Recovery from Non-Decomposable Distance Oracles

Zhuangfei Hu 🖂 University of Waterloo, Canada

Xinda Li ⊠ University of Waterloo, Canada

David P. Woodruff \square Carnegie Mellon University, Pittsburgh, PA, USA

Hongyang Zhang ⊠ University of Waterloo, Canada

Shufan Zhang ⊠ University of Waterloo, Canada

- Abstract

A line of work has looked at the problem of recovering an input from distance queries. In this setting, there is an unknown sequence $s \in \{0,1\}^{\leq n}$, and one chooses a set of queries $y \in \{0,1\}^{\mathcal{O}(n)}$ and receives d(s, y) for a distance function d. The goal is to make as few queries as possible to recover s. Although this problem is well-studied for decomposable distances, i.e., distances of the form $d(s, y) = \sum_{i=1}^{n} f(s_i, y_i)$ for some function f, which includes the important cases of Hamming distance, ℓ_p -norms, and M-estimators, to the best of our knowledge this problem has not been studied for non-decomposable distances, for which there are important special cases such as edit distance, dynamic time warping (DTW), Fréchet distance, earth mover's distance, and so on. We initiate the study and develop a general framework for such distances. Interestingly, for some distances such as DTW or Fréchet, exact recovery of the sequence s is provably impossible, and so we show by allowing the characters in y to be drawn from a slightly larger alphabet this then becomes possible. In a number of cases we obtain optimal or near-optimal query complexity. We also study the role of adaptivity for a number of different distance functions. One motivation for understanding non-adaptivity is that the query sequence can be fixed and the distances of the input to the queries provide a non-linear embedding of the input, which can be used in downstream applications involving, e.g., neural networks for natural language processing.

2012 ACM Subject Classification Theory of computation \rightarrow Lower bounds and information complexity: Theory of computation \rightarrow Parameterized complexity and exact algorithms: Theory of computation \rightarrow Algorithm design techniques

Keywords and phrases Sequence Recovery, Edit Distance, DTW Distance, Fréchet Distance

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.73

Related Version Full Version: https://arxiv.org/abs/2209.05676 [23]

Funding David P. Woodruff: David P. Woodruff would like to thank support from ONR grant N00014-18-1-2562 and a Simons Investigator Award.

Hongyang Zhang: Hongyang Zhang would like to thank support from NSERC Discovery Grant RGPIN-2022-03215, DGECR-2022-00357.

1 Introduction

We study the problem of exact recovery of a sequence from queries to a distance oracle. Suppose there is an unknown input sequence s with length at most n, defined on a binary alphabet $\{0,1\}$. Assume we have a distance oracle which returns the distance d(s,q) between



© Duangfei Hu, Xinda Li, David P. Woodruff, Hongyang Zhang, and Shufan Zhang;

licensed under Creative Commons License CC-BY 4.0 14th Innovations in Theoretical Computer Science Conference (ITCS 2023).

Editor: Yael Tauman Kalai; Article No. 73; pp. 73:1–73:22

Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

73:2 Recovery from Non-Decomposable Distance Oracles

a query sequence q and the unknown sequence s, where the query sequence q is chosen either adaptively or non-adaptively. The problem is to determine the sequence s with a minimal number of queries to the distance oracle. To the best of our knowledge, this problem has not been studied for *non-decomposable* distances, where the distance function cannot be decomposed into a sum of functions of each entry. Among all non-decomposable distances, we are particularly interested in the edit distance, (p)-Dynamic Time Warping (p-DTW), and Fréchet distances. The edit distance measures the minimum number of edit operations (i.e., insertions, deletions, and substitutions) for transforming one sequence to another. The p-DTW distance $(1 \le p < \infty)$ between two sequences x, y is defined to be the minimum ℓ_p distance between two equal-length expansions of x, y, where the expansion of a sequence means you can duplicate each character of each sequence an arbitrary number of times. When p = 1, the p-DTW distance is called the DTW distance. If we consider the ℓ_{∞} norm instead of the ℓ_p norm, we obtain the Fréchet distance.

The problem of exact recovery for *decomposable distances* is well-studied in the literature, and is related to the coin-weighing problem [34, 10] and the group testing problems [21, 4, 19]. The coin-weighing problem is to identify the weight of each coin from a collection of ncoins, each being of weight either w_0 or w_1 (w_0 and w_1 are distinct). In this problem, our only access to the coins is via weighing a subset of the coins on a spring scale. The group testing problem has also been shown to be equivalent to the coin-weighing problem in some settings [37]. This line of research has been extensively studied with interesting applications. For example, the coin-weighing problem can be used in the detection problem [35], in the problem of determining a collection [13], and in the distinguishing family problem [31].

The query complexity of the adaptive version of the problem is also related to the original Mastermind game [26]. The Mastermind problem can be phrased as guessing an input sequence based on Hamming distance queries. The non-adaptive version of the problem can be shown to be equivalent to the well-studied coin-weighing problem [10]. One can then consider other variants of the Mastermind game where the input sequence is guessed based on other distance metrics, such as permutation-based distances [2], ℓ_p distances [22] and graph distances [32, 24]. However, general distance metrics that do not decompose into coordinate sums are less understood. In this paper, we initiate the study of this exact recovery problem on *non-decomposable* distances.

One motivation of our exact recovery problem is its application to adversarially robust learning on discrete domains. It is well-known that deep neural networks are vulnerable to adversarial examples: test inputs that have been modified slightly in the ℓ_p space can lead to problematic machine predictions. Though there exist various techniques such as Pixel-DP [29] and randomized smoothing [18] that achieve certified robustness against ℓ_p -norm perturbations in continuous domains, in many tasks such as natural language processing, the ℓ_p norm is not well-defined for discrete perturbations. To resolve this issue, inputs from a discrete domain are usually mapped to vectors in the ℓ_p space before being passed to a classifier; this is also known as a word embedding. We require two properties of such a mapping: 1) zero information loss; 2) Lipschitzness with respect to the distance metric in the input space. We show that the exact recovery problem yields a direct construction of such mappings: suppose the set of query sequences is $\{q_1, \ldots, q_m\}$ and s is the unknown input sequence; the mapping for s: $\phi(s) = [d(s, q_1), \dots, d(s, q_m)]$ has Lipschitz constant at most \sqrt{m} (in the ℓ_2 norm) and maintains complete information about s. Similar to edit distance, which can be used for describing the adversarial capability in changing sequences, the DTW and Fréchet distances have received significant attention for their flexibility in handling temporal sequences. The special instance of our problem on DTW and Fréchet distances may be useful for analyzing the robustness of DTW neural networks [12].

A distance embedding further inspires theoretical applications in functional analysis [36]. While the space of input sequences s is a metric space, it may not be a Hilbert space with a definition of norm and inner product. Our result provides us with a tool to define a mapping from a metric space to a Hilbert space without loss of information about the input sequences. One can then use the norm or inner product to analyze input sequences, e.g., when two input sequences are orthogonal and how to normalize an input sequence to have norm 1.

1.1 Our Results

Assumptions. Throughout the paper, we assume the alphabet of the unknown input sequence s is $\{0,1\}$. We note that under this assumption, our results for DTW described below will apply to p-DTW. To recover the sequence s, we submit adaptive or non-adaptive query sequences to a distance oracle. As we will show in Section 1.1.1, for some distance metrics, there exist input sequences that any sequence on a binary alphabet cannot distinguish. Therefore, our query sequences may be allowed to utilize alphabets outside $\{0,1\}$ with $\mathcal{O}(1)$ extra characters to exactly recover the input sequence. We assume the maximum length of s is n, while the exact length of s is unknown.

