Dynamic Time Warping in Strongly Subquadratic Time: Algorithms for the Low-Distance Regime and Approximate Evaluation

William Kuszmaul
Massachusetts Institute of Technology, Cambridge, USA
kuszmaul@mit.edu

Abstract

Dynamic time warping distance (DTW) is a widely used distance measure between time series, with applications in areas such as speech recognition and bioinformatics. The best known algorithms for computing DTW run in near quadratic time, and conditional lower bounds prohibit the existence of significantly faster algorithms.

The lower bounds do not prevent a faster algorithm for the important special case in which the DTW is small, however. For an arbitrary metric space Σ with distances normalized so that the smallest non-zero distance is one, we present an algorithm which computes $\mathrm{dtw}(x, y)$ for two strings x and y over Σ in time $O(n \cdot \mathrm{dtw}(x, y))$. When $\mathrm{dtw}(x, y)$ is small, this represents a significant speedup over the standard quadratic-time algorithm.

Using our low-distance regime algorithm as a building block, we also present an approximation algorithm which computes $\mathrm{dtw}(x, y)$ within a factor of $O(n^{\epsilon})$ in time $\tilde{O}(n^{2-\epsilon})$ for $0 < \epsilon < 1$. The algorithm allows for the strings x and y to be taken over an arbitrary well-separated tree metric with logarithmic depth and at most exponential aspect ratio. Notably, any polynomial-size metric space can be efficiently embedded into such a tree metric with logarithmic expected distortion. Extending our techniques further, we also obtain the first approximation algorithm for edit distance to work with characters taken from an *arbitrary metric space*, providing an n^{ϵ}-approximation in time $\tilde{O}(n^{2-\epsilon})$, with high probability.

Finally, we turn our attention to the relationship between edit distance and dynamic time warping distance. We prove a reduction from computing edit distance over an arbitrary metric space to computing DTW over the same metric space, except with an added null character (whose distance to a letter l is defined to be the edit-distance insertion cost of l). Applying our reduction to a conditional lower bound of Bringmann and Künnemann pertaining to edit distance over $\{0, 1\}$, we obtain a conditional lower bound for computing DTW over a three letter alphabet (with distances of zero and one). This improves on a previous result of Abboud, Backurs, and Williams, who gave a conditional lower bound for DTW over an alphabet of size five.

With a similar approach, we also prove a reduction from computing edit distance (over generalized Hamming Space) to computing longest-common-subsequence length (LCS) over an alphabet with an added null character. Surprisingly, this means that one can recover conditional lower bounds for LCS directly from those for edit distance, which was not previously thought to be the case.

2012 ACM Subject Classification Theory of computation; Theory of computation → Design and analysis of algorithms

Keywords and phrases dynamic time warping, edit distance, approximation algorithm, tree metrics

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.80

Category Track A: Algorithms, Complexity and Games

Related Version `https://arxiv.org/abs/1904.09690`

Funding *William Kuszmaul*: Supported by an MIT Akamai Fellowship and a Fannie & John Hertz Foundation Fellowship. Also supported by NSF Grants 1314547 and 1533644. Parts of this research were performed during the Stanford CURIS research program.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 80; pp. 80:1–80:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements The author would like to thank Moses Charikar for his mentoring and advice throughout the project, Ofir Geri for his support and for many useful conversations, and Virginia Williams for suggesting the problem of reducing between edit distance and LCS.

1 Introduction

Dynamic Time Warping distance (DTW) is a widely used distance measure between time series. DTW is particularly flexible in dealing with temporal sequences that vary in speed. To measure the distance between two sequences, portions of each sequence are allowed to be warped (meaning that a character may be replaced with multiple consecutive copies of itself), and then the warped sequences are compared by summing the distances between corresponding pairs of characters. DTW's many applications include phone authentication [19], signature verification [35], speech recognition [34], bioinformatics [1], cardiac medicine [15], and song identification [44].

The textbook dynamic-programming algorithm for DTW runs in time $O(n^2)$, which can be prohibitively slow for large inputs. Moreover, conditional lower bounds [13, 3] prohibit the existence of a strongly subquadratic-time algorithm[1], unless the Strong Exponential Time Hypothesis is false.

The difficulty of computing DTW directly has motivated the development of fast heuristics [39, 28, 29, 27, 11, 38] which typically lack provable guarantees.

On the theoretical side, researchers have considered DTW in the contexts of locality sensitive hashing [20] and nearest neighbor search [21], but very little additional progress has been made on the general problem in which one is given two strings x and y with characters from a metric space Σ, and one wishes to compute (or approximate) $\mathrm{dtw}(x, y)$.

To see the spectrum of results one might aim for, it is helpful to consider edit distance, another string-similarity measure, defined to be the minimum number of insertions, deletions, and substitutions needed to get between two strings. Like DTW, edit distance can be computed in time $O(n^2)$ using dynamic programming [42, 37, 41], and conditional lower bounds suggest that no algorithm can do significantly better [7, 13]. This has led to researchers focusing on specialized versions of the problem, especially in two important directions:

- **Low-Distance Regime Algorithms:** In the special case where the edit distance between two strings is small (less than $\sqrt{n}$), the algorithm of Landau, Myers and Schmidt [33] can be used to compute the exact distance in time $O(n)$. In general, the algorithm runs in time $O(n + \mathrm{ed}(x, y)^2)$. Significant effort has also been made to design variants of the algorithm which exhibit small constant overhead in practice [17].
- **Approximation Algorithms:** Andoni, Krauthgamer and Onak introduced an algorithm estimating edit distance within a factor of $(\log n)^{O(1/\varepsilon)}$ in time $O(n^{1+\varepsilon})$ [5], culminating a long line of research on approximation algorithms that run in close to linear time [6, 10, 9, 18]. Recently, Chakraborty et al. gave the first strongly subquadratic algorithm to achieve a *constant* approximation, running time $O(n^{12/7})$ [16].

This paper presents the first theoretical results in these directions for DTW.

[1] An algorithm is said to run in strongly subquadratic time if it runs in time $O(n^{2-\epsilon})$ for some constant $\epsilon > 0$. Although strongly subquadratic time algorithms are prohibited by conditional lower bounds, runtime improvements by subpolynomial factors are not. Such improvements have been achieved [25].

