q). Since the weight of $Q[K[k]]$ with $1 \leq k \leq |K|$ on $\tilde{G}$ is equal to the weight of (i_k, j_k) on G, it suffices to show that, for any index k with $1 \leq k \leq |K|-1$, the weight of



k	$S_G[k]$
1	1
2	5
5	17
7	6
9	14
11	3
14	15
15	19
17	8
18	12

k	$\hat{S}_G[k]$
1	1
2	3
3	9
4	4
5	7
6	2
7	8
8	10
9	5
10	6

Figure 3 (a) A 4×5 binary vh-path grid graph G, (b) locations $(5(i-1)+j, i+4(j-1))$ for vertices (i,j) in G, (c) the sequence S_G for G with all dummy integers omitted, and (d) the sequence $\hat{S}_G$ with $I_G = \langle 0,3,5,8,10 \rangle$ and $J_G = \langle 0,2,5,6,8,10 \rangle$ are presented, where each zero-weighted (resp. one-weighted) vertex (i,j) in G and its location $(5(i-1)+j, i+4(j-1))$ are indicated by open (resp. solid) circles. Each element $I_G[i]$ (resp. $J_G[j]$) of I_G (resp. J_G) represents the number of zero-weighted vertices, each indicated by an open circle, on the first through ith rows (resp. jth columns) of vertices in G. The paths indicated by polygonal lines in (a) and (b) and the sequences consisting of framed integers in (c) and (d) represent each other.

P_k is at most the weight of Q_k minus $(i_{k+1}-i_k)(n-1)+(j_{k+1}-j_k)(m-1)$. Since the weight of any optimal path on G from (i_k,j_k) to (i_{k+1},j_{k+1}) is at most $\max(i_{k+1}-i_k, j_{k+1}-j_k)+1$, the weight of P_k is at most $\max(i_{k+1}-i_k, j_{k+1}-j_k)-1$. On the other hand, since the number of checkpoint vertices in Q_k is at most $\min(i_{k+1}-i_k, j_{k+1}-j_k)$ due to Property 9 and the length of Q_k is $(i_{k+1}-i_k)n+(j_{k+1}-j_k)m-1$, the weight of Q_k is at least $(i_{k+1}-i_k)n+(j_{k+1}-j_k)m-1-\min(i_{k+1}-i_k, j_{k+1}-j_k) = (i_{k+1}-i_k)(n-1)+(j_{k+1}-j_k)(m-1)+\max(i_{k+1}-i_k, j_{k+1}-j_k)-1$, which completes the proof. ◀

3.3 An integer sequence representing a binary vh-path grid graph

Let G be an arbitrary $m \times n$ binary vh-path grid graph. We introduce here a sequence S_G of integers that represents the minimum path weight on G from any vertex to any by the banded LIS length of its contiguous subsequence. Each integer in S_G corresponds to a zero-weighted vertex in G. Hence, the length of S_G is equal to the number of zero-weighted vertices in G. We design S_G so that $s \leq s'$ if and only if $i \leq i'$ and $j \leq j'$, for any integers s and s' in S_G and their respectively corresponding vertices (i,j) and (i',j') in G. This strategy works because any optimal path on G maximizes the number of zero-weighted vertices contained. The minimum path weight on G is hence calculated by subtracting the banded LIS length of a contiguous subsequence of S_G from the path length.

▶ **Definition 11.** Let us introduce, for convenience, a dummy integer that is ignored when considering increasing subsequences in a sequence of integers. Let S_G denote the sequence of mn integers such that $S_G[(i-1)n+j] = i+(j-1)m$, if vertex (i,j) in G is zero-weighted, or it is set to a dummy integer, otherwise. Let $\hat{S}_G$ denote the sequence of integers obtained from S_G by deleting all dummy integers and replacing each integer remained by its rank. (See Figure 3.) Let I_G (resp. J_G) be the array of $m+1$ (resp. $n+1$) integers such that $I_G[i'']$ (resp. $J_G[j'']$) with $0 \leq i'' \leq m$ (resp. $0 \leq j'' \leq n$) is the number of zero-weighted

vertices (i, j) in G such that $(i-1)n + j \leq i''n$ (resp. $i + (j-1)m \leq j''m$), i.e., the number of zero-weighted vertices in G located on the first through i''th rows (resp. j''th columns) of vertices.