Extension to non-binary alphabets. Our results are presented for the binary alphabet $\{0, 1\}$. We note that these results can be extended to any non-binary alphabet Σ by encoding the non-binary alphabet in a binary domain. This will increase the query complexity by a factor of $|\Sigma|$ if using a one-hot encoding, or a factor of $\log(|\Sigma|)$ if using a binary encoding, for example. This extension works for the results for all distance metrics shown in this paper.

We begin with a general coordinate descent framework that can help recover sequences from a large class of distance oracles, including but not limited to earth mover's distance (EMD), cascaded norms (ℓ_p of ℓ_q), and A norms (a.k.a. Mahalanobis distance). We then present improved results on three specific distance metrics: edit distance, DTW distance, and Fréchet distance. We first provide several observations on the sequence recovery problem, showing the existence of indistinguishable input sequences despite the fact that we can query their DTW and Fréchet distances with all possible binary query sequences. We also prove lower bounds on the query complexity in our distance recovery problem with respect to DTW, edit, and Fréchet distances. Then we present our main results on recovering sequences from edit, DTW, and Fréchet distance oracles, with adaptive and non-adaptive strategies.

1.1.1 Existence of Indistinguishable Sequences

We observe that, for some distances, there exist sequences that cannot be distinguished by any query sequence over a binary alphabet. This can be proved by showing concrete examples, i.e., a pair of sequences that cannot be distinguished, which we show is true for the DTW and the Fréchet distances, as stated in the following theorem.

▶ **Theorem 1** (Informal, existence of indistinguishable sequences). There exists a pair of sequences (s, s') such that s and s' cannot be distinguished by any query sequence on a binary alphabet, for the DTW distance and the Fréchet distance.

The formal proof of this theorem for the DTW distance is deferred to Theorem 24. The analogous discussion for the Fréchet distance can be found in Section 6. Due to the existence of indistinguishable sequences, we define the concept of an *equivalence class* of sequences, which is a set of input sequences which are indistinguishable from all queries by a given distance oracle.

73:4 Recovery from Non-Decomposable Distance Oracles

This observation suggests the scope of the distance recovery problem we study. We further *categorize the recovery guarantee into the following three levels*, from strong to weak: 1) recover the **exact input sequence**; 2) recover any sequence in the **same equivalence class of the input sequence**, where the equivalence class is defined to be a set for which any two input sequences in the equivalence class cannot be distinguished by calling the distance oracle on all query sequences; 3) recover any sequence which has **zero distance to the input sequence**. While the third level is the weakest one, in certain cases it can be reduced to the first two levels – for norm-induced distance functions, the recovered sequence is exactly the input sequence; for semi-norm-induced distance functions, the recovered sequence is in the same equivalence class. We will show that our general coordinate descent framework can recover sequences with the third-level guarantee.

1.1.2 General *Coordinate Descent* Framework for Adaptively Querying Distance Oracles

We develop a general framework for recovering an input sequence from adaptive queries, which models the problem as a zero-th order optimization problem and utilizes a coordinate-descentbased algorithm to give a solution. The coordinate descent framework defines the distance between the input sequence and the query sequence as the loss function. The objective of the optimization is to reduce the loss function to 0, which guarantees what we call the third level of recovery. We define a step operation to modify the query sequence. For example, in the context of edit distance, a step operation is defined as adding/removing/substituting a character of the query sequence. To perform coordinate descent, our algorithm performs one step operation each time and queries the oracle to find a direction for which the loss decreases by 1. By iteratively performing this method, the loss can be reduced to 0 and we show that the overall complexity of this method is poly(n), given that the maximum length of the sequence is n. For a large class of non-decomposable distance functions, such as the earth mover's distance (EMD), the cascaded norm $(\ell_p \text{ of } \ell_q)$, and the A norm, we can use this framework to yield a solution, as stated in the following theorem.

▶ Theorem 2 (Informal, Coordinate Descent for Adaptive Distance Queries). For an arbitrary distance oracle, a binary alphabet $\{0, 1\}$ and any input sequence $s \in \{0, 1\}^i$ where $0 \le i \le n$, using coordinate descent can reduce the distance to the input sequence s to 0, by adaptively querying the distance oracle between s and a set of query sequences with query complexity at most poly(n).

Sufficient conditions for using this framework and details can be found in Theorem 15.

1.1.3 Lower Bounds on the Recovery Problem

If we study the problem of exact recovery (the first level of recovery), we can obtain an information-theoretic lower bound of $\tilde{\Omega}(n)$ for various distance oracles, given by the following theorem. Here $f(n) = \tilde{\Omega}(g(n))$ if $f(n) = \Omega(g(n)/\text{polylog}(n))$.

▶ Theorem 3 (Informal, Lower Bounds for Exact Recovery). For any input sequence $s \in \{0, 1\}^i$ where $0 \le i \le n$, if for any input sequence and query the distance oracle has poly(n) possible output values, any algorithm which exactly recovers s by querying the distance oracle between s and a set of query sequences requires query complexity at least $\tilde{\Omega}(n)$.

The idea behind this bound is that, there are exponentially many possible input sequences with length at most n, while the output of each query is a distance between two sequences which only has poly(n) possibilities. Hence, we need at least $\tilde{\Omega}(n) = \log_{poly(n)}(2^{n+1})$ queries.

We also show the same lower bound holds for any adaptive randomized query algorithm, by a reduction to a one-way communication game called *INDEX* [28]. We instantiate this theorem for the edit distance and DTW distance in Theorem 19 and Theorem 32.

We note for the DTW distance and Fréchet distance, there exist indistinguishable sequences which lead to the recovery problem for an equivalence class. Since the total number of equivalence classes is smaller than the number of input sequences, the previous lower bound no longer holds. So we need a different argument, as we give in the following theorem:

▶ Theorem 4 (Informal, Lower Bounds for Equivalence Class Recovery). For a binary alphabet $\{0,1\}$ and any input sequence $s \in \{0,1\}^i$ where $0 \le i \le n$, any algorithm which recovers the sequence in the same equivalence class as s, by querying the DTW or Fréchet distance oracle between s and a set of query sequences, requires query complexity at least $\Omega(n)$.

The formal proof is given in Theorem 31 and Theorem 40.

1.1.4 Adaptively Querying Distance Oracles, Optimally

We first answer the distance recovery problem with adaptive query strategies. Our solutions are summarized in the theorem below.

▶ **Theorem 5** (Informal, Upper Bounds for Adaptive Exact Recovery). For a binary alphabet $\{0,1\}$ and any input sequence $s \in \{0,1\}^i$ where $0 \le i \le n$, there exists an algorithm which can exactly recover the input sequence s, by adaptively querying the distance oracle (for the edit and DTW distances) between s and a set of query sequences with query complexity at most $\mathcal{O}(n)$.

All results in Theorem 5 match our lower bounds on the query complexity. Without extra character(s), using the DTW distance oracle we can only recover a sequence in the same equivalence class. Our result in Theorem 5 for the DTW distance is achieved with the assistance of 1 extra character outside the alphabet $\{0, 1\}$, and the proof and algorithm can be found in Theorem 18.