A Low-Distance Regime Algorithm for DTW (Section 3)

We present the first algorithm for computing DTW in the low-distance regime. Our algorithm computes $\mathrm{dtw}(x, y)$ in time $O(n \cdot \mathrm{dtw}(x, y))$ for strings x and y with characters taken from an arbitrary metric space in which the minimum non-zero distance is one. The key step in our algorithm is the design of a new dynamic-programming algorithm for DTW, which lends itself especially well to the low-distance setting.

Our dynamic program relies on a recursive structure in which the two strings x and y are treated asymmetrically within each subproblem: One of the strings is considered as a sequence of letters, while the other string is considered as a sequence of runs (of equal letters). The subproblems build on one-another in a way so that at appropriate points in the recursion, we toggle the role that the two strings play. The asymmetric treatment of the strings limits the number of subproblems that can have return-values less than any given threshold K to $O(nK)$, allowing for a fast algorithm in the low-distance setting.

We remark that the requirement of having the smallest distance between distinct characters be 1 is necessary for the low-distance regime algorithm to be feasible, since otherwise distances can simply be scaled down to make every DTW instance be low-distance.

Approximating DTW Over Well Separated Tree Metrics (Section 4)

We design the first approximation algorithm for DTW to run in strongly subquadratic time. Our algorithm computes $\mathrm{dtw}(x, y)$ within an n^{ϵ}-approximation in time $\tilde{O}(n^{2-\epsilon})$. The algorithm allows for the strings x and y have characters taken from an arbitrary well-separated tree metric of logarithmic depth and at most exponential aspect ratio.[2] These metric spaces are universal in the sense that any finite metric space M of polynomial size can be efficiently embedded into a well-separated tree metric with expected distortion $O(\log |M|)$ and logarithmic depth [22, 8].

An important consequence of our approximation algorithm is for the special case of DTW over the reals. Exploiting a folklore embedding from $\mathbb{R}$ to a well-separated tree metric metric, we are able to obtain with high probability an $O(n^{\epsilon})$-approximation for $\mathrm{dtw}(x, y)$ in time $\tilde{O}(n^{2-\epsilon})$, for any strings x and y of length at most n over a subset of the reals with a polynomial aspect ratio.

In the special case of DTW over the reals, previous work has been done to find approximation algorithms under certain geometric assumptions about the inputs x and y [4, 43]. To the best of our knowledge, our approximation algorithm is the first to not rely on any such assumptions.

It is interesting to note that our results on low-distance regime and approximation algorithms for DTW have bounds very similar to the earliest results for edit distance in the same directions. Indeed, the first algorithm to compute edit distance in the low-distance regime [23] exploited properties of a (now standard) dynamic-programming algorithm in order to compute $\mathrm{ed}(x, y)$ in time $O(n \cdot \mathrm{ed}(x, y))$. This implicitly resulted in the first approximation algorithm for edit distance, allowing one to compute an $O(n^{\epsilon})$-approximation in time $O(n^{2-\epsilon})$. Until the work of [6] and [5], which culminated in an algorithm with a polylogarithmic approximation ratio, the best known approximation ratio for edit distance remained polynomial for roughly twenty years [33, 9, 10].

[2] The *aspect* ratio of a metric space is the ratio between the largest and smallest non-zero distances in the space.

The $\tilde{O}(n^{2-\epsilon})$-time $O(n^\epsilon)$-approximation tradeoff is also the current state-of-the-art for another related distance measurement known as Fréchet distance [14], and is achieved using an algorithm that differs significantly from its edit-distance and DTW counterparts.

Reduction from Edit Distance to DTW (Section 5)

We show that the similarity between our results for DTW and the earliest such results for edit distance is not coincidental. In particular, we prove a simple reduction from computing edit distance over an arbitrary metric space to computing DTW over the same metric space (with an added null character). Consequently, any algorithmic result for computing DTW in the low-distance regime or approximating DTW immediately implies the analogous result for edit distance. The opposite direction is true for lower bounds. For example, the conditional lower bound of Bringmann and Künnemann [13], which applies to edit distance over the alphabet $\{0, 1\}$, now immediately implies a conditional lower bound for DTW over an alphabet of size three (in which characters are compared with distances zero and one). This resolves a direction of work posed by Abboud, Backurs, and Williams [3], who gave a conditional lower bound for DTW over an alphabet of size five, and noted that if one could prove the same lower bound for an alphabet of size three, then the runtime complexity of DTW over generalized Hamming space would be settled (modulo the Strong Exponential Time Hypothesis). Indeed, it is known that over an alphabet of size two, DTW can be computed in strongly subquadratic time [3].

Using a similar approach we also prove a simple reduction from computing edit distance (over generalized Hamming space) to computing the longest-common-subsequence length (LCS) between two strings. Thus conditional lower bounds for computing edit distance directly imply conditional lower bounds for computing LCS (over an alphabet with one additional character). This was not previously though to be the case. Indeed, the first known conditional lower bounds for LCS came after those for edit distance [2, 7], and it was noted by Abboud et al. [2] that *"A simple observation is that the computation of the LCS is equivalent to the computation of the Edit-Distance when only deletions and insertions are allowed, but no substitutions. Thus, intuitively, LCS seems like an easier version of Edit Distance, since a solution has fewer degrees of freedom, and the lower bound for Edit-Distance does not immediately imply any hardness for LCS."* Our reduction violates this intuition by showing that edit distance without substitutions can be used to efficiently simulate edit distance without substitutions.

Approximating Edit Distance Over an Arbitrary Metric (Section 6)

The aforementioned results for approximating edit distance [23, 9, 10, 33, 6, 5, 16, 32] consider only the case in which insertion, deletion, and substitution costs are all constant. To the best of our knowledge, no approximation algorithm is known for the more general case in which characters are taken from an arbitrary metric space and edit costs are assigned based on metric distances between characters. This variant of edit distance is sometimes referred to as *general edit distance* [36]. The study of general edit distance dates back to the first papers on edit distance [42, 40], and allowing for nonuniform costs is important in many applications, including in computational biology [26].

We present an approximation algorithm for edit distance over an an arbitrary metric. Our algorithm runs in time $\tilde{O}(n^{2-\epsilon})$ and computes an $O(n^\epsilon)$-approximation for $\mathrm{ed}(x, y)$ with high probability. Note that for the case where characters are taken from a well-separated tree metric with logarithmic depth and at most exponential aspect ratio, the result already follows from our approximation algorithm for DTW, and our reduction from edit distance to DTW. The approach taken in Section 6 is particularly interesting in that it places no restrictions on the underlying metric space.