Sequences S_G and $\hat{S}_G$ inherently represent G in the sense as follows.

▶ **Lemma 12.** *For any vertices (i', j') and (i'', j'') of G with $i' \leq i''$ and $j' \leq j''$, the minimum path weight on G from (i', j') to (i'', j'') is equal to $(i'' - i') + (j'' - j') + 1$ minus the $[(j'-1)m + 1 : j''m]$-banded LIS length of $S_G[(i'-1)n + 1 : i''n]$, or equivalently, $(i''-i')+(j''-j')+1$ minus the $[J_G[j'-1]+1 : J_G[j'']]$-banded LIS length of $\hat{S}_G[I_G[i'-1]+1 : I_G[i'']]$.*

Proof. For any ℓ different zero-weighted vertices (i_k, j_k) with $1 \leq k \leq \ell$ in G, $(i_1, j_1) \circ (i_2, j_2) \circ \cdots \circ (i_\ell, j_\ell)$ is a subsequence of some path from (i', j') to (i'', j'') if and only if $(i_1 + (j_1 - 1)m) \circ (i_2 + (j_2 - 1)m) \circ \cdots \circ (i_\ell + (j_\ell - 1)m)$ is a $[(j'-1)m + 1 : j''m]$-banded increasing subsequence of $S_G[(i'-1)n + 1 : i''n]$. This is because the condition of any side holds if and only if $(i'-1)n + 1 \leq i_k \leq i''n$ and $(j'-1)m \leq j_k \leq j''m$ for any index k with $1 \leq k \leq \ell$ and $i_k \leq i_{k+1}$ and $j_k \leq j_{k+1}$ for any index k with $1 \leq k \leq \ell - 1$. The length of any path on G from (i', j') to (i'', j'') is $(i'' - i') + (j'' - j') + 1$. Thus the lemma is proven easily. ◀

3.4 The DTW distance sequence

Lemmas 3, 7, 10, and 12 naturally define the DTW distance sequence, representing the DTW distance between any $A[i' : i'']$ and any $B[j' : j'']$, as follows.

▶ **Definition 13.** Let the binary DTW distance graph for A and B be the $c^2|A| \times c^2|B|$ binary vhd-path graph $\bar{G}$ in Definition 4 for the DTW distance graph for A and B as G. Let the binary vh-path DTW distance graph for A and B be the $(c^2|A| + (c^2|A| - 1)(c^2|B| - 1)) \times (c^2|B| + (c^2|A| - 1)(c^2|B| - 1))$ binary vh-path graph $\tilde{G}$ in Definition 8 for the binary DTW distance graph for A and B as G. Consider $\hat{S}_G$, I_G, and J_G in Definition 11 for the binary vh-path DTW distance graph as G. We define the DTW distance sequence for A and B as $\hat{S}_G$ and denote it by $S_{A,B}$. As auxiliary arrays of $S_{A,B}$, let $I'_{A,B}$ (resp. $I''_{A,B}$) be the array of $|A|$ integers such that $I'_{A,B}[i] = I_G[c^4|B|(i-1)]$ (resp. $I''_{A,B}[i] = I_G[c^4|B|i+1]$), and let $J'_{A,B}$ (resp. $J''_{A,B}$) be the array of $|B|$ integers such that $J'_{A,B}[j] = J_G[c^4|A|(j-1)]$ (resp. $J''_{A,B}[j] = J_G[c^4|A|j+1]$).