For the edit distance, we have two different adaptive algorithms that can achieve the $\mathcal{O}(n)$ bound. The first algorithm makes use of the property that, for two sequences, the edit distance is equal to the difference in their lengths, if and only if one sequence is a subsequence of the other. We construct an $\mathcal{O}(n)$ adaptive query set and a binary search algorithm utilizing this property to recover the input sequence. Our second algorithm instead queries the length of the input sequence by an empty sequence and then finds a set of $\mathcal{O}(n)$ bases as the query set, from which we can reconstruct the input sequence. These are further detailed in Theorem 16.

For the Fréchet distance, adaptive and non-adaptive strategies are essentially the same, because we prove that 2n-1 queries are necessary and sufficient for recovering from a Fréchet distance oracle. However, we can only recover a sequence in the equivalence class in this setting. This result is described as a non-adaptive query strategy in Theorem 42.

1.1.5 Non-adaptively Querying Distance Oracles, Optimally

Next we describe our non-adaptive query strategies for the distance recovery problem. Theorem 6 shows upper bounds for exact sequence recovery, while Theorem 7 summarizes our results on the recovery problem of finding a sequence in the same equivalence class as the input sequence.

73:6 Recovery from Non-Decomposable Distance Oracles

Oracle	Query Complexity	LB	Adaptive?	#EC	Level of Recovery	Positions
Edit	$\mathcal{O}(n)$	$\tilde{\Omega}(n)$	Adaptive	0	Exact sequence	Theorem 16
Edit	n + 1	$\tilde{\Omega}(n)$	Non-adaptive	1	Exact sequence	Theorem 21
Edit	n^2	$\tilde{\Omega}(n)$	Non-adaptive	0	Exact sequence	Theorem 23
(<i>p</i> -)DTW	n + 1	$\tilde{\Omega}(n)$	Adaptive	1	Exact sequence	Theorem 18
(<i>p</i> -)DTW	n	$\Omega(n)$	Non-adaptive	0	Equivalent class	Theorem 27
(<i>p</i> -)DTW	$n^2 + n$	$\tilde{\Omega}(n)$	Non-adaptive	1	Exact sequence	Theorem 33
(p-)DTW	n+2	$\tilde{\Omega}(n)$	Non-adaptive	2^*	Exact sequence	Theorem 34
Fréchet	2n - 1	2n - 1	N/A^{\dagger}	0**	Equivalent class	Theorem 42
Any distance	poly(n)	-	Adaptive	0	Zero distance to input	Theorem 15

Table 1 Summary of our results for recovering arbitrary input sequences of length n under the constraint that the query length is $\mathcal{O}(n)$. LB: Lower Bound. #EC: Number of Extra Characters.

[†] For both adaptively and non-adaptively querying the Fréchet distance oracle, the optimal bound on the query complexity is 2n - 1.

* Increasing #EC to a larger constant cannot improve the query complexity to be better than $\tilde{\mathcal{O}}(n)$.

^{**} Involving extra characters not only cannot improve the level of recovery from "equivalence class" to "exact sequence", but also cannot improve the query complexity (see Theorem 41).

▶ **Theorem 6** (Informal, Upper Bounds for Non-adaptive Exact Recovery). For a binary alphabet $\{0,1\}$ and any input sequence $s \in \{0,1\}^i$ where $0 \le i \le n$, there exists an algorithm which can exactly recover the input sequence s by querying the distance oracle (for the edit and DTW distances) between s and a non-adaptive set of query sequences with query complexity at most $\mathcal{O}(n)$, with the assistance of $\mathcal{O}(1)$ extra characters in the query sequences.

With 1 extra character, we show the construction of a set of non-adaptive queries that can exactly recover sequences from the edit distance (Theorem 21), while with 2 extra characters, we can exactly recover input sequences from the DTW distance (Theorem 34). Both results match our lower bound on the query complexity, while we complement our results with an $\mathcal{O}(n^2)$ query complexity algorithm for the DTW distance with 1 extra character (Theorem 33). We note that non-adaptive strategies have limited power compared to adaptive strategies. Hence, we need to use more extra characters to construct the query set to solve the problem. For the edit distance, introducing more than 1 extra character cannot encode more information in the query results, because the cost between 0 (or 1) and any other additional character is always the same.

▶ **Theorem 7** (Informal, Upper Bounds for Non-adaptive Equivalence Class Recovery). For a binary alphabet $\{0,1\}$ and any input sequence $s \in \{0,1\}^i$ where $0 \le i \le n$, there exists an algorithm which can recover the sequence in the same equivalence class as the input sequence s, by querying the distance oracle (for the DTW and Fréchet distances) between s and a non-adaptive set of query sequences with query complexity at most O(n), without extra characters in the query sequence.

By Theorem 7, if we are not allowed to use extra characters, we can only recover the sequence in the same equivalence class as the input sequence for the DTW distance. Our query construction and proof are shown in Theorem 27. We also remark that for Fréchet distance, using extra characters cannot help to improve the results of Theorem 42, as shown in Theorem 41.

Summary. The main technical results of this paper are summarized in Table 1.

2 Preliminaries

We briefly introduce the definitions and notations of the distance metrics we consider, namely, the edit distance, DTW distance, and Fréchet distance.

▶ Definition 8 (Edit Distance, Levenshtein Distance). Given two sequences x and y, the edit distance $d_L(x, y)$ equals the minimal number of edit operations required for a sequence x to be transformed to sequence y. Specifically, we consider the Levenshtein distance [30] which captures the addition, deletion, and substitution of single symbols.

Definition 9 (Runs and Expansion, [9]). The runs of a sequence x are the maximal substrings consisting of a single repeated character. Any sequence obtained from x by extending x's runs is an expansion of x. We denote the number of runs of a sequence x by RUNS(x).

Definition 10 (Condensed Expression). We say y is a condensed expression of x if (i) y has the same number of runs as x, (ii) each run of y only has 1 character.

▶ Definition 11 (Subsequence and Substring). Given a sequence y, its subsequence x is derived by deleting zero or more characters from y without changing the order of the remaining characters. The substring x' is a contiguous subsequence of y. We use x[a] to denote the a-th character of the sequence x, and x[a,b] to denote a substring of x which starts from the a-th character and ends at the b-th character.

▶ Definition 12 (DTW Distance, [9]). Consider two sequences x, y of length m_1 (and m_2 , respectively). A correspondence $(\overline{x}, \overline{y})$ between x and y is a pair of equal-length expansions of x and y. The cost of a correspondence is calculated as the ℓ_1 distance between $\overline{x}, \overline{y}$: $\|\overline{x} - \overline{y}\|_1$. A correspondence between x and y is said to be optimal if it has the minimum attainable cost, and the resulting cost is called the dynamic time warping distance $d_{DTW}(x, y)$, that is $d_{DTW}(x, y) = \min_{(\overline{x}, \overline{y}) \in \mathcal{W}_{x,y}} \|\overline{x} - \overline{y}\|_1$, where $\mathcal{W}_{x,y}$ denotes the set of all correspondences $(\overline{x}, \overline{y})$.

▶ **Definition 13** (*p*-DTW Distance, [11]). By replacing the l_1 norm in Definition 12 with the l_p norm $(1 \le p < \infty)$, we obtain the definition for the *p*-DTW distance.

▶ Definition 14 (Fréchet Distance). By replacing the l_1 norm in Definition 12 with the l_{∞} norm, we obtain the definition of the Fréchet distance.

Note. The Fréchet distance in our paper is equivalent to the discrete Fréchet distance in the prior works of [3, 7].