Both our approximation algorithm for DTW and our approximation algorithm for edit distance exhibit relatively weak runtime/approximation tradeoffs. To the best of our knowledge, however, they are the first such algorithms to run in strongly subquadratic time.

2 Preliminaries

In this section, we present preliminary definitions and background on dynamic time warping distance (DTW) and edit distance.

Dynamic Time Warping Distance

For a metric space Σ, the dynamic time warping distance (DTW) between two strings $x, y \in \Sigma^n$ is a natural measure of similarity between the strings.

Before fully defining DTW, we first introduce the notion of an *expansion* of a string.

▶ **Definition 2.1.** *The* runs *of a string $x \in \Sigma^n$ are the maximal subsequences of consecutive letters with the same value. One can* extend *a run by replacing it with a longer run of the same letter. An* expansion *of the string x is any string which can be obtained from x by extending runs.*

As an example, consider $x = aaaccbbd$. Then the runs of x are aaa, cc, bb, and d. The string $\overline{x} = aaacccccbbdd$ is an expansion of x and extends the runs containing c and d.

Using the terminology of expansions, we now define DTW.

▶ **Definition 2.2.** *Consider strings x and y of length n over a metric (Σ, d). A* correspondence *$(\overline{x}, \overline{y})$ between x and y is a pair of equal-length expansions $\overline{x}$ of x and $\overline{y}$ of y. The* cost *of a correspondence is given by $\sum_i d(\overline{x}_i, \overline{y}_i)$.*

The dynamic time warping distance $\mathrm{dtw}(x, y)$ *is defined to be the minimum cost of a correspondence between x and y.*

When referring to a run r in one of x or y, and when talking about a correspondence $(\overline{x}, \overline{y})$, we will often use r to implicitly refer to the extended run corresponding with r in the correspondence. Whether we are referring to the original run or the extended version of the run should be clear from context.

Note that any minimum-length optimal correspondence between strings $x, y \in \Sigma^n$ will be of length at most $2n$. This is because if a run r_1 in x overlaps a run r_2 in y in the correspondence, then we may assume without loss of generality that at most one of the two runs is extended by the correspondence. (Otherwise, we could un-extend each run by one and arrive at a shorter correspondence with no added cost.)

Edit Distance Over an Arbitrary Metric

The *simple edit distance* between two strings x and y is the minimum number of insertions, deletions, and substitutions needed to transform x into y. In this paper we will mostly focus on a more general variant of edit distance, in which characters are taken from an arbitrary metric:

▶ **Definition 2.3.** *Let x and y be strings over an alphabet Σ, where $(\Sigma \cup \{\emptyset\}, d)$ is a metric space. We say that the* magnitude $|l|$ *of a letter $l \in \Sigma$ is $d(\emptyset, l)$. We define the* edit distance *between x and y to be the minimum cost of a sequence of edits from x to y, where the insertion or deletion of a letter l costs $d(\emptyset, l)$, and the substitution of a letter l to a letter l' costs $d(l, l')$.*

3 Computing DTW in the Low-Distance Regime

In this section, we present a low-distance regime algorithm for DTW (with characters from an arbitrary metric in which all non-zero distances are at least one). Given that $\mathrm{dtw}(x, y)$ is bounded above by a parameter K, our algorithm can compute $\mathrm{dtw}(x, y)$ in time $O(nK)$. Moreover, if $\mathrm{dtw}(x, y) > K$, then the algorithm will conclude as much. Consequently, by doubling our guess for K repeatedly, one can compute $\mathrm{dtw}(x, y)$ in time $O(n \cdot \mathrm{dtw}(x, y))$.

Consider x and y of length n with characters taken from a metric space Σ in which all non-zero distances are at least one. In the textbook dynamic program for DTW [30], each pair of indices $i, j \in [n]$ represents a subproblem $T(i, j)$ whose value is $\mathrm{dtw}(x[1 : i], y[1 : j])$. Since $T(i, j)$ can be determined using $T(i-1, j), T(i, j-1), T(i-1, j-1)$, and knowledge of x_i and y_j, this leads to an $O(n^2)$ algorithm for DTW. A common heuristic in practice is to construct only a small band around the main diagonal of the dynamic programming grid; by computing only entries $T(i, j)$ with $|i - j| \le 2K$, and treating other subproblems as having infinite return values, one can obtain a correct computation for DTW as long as there is an optimal correspondence which matches only letters which are within K of each other in position. This heuristic is known as the Sakoe-Chiba Band [39] and is employed, for example, in the commonly used library of Giorgino [24].

The Sakoe-Chiba Band heuristic can perform badly even when $\mathrm{dtw}(x, y)$ is very small, however. Consider $x = abbb\cdots b$ and $y = aaa\cdots ab$. Although $\mathrm{dtw}(x, y) = 0$, if we restrict ourselves to matching letters within K positions of each other for some small K, then the resulting correspondence will cost $\Omega(n)$.

In order to obtain an algorithm which performs well in the low-distance regime, we introduce a new dynamic program for DTW. The new dynamic program treats x and y asymmetrically within each subproblem. Loosely speaking, for indices i and j, there are two subproblems $\mathrm{SP}(x, y, i, j)$ and $\mathrm{SP}(y, x, i, j)$. The first of these subproblems evaluates to the DTW between the first i runs of x and the first j letters of y, with the added condition that the final run of $y[1 : j]$ is not extended. The second of the subproblems is analogously defined as the DTW between the first i runs of y and the first j letters of x with the added condition that the final run of $x[1 : j]$ is not extended.

The recursion connecting the new subproblems is somewhat more intricate than for the textbook dynamic program. By matching the i-th run with the j-th letter, however, we limit the number of subproblems which can evaluate to less than K. In particular, if the j-th letter of y is in y's t-th run, then any correspondence which matches the i-th run of x to the j-th letter of y must cost at least $\Omega(|i - t|)$. (This is formally shown in the extended paper [31].) Thus for a given j, there are only $O(K)$ options for i such that $\mathrm{SP}(x, y, i, j)$ can possibly be at most K, and similarly for $\mathrm{SP}(y, x, i, j)$. Since we are interested in the case of $\mathrm{dtw}(x, y) \le K$, we can restrict ourselves to the $O(nK)$ subproblems which have the potential to evaluate to at most $O(K)$. Notice that, in fact, our algorithm will work even when $\mathrm{dtw}(x, y) > K$ as long as there is an optimal correspondence between x and y which only matches letters from x from the r_x-th run with letters from y from the r_y-th run if $|r_x - r_y| \le O(K)$.