Since the binary DTW distance graph consists of $c^4|A||B|$ vertices, the binary vh-path DTW distance graph contains at most $2c^4|A||B|$ zero-weighted vertices, implying that the length of the DTW distance sequence is also at most $2c^4|A||B|$. Although the binary vh-path DTW distance graph consists of $O(c^8|A|^2|B|^2)$ vertices, explicitly constructing this intermediate grid graph is unnecessary. Instead, we can construct the DTW distance sequence, together with its auxiliary arrays, from the binary DTW distance graph, or even directly from A and B, in $O(c^4|A||B|)$ time in a straightforward way.

▶ **Lemma 14.** *The DTW distance sequence, together with its auxiliary arrays, is constructible in $O(c^4|A||B|)$ time.*

Lemmas 3, 7, 10, and 12 immediately imply the following lemma.

▶ **Lemma 15.** *The DTW distance between $A[i' : i'']$ and $B[j' : j'']$ is equal to $c^2((i''-i')+(j''-j')) + 1$ minus the $[J'_{A,B}[j'] + 1 : J''_{A,B}[j'']]$-banded LIS length of $S_{A,B}[I'_{A,B}[i'] + 1 : I''_{A,B}[i'']]$.*

It is well known that the dynamic programming (DP) algorithm determines the DTW distance between any subsequences $A[i' : j'']$ and $B[j' : j'']$ in $O((i'' - i')(j'' - j'))$ time from scratch. On the other hand, even if c can be treated as a constant, it takes $O((i'' - i')(j'' - j') \log\log \ell)$ time [4] to determine the $[J'_{A,B}[j'] + 1 : J''_{A,B}[j'']]$-banded LIS length of $S_{A,B}[I'_{A,B}[i'] + 1 : I''_{A,B}[i'']]$ from $S_{A,B}$, where ℓ is the banded LIS length to be determined. Consequently, as long as we are in the situation where determining the DTW distance between any given subsequences $A[i' : i'']$ and $B[j' : j'']$ is required, naively using the DP algorithm is better than maintaining $S_{A,B}$ to apply Lemma 15. However, certain kinds of the DTW distance-related problems are relevant to the DTW distances between $A[i' : i'']$ and $B[j' : j'']$ only for restricted pairs of them, and in such cases, our elaborate representation of the DTW distances by the banded LIS lengths makes sense, as demonstrated in the next section.

4 Applications

Reduction of the DTW distance problem to the banded LIS length problem proposed in Section 3 becomes meaningful, when we apply the semi-local sequence comparison technique for a pair of sequences, developed by Tiskin [26]. Here, by semi-local we mean that any pair of an arbitrary prefix of one sequence and an arbitrary suffix of the other or any pair of an arbitrary contiguous subsequence of one and the entire sequence of the other. This technique was developed so as to be applicable to the longest common subsequence length problem and guarantees, for our particular case considering the banded LIS length, existence of the following useful permutation, which can be constructed efficiently.

▶ **Lemma 16** ([26]). *For any pair (A, B) of time series, there exists a permutation sequence $\Pi_{A,B}$ of integers from 1 to $2|S_{A,B}|$ such that, for any indices i' and i'' with $1 \le i' \le i'' \le |A|$ and any indices j' and j'' with $1 \le j' \le j'' \le |B|$,*