3 Recovery Using Adaptive Queries

3.1 General Framework of Coordinate Descent

▶ **Theorem 15** (Coordinate Descent). There exists an adaptive algorithm which returns a sequence s' such that its distance to the input sequence s satisfies dist(s', s) = 0 using poly(n) queries, given that the following two conditions are true:

∀q,q'(q' ≠ q), we can find q" in poly(n) queries such that dist(q,q') > dist(q,q");
 ∀q,q', dist(q,q') ≤ poly(n).

Proof sketch. The two above conditions naturally imply a local search algorithm. To recover the sequence q, we perform the following steps: 1) randomly initialize q'. 2) find q'' such that dist(q,q') > dist(q,q''). 3) set q' to q'' and repeat 2) to 3). The algorithm

73:8 Recovery from Non-Decomposable Distance Oracles

terminates if dist(q, q') = 0. Since we reduce dist(q, q') by least 1 in each iteration, and $dist(q, q') \leq poly(n)$, the algorithm terminates in at most poly(n) iterations. Therefore, the total number of queries is $\mathcal{O}(poly(n))$. Notice that the above local search algorithm can be applied to all aforementioned distances. Specifically, the complexity for the edit distance, DTW distance and Fréchet distance is $\mathcal{O}(n^2)$, $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively. A detailed instantiation of the algorithm on these distances can be found in Appendix A.

3.2 Edit Distance

We show that a binary input sequence with maximum length n can be adaptively recovered using at most $\mathcal{O}(n)$ queries to the edit distance oracle, by the following theorem.

▶ **Theorem 16** (Adaptive Strategy for Edit Distance). For a binary alphabet $\{0,1\}$, and any input sequence $s \in \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$, there exists an adaptive algorithm to recover the input sequence s using at most n + 2 queries **Q** of length $\leq n$ and the exact Levenshtein distance of s to each query sequence $q_i \in \mathbf{Q}$, where the query sequences use no extra characters.

Proof. The adaptive query strategy is the following. We first use an empty sequence to query the length $\ell \in [n] = \{1, 2, ..., n\}$ of the input sequence. Then we use $\ell + 1 \leq n + 1$ queries: an $e_0 = 0^{\ell}$ query and a set of $e_i = 0^{i-1} 10^{\ell-i}, i \in [\ell]$ queries (all with length ℓ).

 $\rhd \text{ Claim 17.} \quad s[i] = \begin{cases} 0, & \text{if } d_L(s, e_0) - d_L(s, e_i) \leq 0; \\ 1, & \text{if } d_L(s, e_0) - d_L(s, e_i) = 1. \end{cases}$

Proof of claim. If s[i] = 1, $d_L(s, e_0) - d_L(s, e_i) = (\#1$'s in s) - (#1's in s-1) = 1. If s[i] = 0, $d_L(s, e_0) = (\#1$'s in s). We show that $d_L(s, e_i) \ge (\#1$'s in s). First, $d_L(s, e_i) \ge (\#1$'s in s) - (#1's in $e_i) = \#1$'s in s - 1. Consider the series of transformations from s to e_i : 1) If we only perform substitutions on s, we need at least #1's in s + 1 operations. 2) Otherwise we show that we have at least one insertion. If we perform at least one insertion(s) on s, since s and e_i are of the same length, we would need at least one insertion(s) on s. Note that insertions on s cannot reduce the difference of the number of 1's between s and e_i . Thus, we need at least (#1's in s - 1) extra operations to reduce the difference to 0 and we have $d_L(s, e_i) \ge (\#1$'s in s - 1) +1 = #1's in s. Combining these cases, we obtain $d_L(s, e_i) \ge (\#1$'s in s).

By claim 17, we can recover the sequence s character by character.

•

▶ Remark. We remark that we have two meaningful strategies for edit distance, for details please refer to the full version of this paper.

3.3 DTW Distance

▶ **Theorem 18** (Adaptive Strategy for DTW Distance). For a binary alphabet $\{0,1\}$, and any input sequence $s \in \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$, there exists an adaptive algorithm to recover the input sequence s using at most n + 1 queries **Q** of length $\leq n$ and the exact DTW distance of s to each query sequence $q_i \in \mathbf{Q}$, where the query sequences use 1 extra character.

Proof. Using an adaptive method, for a binary alphabet $\{0, 1\}$, an input sequence $s \in \{0, 1\}^i$ where $0 \le i \le n$ can be exactly recovered with at most n + 1 queries to the DTW distance oracle. We need 1 additional character, which is the fractional character $\frac{1}{2}$, to construct the set of query sequences. The details are presented as follows. Here we use $q^{(i)}$ to denote the *i*-th query, s_j to denote the *j*-th character in *s* and $q_j^{(i)}$ to denote the *j*-th character in $q^{(i)}$.

First, with a single-character query sequence $q^{(1)} = \frac{1}{2}$, we can obtain the length of the input sequence s, which is $\ell = 2d_{DTW}(s, q^{(1)})$. We show how to recover the whole sequence by induction.

Base case: To recover s_1 , we consider the query $q^{(2)} = 0$ $(\frac{1}{2})^{\ell-1}$. Note that each $\frac{1}{2}$ in the $q^{(2)}$ corresponds to at least $\frac{1}{2}$ cost in the query result, and we have $d_{DTW}(s, q^{(2)}) \ge \frac{\ell-1}{2}$. If $s_1 = 0$, then s_1 and $q_1^{(2)}$ are perfectly matched, so $d_{DTW}(s, q^{(2)}) = \frac{\ell-1}{2}$. Otherwise, the first character 0 in $q^{(2)}$ would correspond to cost > 0 in the query result, so $d_{DTW}(s, q^{(2)}) > \frac{\ell-1}{2}$. **Induction step:** Suppose we have recovered s_1, s_2, \ldots, s_k . We show that we can recover s_{k+1} with the query sequence $q^{(k+2)} = s_1 s_2 \cdots s_k^2 (\frac{1}{2})^{\ell-k-1}$. Noting that each $\frac{1}{2}$ in $q^{(k+2)}$ corresponds to at least a $\frac{1}{2}$ cost in the query result, we have $d_{DTW}(s, q^{(k+2)}) \ge \frac{\ell-k-1}{2}$. If $s_{k+1} = s_k$, then $s_1, s_2, \ldots, s_k, s_{k+1}$ and $q_1^{(k+2)}, q_2^{(k+2)}, \ldots, q_k^{(k+2)}, q_{k+1}^{(k+2)}$ can be perfectly matched, so $d_{DTW}(s, q^{(k+2)}) = \frac{\ell-k-1}{2}$. Otherwise, we claim that $d_{DTW}(s, q^{(k+2)}) > \frac{\ell-k-1}{2}$. If the cost corresponding to $q_{k+1}^{(k+2)} > 0$, we would already have $d_{DTW}(s, q^{(k+2)}) > \frac{\ell-k-1}{2}$, so we can assume that the cost corresponding to $q_{k+1}^{(k+2)}$ is 0. Since $q_{k+1}^{(k+2)} = s_k \neq s_{k+1}$, we know that $q_{k+1}^{(k+2)}$ cannot be matched with s_{k+1} . Suppose $q_{k+1}^{(k+2)}$ is matched with substring $s_u, s_{u+1}, \ldots, s_{u+t}$ in the optimal DTW matching, where $t \ge 0$ and $s_u = s_{u+1} = \cdots = s_{u+t} = q_{k+1}^{(k+2)} = s_k$. Since $k + 1 \notin [u, u + t]$, we either have k + 1 > u + t or k + 1 < u. If k + 1 > u + t, since $\forall u + t < j \le \ell$ we have s_j matched to a $\frac{1}{2}$, the total cost would be at least $\frac{\ell-(u+1)}{2} > \frac{\ell-k-1}{2}$. Otherwise if k + 1 < u, note that s_1, s_2, \ldots, s_u are matched to $q_1^{(k+2)}, q_2^{(k+2)}, \ldots, q_{k+1}^{(k+2)}$ in the optimal DTW matching. Since $s_{k+1} \neq s_k$, the number of runs in $s_1, s_2, \ldots, s_{k+1}$ would be greater than the number of runs in $q_1^{(k+2)}, q_2^{(k+2)}, \ldots, q_{k+1}^{(k+2)}$

4 Recovery Using a Non-Adaptive Edit Distance Oracle

We begin with a lower bound for edit distance.