Formally we define our recursive problems in a manner slightly different from that described above. Let x and y be strings of length at most n and let K be a parameter which we assume is greater than $\text{dtw}(x, y)$. Our subproblems will be the form $\text{SP}(x, y, r_x, r_y, o_y)$, which is defined as follows. Let x' consist of the first r_x runs of x and y' consist of the first r_y runs of y until the o_y-th letter in the r_y-th run. Then $\text{SP}(x, y, r_x, r_y, o_y)$ is the value of the optimal correspondence between x' and y' such that the r_y-th run in y' is not extended.[3] If no such correspondence exists (which can only happen if $r_y \leq 1$ or $r_x = 0$), then the value of the subproblem is ∞. Note that we allow r_x, r_y, o_y to be zero, and if r_y is zero, then o_y must be zero as well. We also consider the symmetrically defined subproblems of the form $\text{SP}(y, x, r_y, r_x, o_x)$. We will focus on the subproblems of the first types, implicitly treating subproblems of the second type symmetrically.

▶ Example 3.1. Suppose characters are taken from generalized Hamming space, with distances of 0 and 1. The subproblem $\text{SP}(efabbccccd, ffaabcccddd, 5, 4, 2)$ takes the value of the optimal correspondence between $efabbcccc$ and $ffaabcc$ such that the final cc run in the latter is not extended. The subproblem's value turns out to be 3, due to the correspondence:

$$\begin{array}{cccccccccc} \text{e} & \text{f} & \text{a} & \text{a} & \text{b} & \text{b} & \text{c} & \text{c} & \text{c} & \text{c} \\ \text{f} & \text{f} & \text{a} & \text{a} & \text{b} & \text{b} & \text{b} & \text{b} & \text{c} & \text{c} \end{array} .$$

The next lemma presents the key recursive relationship between subproblems. The lemma focuses on the case where $r_x, r_y, o_y \geq 1$.

▶ **Lemma 3.2.** *Suppose that r_x and r_y are both between 1 and the number of runs in x and y respectively; and that o_y is between 1 and the length of the r_y-th run in y. Let l_x be the length of the r_x-th run in x and l_y be the length of the r_y-th run in y. Let d be the distance between the letter populating the r_x-th run in x and the letter populating the r_y-th run in y. Then $\text{SP}(x, y, r_x, r_y, o_y)$ is given by*

$$\begin{cases} \min\left(\text{SP}(x, y, r_x, r_y, o_y - 1) + d, \text{SP}(x, y, r_x - 1, r_y, o_y - l_x) + d \cdot l_x\right) & \text{if } l_x \leq o_y \\ \min\left(\text{SP}(x, y, r_x, r_y, o_y - 1) + d, \text{SP}(y, x, r_y - 1, r_x, l_x - o_y) + d \cdot o_y\right) & \text{if } l_x > o_y. \end{cases}$$

Proof. Consider a minimum-cost correspondence A between the first r_x runs of x and the portion of y up until the o_y-th letter in the r_y-th run, such that the r_y-th run in y is not extended.

If the r_x-th run in A is extended, then the cost of A will be $\text{SP}(x, y, r_x, r_y, o_y - 1) + d$. If the r_x-th run in A is not extended, then we consider two cases.

In the first case, $l_x \leq o_y$. In this case, the entirety of the r_x-th run of x is engulfed by the r_y-th run of y in the correspondence A. Since the r_x-th run is not extended, the cost of the overlap is $l_x \cdot d$. Thus the cost of A must be $\text{SP}(x, y, r_x - 1, r_y, o_y - l_x) + d \cdot l_x$.

Moreover, since A is minimum-cost, as long as $l_x \leq o_y$, the cost of A is at most the above expression, regardless of whether the r_x-th run in x is extended in A.

In the second case, $l_x > o_y$. In this case, the first o_y letters in the r_y-th run of y all overlap the r_x-th run of x in A. Since the r_y-th run is not extended, the cost of the overlap is $d \cdot o_y$. Thus, since the r_x-th run in x is also not extended in A, the cost of A must be $\text{SP}(y, x, r_y - 1, r_x, l_x - o_y) + d \cdot o_y$. Moreover, since A is minimal, as long as $l_x > o_y$, the cost of A is at most the above expression, regardless of whether the r_x-th run in x is extended. ◀

[3] If $o_y = 0$, then the r_y-th run in y' is empty and thus trivially cannot be extended.

The above lemma handles cases where $r_x, r_y, o_y > 0$. In the case where $r_x > 0$, $r_y > 0$, and $o_y = 0$, $\mathrm{SP}(x, y, r_x, r_y, o_y)$ is just the dynamic time warping distance between the first r_x runs of x and the first $r_y - 1$ runs of y, given by $\min\left(\mathrm{SP}(x, y, r_x, r_y - 1, t_1), \mathrm{SP}(y, x, r_y - 1, r_x, t_2)\right)$, where t_1 is the length of the $(r_y - 1)$-th run in y and t_2 is the length of the r_x-th run in x. The remaining cases are edge-cases with $\mathrm{SP}(x, y, r_x, r_y, o_y) \in \{0, \infty\}$. (See the extended paper [31].)

One can show that any correspondence A in which a letter from the r_x-th run of x is matched with a letter from the r_y-th run of y must contain must contain at least $\frac{|r_x - r_y| - 1}{2}$ instances of unequal letters being matched. It follows that if $\mathrm{dtw}(x, y) \leq K$, then we can limit ourselves to subproblems in which $|r_x - r_y| \leq O(K)$. For each of the n options of (r_y, o_y), there are only $O(K)$ options of r_x that must be considered. This limits the total number of subproblems to $O(nK)$. The resulting dynamic program yields the following theorem, the full proof of which appears in the extended paper [31].

▶ **Theorem 3.3.** *Let x and y be strings of length n taken from a metric space Σ with minimum non-zero distance at least one, and let K be parameter such that $\mathrm{dtw}(x, y) \leq K$. Then there exists a dynamic program for computing $\mathrm{dtw}(x, y)$ in time $O(nK)$. Moreover, if $\mathrm{dtw}(x, y) > K$, then the dynamic program will return a value greater than K.*

By repeatedly doubling one's guess for K until the computed value of $\mathrm{dtw}(x, y)$ evaluates to less than K, one can therefore compute $\mathrm{dtw}(x, y)$ in time $O(n \cdot \mathrm{dtw}(x, y))$.