- *$(A[1 : i'']$ vs $B[j' : |B|])$ the $[J'_{A,B}[j']+1 : J''_{A,B}[|B|]]$-banded LIS length of $S_{A,B}[I'_{A,B}[1]+1 : I''_{A,B}[i'']]$ is equal to $J''_{A,B}[|B|] - J'_{A,B}[j']$ minus the number of indices k with $|S_{A,B}| + J'_{A,B}[j'] < k \le 2|S_{A,B}|$ and $\Pi_{A,B}(k) \le 2|S_{A,B}| - I''_{A,B}[i'']$,*
- *$(A[i' : |A|]$ vs $B[1 : j''])$ the $[J'_{A,B}[1]+1 : J''_{A,B}[j'']]$-banded LIS length of $S_{A,B}[I'_{A,B}[i']+1 : I''_{A,B}[|A|]]$ is equal to $J''_{A,B}[j''] - J'_{A,B}[1]$ minus the number of indices k with $|S_{A,B}| - I'_{A,B}[i'] < k \le 2|S_{A,B}|$ and $\Pi_{A,B}(k) \le J''_{A,B}[j'']$,*
- *$(A[i' : i'']$ vs $B[1 : |B|])$ the $[J'_{A,B}[1]+1 : J''_{A,B}[|B|]]$-banded LIS length of $S_{A,B}[I'_{A,B}[i']+1 : I''_{A,B}[i'']]$ is equal to $J''_{A,B}[|B|] - J'_{A,B}[1]$ minus the number of indices k with $|S_{A,B}| - I'_{A,B}[i'] < k \le 2|S_{A,B}|$ and $\Pi_{A,B}(k) \le 2|S_{A,B}| - I''_{A,B}[i'']$, and*
- *$(A[1 : |A|]$ vs $B[j' : j''])$ the $[J'_{A,B}[j']+1 : J''_{A,B}[j'']]$-banded LIS length of $S_{A,B}[I'_{A,B}[1]+1 : I''_{A,B}[|A|]]$ is equal to $J''_{A,B}[j''] - J'_{A,B}[j']$ minus the number of indices k with $|S_{A,B}| + J'_{A,B}[j'] < k \le 2|S_{A,B}|$ and $\Pi_{A,B}(k) \le J''_{A,B}[j'']$.*

(Once $\Pi_{A,B}$ is implemented as the two-dimensional range counting tree [3], the size of which remains $O(|S_{A,B}|)$, any of the above banded LIS lengths can be determined in $O(\log |S_{A,B}|)$ time.)

▶ **Lemma 17** ([26] with any of [27] or [22]). *If $S_{A,B}$ is available, then $\Pi_{A,B}$ in Lemma 16 can be constructed in $O(|S_{A,B}| \log^2 |S_{A,B}|)$ time and $O(|S_{A,B}|)$ space. (The two-dimensional range counting tree in Lemma 16 is constructed from $\Pi_{A,B}$ in $O(|S_{A,B}| \log |S_{A,B}|)$ time and $O(|S_{A,B}|)$ space.)*

The following DTW distance-related problems are included in typical kinds of problems efficiently handled by the semi-local sequence comparison technique. In what follows, we assume that any time series is a sequence of integers from 0 to c with $c = O(1)$. Compared

to a naive use of the DP technique, our reduction technique allows us to solve these problems asymptotically faster by an almost linear factor. The drawback of our reduction is its space-inefficiency. The DP technique requires only linear space, while ours consumes quadratic space. As a result, the algorithms we will propose for the problems based on our reduction technique balance execution speed and space consumption almost equally.

4.1 The circular DTW distance problem

Given a pair (A, B) of time series, the circular DTW distance problem consists of determining the minimum of the DTW distance between $A'' \circ A'$ and B over all partitions of A into a prefix A' and the remaining suffix A'', together with an arbitrary circular shift $A'' \circ A'$ of A that achieves this minimum distance with B. This problem may arise, for example, when we have a pair of daily temperature data for a year taken at different locations or environments and want to know the similarity and phase shift between them.

A naive algorithm can solve this problem in $O(|A|^2|B|)$ time and $O(\min(|A|, |B|))$ space by determining the DTW distance between $A'' \circ A'$ and B in $O(|A||B|)$ time based on the DP technique for each partition of A into $A' \circ A''$ and taking the minimum. In contrast, if the two-dimensional range counting tree $T_{A\circ A,B}$ for $\Pi_{A\circ A,B}$ is available, then the problem can be solved in $O(|A| \log \max(|A|, |B|))$ time by determining the DTW distance between $(A \circ A)[i : i + |A| - 1]$ $(= A[i : |A|] \circ A[1 : i - 1])$ and B in $O(\log |\Pi_{A\circ A,B}|)$ time for each index i from 1 to $|A|$ and taking the minimum. Tree $T_{A\circ A,B}$ can be constructed from scratch in $O(|A||B| \log^2 \max(|A|, |B|))$ time, because time required to construct $\Pi_{A\circ A,B}$ from $S_{A\circ A,B}$ dominates.