▶ **Theorem 19.** For a binary alphabet $\{0,1\}$, any algorithm to recover an arbitrary input sequence $s \in \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$ by querying the Levenshtein distance to a set of sequences of length $\mathcal{O}(n)$ requires a query complexity of $\Omega(\frac{n}{\log n})$.

Proof. We start with a lower bound for deterministic algorithms. For each query, the result is an integer $d = \mathcal{O}(n)$, yielding $\mathcal{O}(n)$ possibilities. For $0 \le k \le n$, the number of different sequences of length k is 2^k , and the total number of sequences of length no greater than n would be $\sum_{k=0}^{n} 2^k = 2^{n+1} - 1$. Thus, to distinguish all possible sequences, one would need at least $\log_{\mathcal{O}(n)}(2^{n+1}) = \Omega(\frac{n}{\log n})$ queries.

Next we give a proof if one is allowed to use a randomized algorithm. To show this, we introduce a one-way two-party communication game called *INDEX*.

▶ Definition 20 (INDEX Game [28]). Consider two players Alice and Bob. Alice and Bob have access to a common public coin and their computation can depend on this. Alice holds an n-bit string $x \in \{0, 1\}^n$ and is allowed to send a single message M to Bob (i.e., this is a one-way protocol). Bob has an index $i \in [n]$ and his goal is to learn x[i], i.e., $\Pr_r[\mathcal{O}ut(M) = x[i]] \geq \frac{2}{3}$).

73:10 Recovery from Non-Decomposable Distance Oracles

It is shown in [28] that the above problem requires $|M| = \Omega(n)$. To reduce our recovery problem from the INDEX game, let R be an adaptive randomized recovery algorithm which works as follows. First, Alice randomly selects a query q_1 based on the first part r_1 of the shared public coin, and computes $d(q_1, x)$. Alice then adaptively selects a set of queries q_i , where each q_i is chosen based on disjoint parts r_1, \ldots, r_i of the public coin, as well as the responses $d(q_1, x), \ldots, d(q_{i-1}, x)$ to previous queries. Alice then sends all query results $d(q_i, x)$ to Bob as the message M.

We now show that, if the algorithm R is correct w.p. 2/3, then M contains $\Omega(\frac{n}{\log n})$ query results. Given the success probability 2/3 of R, from message M, Bob can reconstruct the string x w.p. at least 2/3, so Bob can learn each bit of x w.p. at least 2/3. According to [28], $|M| = \Omega(n)$ bits. Since each distance query result contains at most $\mathcal{O}(\log n)$ bits, it follows that $\Omega(\frac{n}{\log n})$ queries are required.

4.1 Exact Recovery with Extra Character(s)

We now move on to the analysis of our upper bound for edit distance with the assistance of extra character(s).

▶ **Theorem 21.** For a binary alphabet $\{0,1\}$ and an input sequence $s \in \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$, there exists an algorithm to recover the input sequence s, given $\mathcal{O}(n)$ query sequences \mathbf{Q} of length $\leq n$ and the exact Levenshtein distance of s to each query sequence $q_i \in \mathbf{Q}$, where an extra character 2 is allowed in the query sequences.

Proof. To prove Theorem 21, the following Lemma 22 gives a useful result.

▶ Lemma 22. Consider a sequence defined on a binary alphabet $\{0, 1\}$ with maximum length n, which is defined as s = [prefix]0[suffix] where the length of sequence a is $||s|| = n' \le n$ and the length of the prefix is ||[prefix]|| = j. Consider another sequence $s' = 1^{j+1}2^{n-j-1}$, where 2 denotes a random character not in the binary alphabet $\{0, 1\}$. Let k denote the number of 1's in the prefix [prefix] of the sequence s. The claim is that the edit distance between s and s' is greater than or equal to n - k.

Due to space constraints, the proof of Lemma 22 is deferred to the full version. The construction of the query sequences is an extension of the query sequence we present in Lemma 22. We describe how this can be generalized to distinguish all the different input sequence pairs by constructing a set of n + 1 query sequences. We show that using this set of queries, any two possible input sequences $s \neq s'$ can be distinguished by n + 1 non-adaptive query results.

Query Sequence Construction. We assume the query sequences can include a character that is not in the alphabet. The n + 1 query sequences are constructed as follows. We use an empty sequence together with n sequences of the form $1^{j}2^{n-j}$ for j = 1, ..., n where 2 denotes a "not-in-the-alphabet" character. We will show that any two input sequences $s \neq s'$ are distinguished by this construction of query sequences.

First, it is clear that the empty sequence can distinguish two sequences with different length, i.e., $|s| \neq |s'|$. Next we need to show that, for two input sequences $s \neq s'$ with the same length $n' \leq n$, the remaining query sequences can distinguish them. Without loss of generality, we suppose s and s' have the same prefix with length j and first disagree on the $(j+1)^{th}$ character. That is, $s = [\text{prefix}]1[s_{\text{suffix}}]$ and $s' = [\text{prefix}]0[s'_{\text{suffix}}]$ where s_{suffix} and s'_{suffix} are not necessarily equal. Consider a query sequence $q_i = 1^{j+1}2^{n-j-1}$. We use the

results from Lemma 22 and obtain $dist(s', q_i) \ge n - k$ where k denotes the number of 1's in the prefix of sequence s'. Since we also know that $d(s, q_i) \le n - k - 1$, this query sequence q_i can distinguish s and s'. This finishes the proof.

▶ Remark. The proof for Theorem 21 implicitly implies an exact recovery algorithm that takes the edit distance query results as input and outputs the target sequence. We defer the algorithm to the full version of the paper.

4.2 Exact Recovery without Extra Characters

▶ **Theorem 23.** For a binary alphabet $\{0,1\}$ and an input sequence $s \in \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$, there exists an algorithm to recover the input sequence s, given $\mathcal{O}(n^2)$ query sequences \mathbf{Q} of length $\leq n$ and the exact Levenshtein distance of s to each query sequence $q_i \in \mathbf{Q}$, without extra characters.

Proof. The construction can be obtained by naturally extending the query set in the proof of Theorem 16 to all lengths $\ell \in [n]$. This gives us $\sum_{1}^{n} n = \mathcal{O}(n^2)$ queries.

5 Recovery Using a Non-Adaptive DTW Distance Oracle

5.1 Hardness Result without Extra Characters

▶ **Theorem 24.** There exists a pair of input sequences s and s' such that for any query sequence q, $d_{DTW}(s,q) = d_{DTW}(s',q)$. That is, s and s' cannot be distinguished by DTW Distance Oracle queries without extra characters.