4 Approximating DTW Over Well-Separated Tree Metrics

In this section, we present an $\tilde{O}(n^{2-\epsilon})$-time $O(n^{\epsilon})$-approximation algorithm for DTW over a well-separated tree metric with logarithmic depth. We begin by presenting a brief background on well-separated tree metrics.

▶ **Definition 4.1.** *Consider a tree T whose vertices form an alphabet Σ, and whose edges have positive weights. T is said to be a* well-separated tree metric *if every root-to-leaf path consists of edges ordered by nonincreasing weight. The* distance *between two nodes $u, v \in \Sigma$ is defined as the maximum weight of any edge in the shortest path from u to v.*

Well-separated tree metrics are universal in the sense that any metric Σ can be efficiently embedded (in time $O(|\Sigma|^2)$) into a well-separated tree metric T with expected distortion $O(\log |\Sigma|)$ [22]. Moreover, the tree metric may be made to have logarithmic depth using Theorem 8 of [8]. For strings $x, y \in \Sigma^n$, let $\mathrm{dtw}_T(x, y)$ denote the dynamic time warping distance after embedding Σ into T. Then the tree-metric embedding guarantees that $\mathrm{dtw}(x, y) \leq \mathrm{dtw}_T(x, y)$ and that $\mathbb{E}[\mathrm{dtw}_T(x, y)] \leq O(\log n) \cdot \mathrm{dtw}(x, y)$. (The latter fact may be slightly nonobvious and is further explained in the extended paper [31].)

It follows that any approximation algorithm for DTW over well-separated tree metrics will immediately yield an approximation algorithm over an arbitrary polynomial-size metric Σ, with two caveats: the new algorithm will have its multiplicative error increased by $O(\log n)$; and $O(\log n)$ instances of Σ embedded into a well-separated tree metric must be precomputed for use by the algorithm (requiring, in general, $O(|\Sigma|^2 \log n)$ preprocessing time). In particular, given $O(\log n)$ tree embeddings of Σ, $T_1, \ldots, T_{O(\log n)}$, with high probability $\min_i \left(\mathrm{dtw}_{T_i}(x, y)\right)$ will be within a logarithmic factor of $\mathrm{dtw}(x, y)$.

The remainder of the section will be devoted to designing an approximation algorithm for DTW over a well-separated tree metric. We will prove the following theorem:

▶ **Theorem 4.2.** *Consider* $0 < \epsilon < 1$. *Suppose that* Σ *is a well-separated tree metric of polynomial size and at most logarithmic depth. Moreover, suppose that the aspect ratio of* Σ *is at most exponential in* n *(i.e., the ratio between the largest distance and the smallest non-zero distance). Then in time* $\tilde{O}(n^{2-\epsilon})$ *we can obtain an* $O(n^\epsilon)$*-approximation for* $\mathrm{dtw}(x, y)$ *for any* $x, y \in \Sigma^n$.

An important consequence of the theorem occurs for DTW over the reals. When Σ is an $O(n)$-point subset of the reals with a polynomial aspect ratio, there exists an $O(n \log n)$-time embedding with $O(\log n)$ expected distortion from Σ to a well-separated tree metric of size $O(n)$ with logarithmic depth. (This is further discussed in the extended paper [31].). This gives the following corollary:

▶ **Corollary 4.3.** *Consider* $0 < \epsilon < 1$. *Suppose that* $\Sigma = [0, n^c] \cap \mathbb{Z}$ *for some constant* c. *Then in time* $\tilde{O}(n^{2-\epsilon})$ *we can obtain an* $O(n^\epsilon)$*-approximation for* $\mathrm{dtw}(x, y)$ *with high probability for any* $x, y \in \Sigma^n$.

In proving Theorem 4.2, our approximation algorithm will take advantage of what we refer to as the r*-simplification* of a string over a well-separated tree metric.

▶ **Definition 4.4.** *Let* T *be a well-separated tree metric whose nodes form an alphabet* Σ. *For a string* $x \in \Sigma^n$, *and for any* $r \geq 1$, *the* r-simplification $s_r(x)$ *is constructed by replacing each letter* $l \in x$ *with its highest ancestor* l' *in* T *that can be reached from* l *using only edges of weight at most* $r/4$.

Our approximation algorithm will apply the low-distance regime algorithm from the previous section to $s_r(x)$ and $s_r(y)$ for various r in order to extract information about $\mathrm{dtw}(x, y)$. Notice that using our low-distance regime algorithm for DTW, we get the following useful lemma for free:

▶ **Lemma 4.5.** *Consider* $0 < \epsilon < 1$. *Suppose that for all pairs* l_1, l_2 *of distinct letters in* Σ, $d(l_1, l_2) \geq \gamma$. *Then for* $x, y \in \Sigma^n$ *there is an* $O(n^{2-\epsilon})$ *time algorithm which either computes* $\mathrm{dtw}(x, y)$ *exactly, or concludes that* $\mathrm{dtw}(x, y) > \gamma n^{1-\epsilon}$.

The next lemma states three important properties of r-simplifications. We remark that the same lemma appears in our concurrent work on the communication complexity of DTW, in which we use the lemma in designing an efficient one-way communication protocol [12].

▶ **Lemma 4.6.** *Let* T *be a well-separated tree metric with distance function* d *and whose nodes form the alphabet* Σ. *Consider strings* x *and* y *in* Σ^n.

Then the following three properties of $s_r(x)$ *and* $s_r(y)$ *hold:*

- *For every letter* $l_1 \in s_r(x)$ *and every letter* $l_2 \in s_r(y)$, *if* $l_1 \neq l_2$, *then* $d(l_1, l_2) > r/4$.
- *For all* α, *if* $\mathrm{dtw}(x, y) \leq nr/\alpha$ *then* $\mathrm{dtw}(s_r(x), s_r(y)) \leq nr/\alpha$.
- *If* $\mathrm{dtw}(x, y) > nr$, *then* $\mathrm{dtw}(s_r(x), s_r(y)) > nr/2$.

The first and second parts of Lemma 4.6 are straightforward from the definitions of $s_r(x)$ and $s_r(y)$. The third part follows from the observation that a correspondence C between x and y can cost at most $|C| \cdot \frac{r}{4}$ more than the corresponding correspondence between $s_r(x)$ and $s_r(y)$, where $|C|$ denotes the length of the correspondence. Since there exists an optimal correspondence between $s_r(x)$ and $s_r(y)$ of length no more than $2n$, it follows that $\mathrm{dtw}(x, y) \leq \mathrm{dtw}(s_r(x), s_r(y)) + nr/2$, which implies the third part of the lemma.