▶ **Theorem 18.** *Given a pair (A, B) of time series, the circular DTW distance problem can be solved in $O(|A||B| \log^2 \max(|A|, |B|))$ time and $O(|A||B|)$ space.*

4.2 The square root DTW distance problem

Given a time series A, the square root DTW distance problem consists of determining the minimum of the DTW distance of A' and A'' over all partitions of A into a prefix A' and the remaining suffix A'', together with an arbitrary prefix A' of A that achieves this distance with the remaining suffix A''. This problem may arise, for example, when we want to check if a time series can be thought of as an inexact tandem repeat.

Similarly to the case of the circular DTW distance problem, a naive algorithm can solve this problem in $O(|A|^3)$ time and $O(|A|)$ space by determining the DTW distance between A' and A'' in $O(|A|^2)$ time based on the DP technique for each partition of A into $A' \circ A''$ and taking the minimum. In contrast, if the two-dimensional range counting tree $T_{A,A}$ for $\Pi_{A,A}$ is available, then the problem can be solved in $O(|A| \log |A|)$ time by determining the DTW distance between $A[1 : i]$ and $A[i + 1 : |A|]$ in $O(\log |\Pi_{A,A}|)$ time for each index i from 1 to $|A| - 1$ and taking the minimum. Tree $T_{A,A}$ can be constructed from scratch in $O(|A|^2 \log^2 |A|)$ time.

▶ **Theorem 19.** *Given a time series A, the square root DTW distance problem can be solved in $O(|A|^2 \log^2 |A|)$ time and $O(|A|^2)$ space.*

4.3 The periodic DTW distance problem

Given a pair (A, B) of time series with $|A| \leq |B|$, the periodic DTW distance problem consists of determining the minimum of the DTW distance between A' and B over all contiguous subsequences A' of A^*, together with an arbitrary A' that achieves this distance with B,

where A^* denotes the concatenation of an infinite number of copies of A. This problem may arise, for example, when we have a time series that can be thought of as consisting of a concatenation of an unknown number of inexact occurrences of a specific pattern, possibly with the first (resp. last) occurrence being trimmed into an arbitrary nonempty suffix (resp. prefix), and want to cut it into the inexact occurrences of the pattern.

If the two-dimensional range counting tree $T_{A,B}$ for $\Pi_{A,B}$ is available, then the problem can be solved in $O(\max(|A|\log|B|,|B|)|B|\log|B|)$ time as follows. Let H be the directed acyclic graph consisting of the source vertices u_i with $1 \leq i \leq |A|$, internal vertices w_j with $1 \leq j \leq |B|-1$, the sink vertices v_i with $1 \leq i \leq |A|$,

- an edge from any source vertex u_i to any sink vertex $v_{i'}$ with $i \leq i'$, the weight of which is set to the DTW distance between $A[i:i']$ and B,
- an edge from any source vertex u_i to any internal vertex w_j, the weight of which is set to the DTW distance between $A[i:|A|]$ and $B[1:j]$,
- an edge from any internal vertex w_j to any internal vertex $w_{j'}$ with $j < j'$, the weight of which is set to the DTW distance between A and $B[j+1:j']$, and
- an edge from any internal vertex w_j to any sink vertex v_i, the weight of which is set to the DTW distance between $A[1:i]$ and $B[j+1:|B|]$.