Consider the following counterexample: s = 010110 and s' = 011010. We can argue that this pair of input sequences cannot be distinguished by any binary sequence query q. Due to space constraints, the proof can be found in the full version of the paper. This hardness result shows that sequences cannot be exactly recovered using DTW queries.

5.2 Recovery without Extra Characters w.r.t. Equivalence Classes

For ease of presentation, we say that any two different input sequences s and s' are distinguishable if s and s' can be distinguished by DTW Distance Oracle queries. We categorize mutually indistinguishable sequences into equivalence classes. In this context, using binary queries, the best solution we can provide in this setting is to recover those input sequences up to their equivalence class. First, we would like to characterize the set of indistinguishable binary sequences, given a parameterized sequence length n. The equivalence class is not so simple to describe, as can be seen from Observation 4 of [33]. However, we can propose an optimal query strategy in this setting to distinguish all distinguishable sequences and prove optimality by making use of the reduction between the calculation of DTW distance and the min 1-separated sum problem [1, 33]. We introduce the necessary results from [33] below and interpret them in our setting.

MSS Problem. The min 1-separated sum (MSS) problem takes as input a sequence seq of m positive integers and an integer $r \ge 0$. The problem is to select r integers from seq and minimize their sum, under the constraint that any two adjacent integers cannot be selected simultaneously. We say (seq, r) is an MSS instance.

73:12 Recovery from Non-Decomposable Distance Oracles

▶ **Theorem 25** (DTW-to-MSS Reduction, [33], Theorem 2). Let $x \in \{0,1\}^m$ and $y \in \{0,1\}^n$ be two binary strings such that x[1] = y[1], x[m] = y[n], and $|\tilde{x}| \ge |\tilde{y}|$. Then, the DTW distance between x and y, i.e., $d_{DTW}(x, y)$, equals the sum of a solution for $MSS((|x^{(2)}|, \ldots, |x^{(|\tilde{x}|-1)}|), (|\tilde{x}| - |\tilde{y}|)/2)$.

For ease of presentation, we will use $MSS(x, (|\tilde{x}| - |\tilde{y}|)/2)$ to represent the same MSS instance.

▶ **Theorem 26** ([33], Observation 4). Let $x \in \{0,1\}^m$, $y \in \{0,1\}^n$ with $m' := |\tilde{x}| \ge n' := |\tilde{y}|$. Further, let $a := |x^{(1)}|, a' := |x^{(m')}|, b := |y^{(1)}|$, and $b' := |y^{(n')}|$. The following holds: If $x[1] \ne y[1]$, then:

$$d_{DTW}(x,y) = \begin{cases} \max(a,b), & m' = n' = 1; \\ a + d_{DTW}(x[a+1,m],y), & m' > n' = 1; \\ \min(a + d_{DTW}(x[a+1,m],y), b + d_{DTW}(x,y[b+1,n])), & n' > 1. \end{cases}$$

If x[1] = y[1] and $x[m] \neq y[n]$, then:

$$d_{DTW}(x,y) = \begin{cases} a' + d_{DTW} \left(x \left[1, m - a' \right], y \right), & n' = 1; \\ \min \left(a' + d_{DTW} \left(x \left[1, m - a' \right], y \right), b' + d_{DTW} \left(x, y \left[1, n - b' \right] \right) \right), & n' > 1. \end{cases}$$

In Theorem 26, we call a, b, a' and b' (which are the length of first/last blocks of x or y) offsets. Theorem 26 actually states that, for two sequences x, y with different starting and ending characters, by removing the first/last run of x or y, calculating $d_{DTW}(x, y)$ can be reduced to calculating the offset and solving a DTW sub-problem where the sub-sequences start and end with the same character.

With these useful results at hand, now we prove the following results for recovering sequences using the DTW distance oracle with only binary queries.

▶ **Theorem 27.** There exists a set Q of O(n) queries, each of which has O(n) length, such that for any two different input sequences s and s', s and s' are distinguishable \iff s and s' can be distinguished by Q.

Proof. First (\Leftarrow), for any given query set Q and two different input sequences s and s', if s and s' can be distinguished by Q then s and s' are distinguishable. Then we need to prove the opposite side (\Rightarrow). To see this, we construct the following query set Q and prove the contrapositive: if s and s' cannot be distinguished by Q, then s and s' are not distinguishable.

Let
$$z_i = \begin{cases} 0^n, \ i = 1; \\ 0^n 1(01)^{m-1} 0^n, \ i = 2m+1; \\ 0^n (10)^{m-1} 1^n, \ i = 2m, \end{cases}$$
 and $o_i = \begin{cases} 1^n, \ i = 1; \\ 1^n 0(10)^{m-1} 1^n, \ i = 2m+1; \\ 1^n (01)^{m-1} 0^n, \ i = 2m, \end{cases}$

where $1 \leq i \leq n$ and *m* is a positive integer. It is clear that o_i 's and z_i 's are of $\mathcal{O}(n)$ length. Let $\mathcal{Q} = \{o_i | 1 \leq i \leq n\} \bigcup \{z_i | 1 \leq i \leq n\}$. We show that given any two different input sequences *s* and *s'*, if *s* and *s'* cannot be distinguished by \mathcal{Q} then *s* and *s'* are not distinguishable.

 \triangleright Claim 28. Given two different input sequences s and s', if the condensed expressions of s and s' are different, then s and s' can be distinguished by Q.

Suppose s and s' cannot be distinguished by Q. By Claim 28, we know that the condensed expression of s and s' are the same. Suppose the number of runs in s and s' is k, the length of the *i*-th run in s is α_i , and the length of the *i*-th run in s' is α'_i . We denote the length of a sequence by $len(\cdot)$.

 \triangleright Claim 29. If s and s' cannot be distinguished by \mathcal{Q} , then $k \geq 3$.

\triangleright Claim 30. If s and s' cannot be distinguished by \mathcal{Q} , then $\alpha_1 = \alpha'_1$ and $\alpha_k = \alpha'_k$.

We defer the proof of the claims 28, 29, and 30 to the full paper. We next show that s and s' cannot be distinguished by any binary query r. Suppose the number of runs in r is l, and the length of the *i*-th run in r is β_i . Given s and r, we can calculate $d_{DTW}(s,r)$ with Theorem 26 and Theorem 25. Note that in Theorem 26, we may remove the first/last blocks of s (and s') or r to reduce to the case of Theorem 25. By Claim 30 we have $\alpha_1 = \alpha'_1$ and $\alpha_k = \alpha'_k$, while β_1 and β_l are only related to r but not s and s'. Therefore, the offsets while reducing to the case of Theorem 25 are the same. To prove that $d_{DTW}(s,r) = d_{DTW}(s',r)$, we only need to prove that for each possible reduction, the corresponding reduced MSS instances have the same sum of solutions.

Suppose after applying Theorem 26, s and r are reduced to sub-sequences s^* and r^* (where s^* and r^* have the same beginning and ending characters), while s' and r are reduced to s'^* and r^* . Now we calculate $d_{DTW}(s^*, r^*)$ and $d_{DTW}(s'^*, r^*)$ according to Theorem 25. Suppose s^* and s'^* have k^* runs and r^* have l^* runs.

Case 1. If $k^* = l^*$, then $d_{DTW}(s^*, r^*) = d_{DTW}(s'^*, r^*) = 0$.