A full proof of Lemma 4.6 appears in the extended paper [31]. Next we prove Theorem 4.2.

Proof of Theorem 4.2. Without loss of generality, the minimum non-zero distance in Σ is 1 and the largest distance is some value m, which is at most exponential in n.

We begin by defining the (r, n^ϵ)-*DTW gap* problem for $r \geq 1$, in which for two strings x and y a return value of 0 indicates that $\mathrm{dtw}(x, y) < nr$ and a return value of 1 indicates that $\mathrm{dtw}(x, y) \geq n^{1-\epsilon}r$. By Lemma 4.6, in order to solve the (r, n^ϵ)-DTW gap problem for x and y, it suffices to determine whether $\mathrm{dtw}(s_r(x), s_r(y)) \leq n^{1-\epsilon}r$. Moreover, because the minimum distance between distinct letters in $s_r(x)$ and $s_r(y)$ is at least $r/4$, this can be done in time $O(n^{2-\epsilon}\log n)$ using Lemma 4.5.[4]

In order to obtain an n^ϵ-approximation for $\mathrm{dtw}(x, y)$, we begin by using Lemma 4.5 to either determine $\mathrm{dtw}(x, y)$ or to determine that $\mathrm{dtw}(x, y) \geq n^{1-\epsilon}$. For the rest of the proof, suppose we are in the latter case, meaning that we know $\mathrm{dtw}(x, y) \geq n^{1-\epsilon}$.

We will now consider the $(2^i, n^\epsilon/2)$-DTW gap problem for $i \in \{0, 1, 2, \ldots, \lceil \log m \rceil\}$. (Recall that m is the largest distance in Σ.) If the $(2^0, n^\epsilon/2)$-DTW gap problem returned 0, then we would know that $\mathrm{dtw}(x, y) \leq n$, and thus we could return $n^{1-\epsilon}$ as an n^ϵ-approximation for $\mathrm{dtw}(x, y)$. Therefore, we need only consider the case where the $(2^0, n^\epsilon/2)$-DTW gap returns 1. Moreover we may assume without computing it that $(2^{\lceil \log m \rceil}, n^\epsilon/2)$-DTW gap returns 0 since trivially $\mathrm{dtw}(x, y)$ cannot exceed nm. Because $(2^i, n^\epsilon/2)$-DTW gap returns 1 for $i = 0$ and returns 0 for $i = \lceil \log m \rceil$, there must be some i such that $(2^{i-1}, n^\epsilon/2)$-DTW gap returns 1 and $(2^i, n^\epsilon/2)$-DTW gap returns 0. Moreover, we can find such an i by performing a binary search on i in the range $R = \{0, \ldots, \lceil \log m \rceil\}$. We begin by computing $(2^i, n^\epsilon/2)$-DTW gap for i in the middle of the range R. If the result is a one, then we can recurse on the second half of the range; otherwise we recurse on the first half of the range. Continuing like this, we can find in time $\tilde{O}(n^{2-\epsilon}\log\log m) = \tilde{O}(n^{2-\epsilon})$ some value i for which $(2^{i-1}, n^\epsilon/2)$-DTW gap returns 1 and $(2^i, n^\epsilon/2)$-DTW gap returns 0. Given such an i, we know that $\mathrm{dtw}(x, y) \geq \frac{2^{i-1}n}{n^\epsilon/2} = 2^i n^{1-\epsilon}$ and that $\mathrm{dtw}(x, y) \leq 2^i n$. Thus we can return $2^i n^{1-\epsilon}$ as an n^ϵ approximation of $\mathrm{dtw}(x, y)$. ◀

5 Reducing Edit Distance to DTW and LCS

In this section we present a simple reduction from edit distance over an arbitrary metric to DTW over the same metric.At the end of the section, we prove as a corollary a conditional lower bound for DTW over three-letter Hamming space, prohibiting any algorithm from running in strongly subquadratic time.

Surprisingly, the *exact same reduction*, although with a different analysis, can be used to reduce the computation of edit distance (over generalized Hamming space) to the computation of longest-common-subsequence length (LCS). Since computing LCS is equivalent to computing edit distance without substitutions, this reduction can be interpreted as proving that edit distance without substitutions can be used to efficiently simulate edit distance with substitutions, also known as *simple edit distance*.

Recall that for a metric $\Sigma \cup \{\emptyset\}$, we define the edit distance between two strings $x, y \in \Sigma^n$ such that the cost of a substitution from a letter l_1 to l_2 is $d(l_1, l_2)$, and the cost of a deletion or insertion of a letter l is $d(l, \emptyset)$. Additionally, define the *simple edit distance* $\mathrm{ed}_S(x, y)$ to be the edit distance using only insertions and deletions.

For a string $x \in \Sigma^n$, define the *padded string* $p(x)$ of length $2n + 1$ to be the string $\emptyset x_1 \emptyset x_2 \emptyset x_3 \cdots x_n \emptyset$. In particular, for $i \leq 2n + 1$, $p(x)_i = \emptyset$ when i is odd, and $p(x)_i = x_{i/2}$ when i is even. The following theorem proves that $\mathrm{dtw}(p(x), p(y)) = \mathrm{ed}(x, y)$.

[4] The logarithmic factor comes from the fact that evaluating distances between points may take logarithmic time in our well-separated tree metric.

▶ **Theorem 5.1.** *Let* $\Sigma \cup \{\emptyset\}$ *be a metric. Then for any* $x, y \in \Sigma^n$, $\operatorname{dtw}(p(x), p(y)) = \operatorname{ed}(x, y)$.

Proof sketch. A key observation is that when constructing an optimal correspondence between $p(x)$ and $p(y)$, one may w.l.o.g. extend only runs consisting of $\emptyset$ characters. In particular, suppose that one extends a non-$\emptyset$ character a in $p(x)$ to match a non-$\emptyset$ character b in $p(y)$. Then the extended run of a's must not only overlap b, but also the $\emptyset$-character preceding b. The total cost of extending a to overlap b is therefore $d(a, \emptyset) + d(a, b)$, which by the triangle inequality is at least $d(\emptyset, b)$. Thus instead of extending the run containing a, one could have instead extend a run of $\emptyset$-characters to overlap b at the same cost.