It is easy to verify that the minimum path weight on H from a source vertex u_i to a sink vertex $v_{i'}$ passing through ℓ internal vertices is equal to the DTW distance between $A[i:|A|] \circ A^{\ell-1} \circ A[1:i']$ and B, if $\ell > 0$, or equal to the DTW distance between $A[i:i']$ and B, otherwise, and vice versa, where the weight of a path on H is the sum of the weights of all edges in the path. This implies that the periodic DTW distance problem can be solved by finding an arbitrary path on H from a source vertex to a sink vertex that has minimum weight. If H is available, then such a path can be found in time linear in the number of edges in H, which is $O(|B|^2)$, and in space linear in the number of vertices in H, which is $O(|B|)$, by determining the midpoint of the path recursively in a straightforward way. Instead of constructing H explicitly, we can use the two-dimensional range counting tree $T_{A,B}$ for $S_{A,B}$ as a data structure that supports $O(\log|B|)$-time queries of the weight of any edge in H, which allows us to obtain the path in $O(|B|^2\log|B|)$ time and $O(|B|)$ space, excluding space for storing $T_{A,B}$. Tree $T_{A,B}$ can be constructed from scratch in $O(|A||B|\log^2|B|)$ time and $O(|A||B|)$ space. (Adopting the same strategy, we can design an $O(|A||B|^2)$-time, $O(|B|)$-space algorithm based on the DP technique.)

▶ **Theorem 20.** *Given a pair (A,B) of time series with $|A| \leq |B|$, the periodic DTW distance problem can be solved in $O(\max(|A|\log|B|,|B|)|B|\log|B|)$ time and $O(|A||B|)$ space.*

5 Concluding remarks

This article showed that for any pair of time series A and B over integers $\{0,1,\ldots,c\}$, there exists a sequence S of $O(c^4|A||B|)$ integers such that the DTW distance between any contiguous subsequence of A and any contiguous subsequence of B can be represented by the banded LIS length of a contiguous subsequence of S. As applications of this reduction of DTW to LIS, novel algorithms for three DTW-related problems, the circular, square root, and periodic DTW distance problems, were presented utilizing the semi-local sequence comparison technique of Tiskin [26] originally developed for LCS-related problems.

An obvious bottleneck of our reduction technique for practical use is the quartic factor c^4 in the asymptotic length of the integer sequence proposed to represent the substring-substring DTW distances. This factor limits the application of the reduction technique only to data with few significant figures. Therefore, the immediate direction of future research

is to somehow reduce the factor to a smaller one. A potential idea to reduce the factor to quadratic is to use the same techniques as ours in a different order: the DTW distance graph $\rightarrow$ a multi-valued vh-path grid graph (conceptual) $\rightarrow$ a weighted integer sequence of length $O(|A||B|) \rightarrow$ an unweighted integer sequence of length $O(c^2|A||B|)$ that represents any substring-substring DTW distance as its band-substring LIS length. In other words, by introducing the weighted generalization of the LIS length problem as an intermediate problem in the reduction, the idea replaces the c^4-fold blowup that takes place as the first step with the c^2-fold blowup to be done as the last step. The authors are optimistic that this idea will work.

It should also be noted that there are a number of variants of the DTW distance that use different cost functions. Recall that the quartic c^4 factor in our time complexity $O(c^4|A||B|)$ comes from the quadratic cost function $(A[i_k] - B[j_k])^2$ in formula (1). If we use another major variant of the DTW distance (c.f. [2]) that is defined by replacing formula (1) in Definition 1 with

$$\sum_{k=1}^{\ell} |A[i_k] - B[j_k]|$$

with the linear cost function $|A[i_k] - B[j_k]|$, then the length of the integer sequence produced by our algorithm is bounded by $O(c^2|A||B|)$, or even by $O(c|A||B|)$ if the aforementioned improvements work.

Compared with the naive DP-based algorithms for the DTW-related problems, the proposed algorithms run asymptotically faster but consume more space. An immediate question from this time-space trade-off is whether space-inefficiency of our algorithms can be removed by reducing the required space from quadratic to linear. Another question also comes from the quadratic length of the integer sequence representing the DTW distance by its LIS length. Due to this length, there is a gap between the size of the permutation sequence used to solve the semi-local LCS and DTW problems: linear for LCS and quadratic for DTW. The fully-local LCS problem, answering queries of an LCS between any given pair of contiguous subsequences, has an interesting trade-off between space consumption and query times. That is, the DP algorithm finds an LCS from scratch in quadratic time using linear space, while a quadratic-time constructible data structure can support linear-time queries of an LCS [23]. Can we have the same trade-off also on the fully-local DTW problem? In other words, are there any quadratic-space (or even quadratic-time constructible) data structures supporting linear-time queries of the DTW distance between any pair of contiguous subsequences?