- **Case 2.** If $k^* < l^*$, by Theorem 25 the generated MSS instance only depends on r^* and k^* . Thus, $d_{DTW}(s^*, r^*) = d_{DTW}(s'^*, r^*)$.
- **Case 3.** If $k^* > l^*$, we have the MSS instances $MSS(s^*, \frac{k^*-l^*}{2})$ and $MSS(s'^*, \frac{k^*-l^*}{2})$. Note that, we can always find a query $q \in Q$ which has l^* runs and has the same starting and ending characters as s^* and s'^* . Consider $d_{DTW}(s,q)$ and $d_{DTW}(s',q)$. Note that the first and last runs of q are both of length n and removing them would yield cost at least n the only possible reduction would be $d_{DTW}(s^*,q)$ and $d_{DTW}(s'^*,q)$. Since q cannot distinguish s and s', we have $d_{DTW}(s,q) = d_{DTW}(s',q)$, so $d_{DTW}(s^*,q) = d_{DTW}(s'^*,q)$, implying that $MSS(q^*, \frac{k^*-l^*}{2})$ and $MSS(q'^*, \frac{k^*-l^*}{2})$ have the same sum of solutions. Therefore, $d_{DTW}(s^*, r^*) = d_{DTW}(s'^*, r^*)$.

Combining the 3 cases above, we always have $d_{DTW}(s^*, r^*) = d_{DTW}(s'^*, r^*)$, so $d_{DTW}(s, r) = d_{DTW}(s', r)$, implying that s and s' cannot be distinguished by r. This finishes the proof for Theorem 27.

We complete the study of equivalence class recovery with a lower bound on the query complexity using binary queries.

▶ **Theorem 31.** For binary alphabet $\{0,1\}$, any algorithm to recover an arbitrary input sequence $s \in \{0,1\}^{\ell}$, where $0 \leq \ell \leq n$, up to equivalence class, by querying the DTW distance to a set of sequences, requires a query complexity of $\Omega(n)$.

Proof sketch. Note that in the query construction of Theorem 27, our constructed query set Q^* contains queries of all numbers of runs. The intuition for the proof of Theorem 31 is that, for each given constant-length interval of number of runs, we can construct a certain pair of input sequences which can only be distinguished by queries with a number of runs within this interval. E.g., it can be proved that $s_1 = 01^301^30^31^30^31^30$ and $s_2 = 01^3021^30^21^30^31^30$ can only be distinguished with queries with a number of runs within [4, 10]. Thus, an $\Omega(n)$ number of such constructed pairs of input sequences can correspond to $\Omega(n)$ disjoint intervals, yielding an $\Omega(n)$ lower bound for this problem.

We now construct a class of pairs of input sequences (s, s') such that s and s' can only be distinguished by queries q with a number of runs within $[\#RUNS(s)+c_1, \#RUNS(s)+c_2]$ for two constants $c_1 < c_2$. According to Theorem 25, as long as the constructed pair of input sequences (s, s') have the same number of runs, for a query q with more than

73:14 Recovery from Non-Decomposable Distance Oracles

 $\#_{\text{RUNS}(s)} \text{ number of runs, } d_{DTW}(q, s) \text{ and } d_{DTW}(q, s') \text{ are only determined by the query } q \text{ and } \#_{\text{RUNS}(s)}, \text{ and thus } q \text{ cannot distinguish } s \text{ and } s'. For a query } q \text{ with fewer than } \\ \#_{\text{RUNS}(s)} \text{ number of runs, } d_{DTW}(q, s) \text{ and } d_{DTW}(q, s') \text{ are reduced to two MSS instances.} \\ \text{Note that for different queries } q, \text{ the sequences (i.e., the first parameter) of MSS instances remain the same, while } \\ \#_{\text{RUNS}(s)}(q) \text{ determines the number of elements selected in the sequences of MSS instances (i.e., <math>\frac{\#_{\text{RUNS}(s)} - \#_{\text{RUNS}(q)}}{2}$). We hope to construct a pair of sequences seq and seq' such that $MSS(seq, 1) \neq MSS(seq', 1)$ and MSS(seq, x) = MSS(seq', x) for all x > 1: let seq and seq' be the sequences corresponding to MSS instances of s and s'; in this way, s and s' would still be distinguishable because $MSS(seq, x) \neq MSS(seq', x)$ for x = 1, but any query q with fewer than $\#_{\text{RUNS}(s)} - 4$ runs cannot distinguish s and s' because MSS(seq, x) = MSS(seq', x) for all $x \ge 2$, where $x = \frac{\#_{\text{RUNS}(s)} - \#_{\text{RUNS}(q)}{2}$.

5.3 Exact Recovery with Extra Character(s)

▶ **Theorem 32.** For a binary alphabet $\{0,1\}$, any algorithm to recover an arbitrary input sequence $s \in \{0,1\}^{\ell}$, where $0 \leq \ell \leq n$, by querying the DTW distance to a set of sequences of length $\mathcal{O}(n)$ requires a query complexity of $\Omega(\frac{n}{\log n})$.

Theorem 32 shows a lower bound on the exact recovery problem for DTW queries, which is given by an information-theoretic argument, referring back to the proof of Theorem 19.

Before diving into the exact recovery problem, we first introduce the concept of matching for the DTW distance. Given input sequence s and query sequence q, if we regard the characters in s and q as two sets of vertices, a matching M between s and q is a bipartite graph where each vertex has degree ≥ 1 and there do not exist "crossing" edges, i.e., $\nexists i < k$ and j > l such that the edges (s[i], q[j]) and (s[k], q[l]) both exist. Fig (a) and Fig (b) in Fig 1 are two examples of matchings. The weight of an edge (s[i], q[j]) in the matching is ||s[i] - q[j]||, and the cost of a matching is defined to be the sum of the weights of all edges in the matching. The DTW distance between sequences corresponds to the smallest cost of a matching between them. For the formal definitions of *Monotonic Sequence, Matching, DTW Matching, and Isomorphic Matching*, please refer to the full version of our paper.

We show that, if we augment the ability of our oracles by introducing extra characters, we can solve the DTW distance oracle recovery problem with optimal query complexity up to polylogarithmic factors.

5.3.1 With $\mathcal{O}(1)$ Extra Characters

▶ **Theorem 33.** For a binary alphabet $\{0,1\}$ and an input sequence $s := \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$, there exists an algorithm to recover the input sequence s, given $\mathcal{O}(n^2)$ query sequences \mathbf{Q} and the values $d_{DTW}(s,q)$ to each query sequence $q \in \mathbf{Q}$, where the query sequences are allowed to use only one extra character.

For the proof of Theorem 33, please refer to the full version of the paper.

▶ **Theorem 34.** For a binary alphabet $\{0,1\}$ and an input sequence $s := \{0,1\}^{\ell}$ where $0 \leq \ell \leq n$, there exists an algorithm to recover the input sequence s, given $\mathcal{O}(n)$ query sequences \mathbf{Q} of length $\leq n$ and the values $d_{DTW}(s,q)$ to each query sequence $q \in \mathbf{Q}$, where the query sequences are allowed to use only $\mathcal{O}(1)$ extra characters.