The fact that optimal correspondences arise by simply extending runs of $\emptyset$-characters can then be used to prove Theorem 5.1; in particular, given such a correspondence, one can obtain a sequence of edits from x to y by performing a substitution every time the correspondence matches two non-$\emptyset$ characters and a insertion or deletion every time the correspondence matches a non-$\emptyset$ character and a $\emptyset$-character. ◀

Theorem 5.2 proves an analogous reduction from edit distance to LCS.

▶ **Theorem 5.2.** *Let* Σ *be a generalized Hamming metric. Then for any* $x, y \in \Sigma^n$, $\operatorname{ed}_S(p(x), p(y)) = 2 \operatorname{ed}(x, y)$.

Proof sketch. Each edit in x can be simulated in $s(x)$ using exactly two insertions/deletions. In particular, the substitution of a character in x corresponds with the deletion and insertion of the same character in $s(x)$; and the insertion/deletion of a character in x corresponds with the insertion/deletion of that character and an additional $\emptyset$-character in $s(x)$.

This establishes that $\operatorname{ed}_S(p(x), p(y)) \leq 2 \operatorname{ed}(x, y)$. The other direction of inequality is somewhat more subtle, and is differed to the extended paper [31]. ◀

Whereas Theorem 5.2 embeds edit distance into simple edit distance with no distortion, Theorem 5.3 shows that no nontrivial embedding in the other direction exists.

▶ **Theorem 5.3.** *Consider edit distance over generalized Hamming space. Any embedding from edit distance to simple edit distance must have distortion at least* 2*.*

Combining Theorem 5.1 with known conditional lower bounds for computing edit distance [13] yields a new conditional lower bound for computing DTW over a three-letter alphabet (in which character distances are zero or one). This concludes a direction of work initiated by Abboud, Backurs, and Williams [3], who proved the same result over five-letter alphabet.

▶ **Corollary 5.4.** *Let* $\Sigma = \{a, b, c\}$ *with distance function* $d(a, b) = d(a, c) = d(b, c) = 1$*. If we assume the Strong Exponential Time Hypothesis, then for all* $\epsilon > 1$*, no algorithm can compute* $\operatorname{dtw}(x, y)$ *for* $x, y \in \Sigma^n$ *in time less than* $O(n^{2-\epsilon})$*.*

The full proofs of Theorems 5.1, 5.2, and 5.3, as well as Corollary 5.4 are differed to the extended paper [31].

6 Approximating Edit Distance Over an Arbitrary Metric

In this section we present an approximation algorithm for edit distance over an arbitrary metric space. Our algorithm achieves approximation ratio at most n^ϵ (with high probability) and runtime $\tilde{O}(n^{2-\epsilon})$. Note that when the metric is a well-separated tree metric, such an algorithm can be obtained by combining the approximation algorithm for DTW from Section

4 with the reduction in Section 5. Indeed the algorithm in this section is structurally quite similar to the one in Section 4, but uses a probability argument exploiting properties of edit distance in order to hold over an arbitrary metric.

▶ **Theorem 6.1.** *Let $(\Sigma \cup \{\emptyset\}, d)$ be an arbitrary metric space such that $|l| \geq 1$ for all $l \in \Sigma$. For all $0 < \epsilon < 1$, and for strings $x, y \in \Sigma^n$, there is an algorithm which computes an $O(n^\epsilon)$-approximation for $\mathrm{ed}(x, y)$ (with high probability) in time $\tilde{O}(n^{2-\epsilon})$.*

Using the standard dynamic-programming algorithm for computing $\mathrm{ed}(x, y)$ [42, 37, 41], one can easily obtain the following observation, analogous to Lemma 4.5 in Section 4:

▶ Observation 6.2. Consider $x, y \in \Sigma^n$, and let R be the smallest magnitude of the letters in x and y. There is an $O(n^{2-\epsilon})$-time algorithm which returns a value at least as large as $\mathrm{ed}(x, y)$; and which returns exactly $\mathrm{ed}(x, y)$ when $\mathrm{ed}(x, y) \leq R \cdot n^{1-\epsilon}$.

In order to prove Theorem 6.1, we present a new definition of the r-simplification of a string. The difference between this definition and the one in the preceding section allows the new definition to be useful when studying edit distance rather than dynamic time warping.

▶ **Definition 6.3.** *For a string $x \in \Sigma^n$ and for $r \geq 1$, we construct the* r-simplification $s_r(x)$ *by removing any letter l satisfying $|l| \leq r$.*

In the proof of Theorem 6.1 we use randomization in the selection of r in order to ensure that $s_r(x)$ satisfies desirable properties in expectation. The key proposition follows:

▶ **Proposition 6.4.** *Consider strings x and y in Σ^n. Consider $0 < \epsilon < 1$ and $R \geq 1$. Select r to be a random real between R and $2R$. Then the following three properties hold:*

- *Every letter l in $s_r(x)$ or $s_r(y)$ satisfies $|l| \geq R$.*
- *If $\mathrm{ed}(x, y) \leq \frac{nR}{15n^\epsilon}$ then $\mathbb{E}[\mathrm{ed}(s_r(x), s_r(y))] \leq \frac{nR}{3n^\epsilon}$.*
- *If $\mathrm{ed}(x, y) > 5nR$, then $\mathrm{ed}(s_r(x), s_r(y)) > nR$.*

The full proofs of Proposition 6.4 and of Theorem 6.1 appear in the extended paper [31]. Structurally, both proofs are similar to the analogous results in Section 4. The key difference appears in the proof of the second part of Proposition 6.4, which uses the random selection of r in order to probabilistically upper-bound $\mathrm{ed}(s_r(x), s_r(y)$. This is presented below.

▶ **Lemma 6.5.** *Consider strings x and y in Σ^n. Consider $R \geq 1$ and select r to be a random real between R and $2R$. Then $\mathbb{E}[\mathrm{ed}(s_r(x), s_r(y))] \leq 5\,\mathrm{ed}(x, y)$.*

Proof. Consider an optimal sequence S of edits from x to y. We will consider the cost of simulating this sequence of edits to transform $s_r(x)$ to $s_r(y)$. Insertions and deletions are easily simulated by either performing the same operation to $s_r(x)$ or performing no operation at all (if the operation involves a letter of magnitude less than or equal to r). Substitutions are slightly more complicated as they may originally be between letters $l_1 \in x$ and $l_2 \in y$ of different magnitudes. By symmetry, we may assume without loss of generality that $|l_1| < |l_2|$. We will show that the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most $5d(l_1, l_2)$. Because insertions and deletions can be simulated with no overhead, it follows that $\mathbb{E}[\mathrm{ed}(s_r(x), s_r(y))] \leq 5\,\mathrm{ed}(x, y)$.