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78. IEEE, 2015.

2 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 79–97. IEEE, 2015.

3 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988.

4 Maxime Crochemore and Ely Porat. Fast computation of a longest increasing subsequence and application. *Information and computation*, 208(9):1054–1059, 2010.

5 Marc Dupont and Pierre-François Marteau. Coarse-DTW for sparse time series alignment. In *International Workshop on Advanced Analysis and Learning on Temporal Data*, pages 157–172. Springer, 2015.

6 Vincent Froese, Brijnesh Jain, Maciej Rymar, and Mathias Weller. Fast exact dynamic time warping on run-length encoded time series. *arXiv preprint*, 2020. `arXiv:1903.03003`.

7 Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Transactions on Algorithms (TALG)*, 14(4):50:1–50:17, 2018.

8 Daniel S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

9 Y Hwang and SB Gelfand. Binary sparse dynamic time warping. In *Proceedings of the 15th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'19)*, volume 2, pages 748–759, 2019.

10 Youngha Hwang and Saul B Gelfand. Sparse dynamic time warping. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 163–175. Springer, 2017.

11 Zahedeh Izakian, Mohammad Saadi Mesgari, and Ajith Abraham. Automated clustering of trajectory data using a particle swarm optimization. *Computers, Environment and Urban Systems*, 55:55–65, 2016.

12 Jyh-Shing Roger Jang and Hong-Ru Lee. Hierarchical filtering method for content-based music retrieval via acoustic input. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 401–410, 2001.

13 Pat Jangyodsuk, Christopher Conly, and Vassilis Athitsos. Sign language recognition using dynamic time warping and hand shape distance based on histogram of oriented gradient features. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, pages 50:1–50:6, 2014.

14 Jingyu Jiang, Yuan Xing, Shuxin Wang, and Ke Liang. Evaluation of robotic surgery skills using dynamic time warping. *Computer methods and programs in biomedicine*, 152:71–83, 2017.

15 Benjamin Johnen and Bernd Kuhlenkötter. A dynamic time warping algorithm for industrial robot motion analysis. In *2016 Annual Conference on Information Science and Systems (CISS)*, pages 18–23. IEEE, 2016.

16 Eamonn Keogh and Chotirat Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, 2005.

17 William Kuszmaul. Dynamic time warping in strongly subquadratic time: Algorithms for the low-distance regime and approximate evaluation. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, pages 80:1–80:15, 2019.

18 Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *arXiv preprint*, 2010. `arXiv:1003.4083`.

19 Abdullah Mueen, Nikan Chavoshi, Noor Abu-El-Rub, Hossein Hamooni, and Amanda Minnich. Awarp: fast warping distance for sparse time series. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 350–359. IEEE, 2016.

20 Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

21 Toni M Rath and Raghavan Manmatha. Word image matching using dynamic time warping. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages 521–527. IEEE, 2003.

22 Yoshifumi Sakai. A fast algorithm for multiplying min-sum permutations. *Discrete applied mathematics*, 159(17):2175–2183, 2011.

23 Yoshifumi Sakai. A substring–substring LCS data structure. *Theoretical Computer Science*, 753:16–34, 2019.

24 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

25 Charles C. Tappert, Ching Y. Suen, and Toru Wakahara. The state of the art in online handwriting recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 12(8):787–808, 1990.

26 Alexander Tiskin. Semi-local string comparison: Algorithmic techniques and applications. *Mathematics in Computer Science*, 1(4):571–603, 2008.

27 Alexander Tiskin. Fast distance multiplication of unit-Monge matrices. *Algorithmica*, 71(4):859–888, 2015.

28 Neil Vaughan and Bogdan Gabrys. Comparing and combining time series trajectories using dynamic time warping. *Procedia Computer Science*, 96:465–474, 2016.

29 Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.