If $|l_1| \leq r < |l_2|$ then l_1 does not appear in $s_r(x)$ but l_2 remains in $s_r(y)$. Thus what was previously a substitution of l_1 with l_2 becomes an insertion of l_2 at cost $|l_2|$. On the other hand, if we do not have $|l_1| \leq r < |l_2|$, then either both l_1 and l_2 are removed from $s_r(x)$ and $s_r(y)$ respectively, in which the substitution operation no longer needs to be performed, or both l_1 and l_2 are still present, in which case the substitution operation can still be performed at cost $d(l_1, l_2)$. Therefore, the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most

$$\Pr[|l_1| \leq r < |l_2|] \cdot |l_2| + d(l_1, l_2). \tag{1}$$

Because r is selected at random from the range $[R, 2R]$, the probability that $|l_1| \le r < |l_2|$ is at most $\frac{|l_2|-|l_1|}{R}$. By the triangle inequality, this is at most $\frac{d(l_1,l_2)}{R}$. If we suppose that $|l_2| \le 4R$, then it follows by (1) that the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most $\frac{d(l_1,l_2)}{R} \cdot 4R + d(l_1, l_2) \le 5d(l_1, l_2)$.

If, on the other hand, $|l_2| > 4R$, then in order for $|l_1| \le r$ to be true, we must have $|l_1| \le 2R$, meaning by the triangle inequality that $d(l_1, l_2) \ge |l_2|/2$. Thus in this case $|l_2| \le 2d(l_1, l_2)$, meaning by (1) that the expected cost of simulating the substitution of l_1 to l_2 in $s_r(x)$ is at most three times as expensive as the original substitution. ◀

References

1 John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *arXiv preprint*, 2015. `arXiv:1501.07053`.

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

4 Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating Dynamic Time Warping and Edit Distance for a Pair of Point Sequences. In *32nd International Symposium on Computational Geometry (SoCG)*, pages 6:1–6:16, 2016.

5 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 377–386, 2010.

6 Alexandr Andoni and Krzysztof Onak. Approximating Edit Distance in Near-Linear Time. *SIAM J. Comput.*, 41(6):1635–1648, 2012.

7 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the 47th Annual Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.

8 Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 267–276, 2011.

9 Ziv Bar-Yossef, T.S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Proceedings of 45th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2004.

10 Tugkan Batu, Funda Ergün, and Süleyman Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the 17th Annual Symposium on Discrete Algorithms (SODA)*, pages 792–801, 2006.

11 Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn J. Keogh. Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 49–58, 2015.

12 Vladimir Braverman, Moses Charikar, William Kuszmaul, David Woodruff, and Lin Yang. The One-Way Communication Complexity of Dynamic Time Warping Distance. Manuscript submitted for publication, 2018.

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

14 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2015.

15 EG Caiani, A Porta, G Baselli, M Turiel, S Muzzupappa, F Pieruzzi, C Crema, A Malliani, and S Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the

cardiac filling phases on left ventricular volume. In *Computers in Cardiology 1998*, pages 73–76, 1998.

16 Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Kouckỳ, and Michael Saks. Approximating Edit Distance Within Constant Factor in Truly Sub-Quadratic Time. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 979–990, 2018.

17 Diptarka Chakraborty, Elazar Goldenberg, and Michal Kouckỳ. Streaming algorithms for computing edit distance without exploiting suffix trees. *arXiv preprint*, 2016. `arXiv:1607.03718`.

18 Moses Charikar, Ofir Geri, Michael P. Kim, and William Kuszmaul. On Estimating Edit Distance: Alignment, Dimension Reduction, and Embeddings. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 34:1–34:14, 2018. `doi:10.4230/LIPIcs.ICALP.2018.34`.

19 Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996, 2012.

20 Anne Driemel and Francesco Silvestri. Locality-Sensitive Hashing of Curves. In *33rd International Symposium on Computational Geometry (SoCG)*, pages 37:1–37:16, 2017.

21 Ioannis Z. Emiris and Ioannis Psarros. Products of Euclidean Metrics and Applications to Proximity Questions among Curves. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 37:1–37:13, 2018.

22 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

23 Zvi Galil and Kunsoo Park. An improved algorithm for approximate string matching. *SIAM Journal on Computing*, 19(6):989–999, 1990.

24 Toni Giorgino et al. Computing and visualizing dynamic time warping alignments in R: the DTW package. *Journal of statistical Software*, 31(7):1–24, 2009.

25 Omer Gold and Micha Sharir. Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier. In *44th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 25:1–25:14, 2017.

26 Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of computational biology*, 9(2):371–388, 2002.

27 Eamonn J. Keogh. Exact Indexing of Dynamic Time Warping. In *28th International Conference on Very Large Data Bases (VLDB)*, pages 406–417, 2002.

28 Eamonn J. Keogh and Michael J. Pazzani. Scaling up Dynamic Time Warping to Massive Dataset. In *Principles of Data Mining and Knowledge Discovery, Third European Conference, (PKDD)*, pages 1–11, 1999.

29 Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289, 2000.

30 Jon Kleinberg and Eva Tardos. *Algorithm design.* Pearson, 2006.

31 William Kuszmaul. Dynamic Time Warping in Strongly Subquadratic time: Algorithms for the Low-Distance Regime and Approximate Evaluation. *arXiv preprint*, 2019. `arXiv:1904.09690`.

32 William Kuszmaul. Efficiently Approximating Edit Distance Between Pseudorandom Strings. In *Proceedings of the thirtieth annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019.

33 Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998.

34 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`.

35 Mario E Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of 7th International Conference on Computer Vision*, volume 1, pages 108–115, 1999.

36 Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

37 Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

38 François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn J. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowl. Inf. Syst.*, 47(1):1–26, 2016.

39 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.

40 Esko Ukkonen. Algorithms for approximate string matching. *Information and control*, 64(1-3):100–118, 1985.

41 Taras K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, 1968.

42 Robert A. Wagner and Michael J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, 1974.

43 Rex Ying, Jiangwei Pan, Kyle Fox, and Pankaj K Agarwal. A simple efficient approximation algorithm for dynamic time warping. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 21, 2016.

44 Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 181–192, 2003.