Proof sketch. We would like to construct a query set of size $\mathcal{O}(n)$ that can recover the input sequence using a DTW distance oracle. A natural idea is to retrieve information about the input sequence by differentiating (i.e., computing the difference between) the query results of neighbouring queries (i.e., queries only differing by 1 character). To achieve this, we construct a query set satisfying the following three properties:

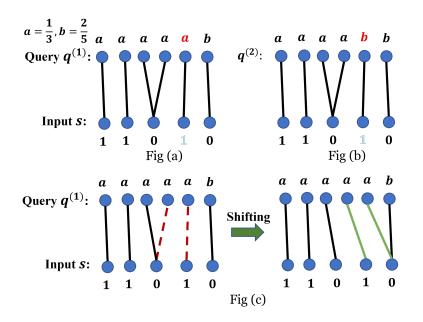


Figure 1 (a) Illustration of input-uniqueness and 0/1-uniqueness; (b) Illustration of isomorphism and differentiation, compared to Fig (a); (c) Illustration of shifting operation.

1) Isomorphism: The matchings corresponding to neighboring queries should be isomorphic. Fig (a) and (b) in Fig 1 is an example of an isomorphism, where only one character of the input sequence is changed, while the structure of both optimal matchings remains identical. With this property, we know that the difference between the query results of neighboring queries only reflects the effects of the different characters in neighboring queries. This property is the essence of guaranteeing the correctness of differentiation.

2) Input-uniqueness: Each character in the query sequence should be matched to exactly 1 character in the input sequence. With this property, we can use differentiation to get the information of a single character in the input sequence with a pair of neighboring queries. Note that if the differing character in the neighboring queries is matched to multiple characters in the input sequence, the difference in the query results can only reflect the sum of the costs over these characters, which makes exact recovery hard. Take Fig (a) and Fig (b) in Fig 1 as an example. Input-uniqueness is satisfied for both Fig (a) and Fig (b), since all characters in the query sequences of both figures have degree 1. Since Fig (a) has $\cos 3(1-a) + 2a + b$ while Fig (b) has $\cos 2(1-a) + 2a + (1-b) + b$, we know that Cost(Fig(a)) - Cost(Fig(b)) = b - a. By differentiation, we can infer that s[4] = 1; otherwise, if s[4] = 0, we would have Cost(Fig(a)) - Cost(Fig(b)) = (a-0) - (b-0) = a - b.

Combining properties 1) and 2), we note that each character in the input sequence can match to 1 or more characters in the query sequence, so we can obtain an expansion of the input sequence. Based on the example, Fig (a) and Fig (b) in Fig 1, we can obtain an expansion, 110010, of the input sequence. We can then infer that the input sequence is of the form $1^x 0^y 10$, where $x, y \in [1, 2]$. To recover the exact input sequence, we require more information given by the following third property.

3) 0/1-uniqueness: In an optimal matching w.r.t. our constructed queries, either all 0's or all 1's in the input sequence have degree 1. Using this property, we can locate the exact position of either all 0's or all 1's in the input sequence, and exactly recover the input

73:16 Recovery from Non-Decomposable Distance Oracles

sequence by combining the two cases. In example Fig 1, 1-uniqueness is satisfied in Fig (a) and Fig (b), while 0-uniqueness is not, since s[3] in both figures has degree 2. According to 1-uniqueness, we can reduce the form of the input sequence from $1^x 0^y 10$ to $110^y 10$. Similarly, we can construct another set of queries that satisfies 0-uniqueness to locate the positions of 0's in the input sequence, which determines y in this example.

Sequence Monotonicity \rightarrow **Input-uniqueness.** We observe that property 2) can be obtained from a *monotonic* design of the query sequences.

▶ Lemma 35. Given a monotonic sequence $r = r_1 r_2 ... r_n$ where

$$\min_{i \in [n]} \max\{|r_i - 0|, |r_i - 1|\} > \max_{i,j \in [n]} |r_i - r_j|,$$
(1)

for any input sequence s with length $\ell \leq n$, given a DTW matching M for (r, s), we have $deg(r_i) = 1$ for all elements r_i in r.

While we defer the formal proof to the full version of this work, the intuition for Lemma 35 is that, with the monotonic property and equation (1) guaranteed in our query construction, we can ensure that there do not exist characters $s[i] \in s$ and $r[j] \in r$ where deg(s[i]) > 1 and deg(r[j]) > 1 are satisfied at the same time. Otherwise, for such a pair of s[i] and r[j], we can always construct a matching with lower cost where one of their degrees is decreased to 1. Therefore, either all characters in s or all characters in r would have degree 1. Since $len(r) = n \ge len(s)$, we know that $deg(r_i) = 1$ for all characters r_i in r. Fig (a) and Fig (b) in Fig 1 satisfy sequence monotonicity since the query sequences in both figures are monotonic sequences of length n and $\min_{i \in [n]} \max\{|q_i - 0|, |q_i - 1|\} = \frac{3}{5} > (\frac{2}{5} - \frac{1}{3}) = \max_{i,j \in [n]} |q_i - q_j|$.

Sequence 0/1-preference $\rightarrow 0/1$ -uniqueness. We observe that property 3) can be guaranteed by the 0/1-preferred design of the query sequences. If all characters in the query sequence are less than (or greater than) $\frac{1}{2}$, then we can guarantee 1-uniqueness (or 0-uniqueness) of the input sequence. Intuitively, this would hold because, if all characters in the query sequence are less than (or greater than) $\frac{1}{2}$, matching them to 0's (or 1's) in the input sequence yields lower cost than matching to 1's (or 0's). Fig (a) and Fig (b) in Fig 1 satisfies 0-preference, since all characters in query sequences (either $a = \frac{1}{3}$ or $b = \frac{2}{5}$) are less than $\frac{1}{2}$.

Query Construction. We now propose the following design of the query sequence. We first need a single 0 query and a single 1 query to obtain the number of 1's and 0's in the input sequence. Let a, b be two fractional characters that satisfy $0 < b - a < a < b < \frac{1}{2}$ and the denominators of a, b are co-prime. Without loss of generality, we can assume $a = \frac{1}{3}$ and $b = \frac{2}{5}$. We will use a, b as the extra characters to construct the query sequences. In particular, the rest of the query sequences (other than the 0 query and the 1 query) consist of queries Q in the form of $q^{(i)} = a^{n-i}b^i$, where $i = 1, \ldots, n$. This query construction satisfies sequence monotonicity and sequence 0/1-preference properties. Now we need to prove it also satisfies isomorphism.

Notation clarification. For the rest of the proof, we will use $q^{(i)}$ to denote the *i*-th query in the query set Q and $q_i^{(i)}$ the *j*-th character in $q^{(i)}$.

▶ Lemma 36. For any input sequence s, there exists an isomorphic set of matchings \mathcal{M} , where $M_i \in \mathcal{M}$ is optimal for query $q^{(i)} \in \mathcal{Q}$.

Lemma 36 guarantees the isomorphism property of the constructed query set Q. Here we construct an isomorphic set of matchings $M_i \in \mathcal{M}$ such that only the first 0 in the input sequence has degree greater than 1, while all other characters in the matching are of degree 1. Fig (a) and Fig (b) in Fig 1 are instances of M_1 and M_2 , where the matchings in both figures are isomorphic to each other. Note that in this construction, the structure of the matchings is only determined by the position of the first 0 in the input sequence and the length of both sequences. Since all query sequences in Q have the same length, an isomorphism of constructed matchings is naturally guaranteed.

To prove the optimality of the M_i , we introduce the notion of a "shifting" operation. Consider two 0's in the input sequence. If any character between them has degree 1 and the first 0 has degree greater than 1, by running the shifting operation we decrease the degree of the first 0 by 1 and increase the degree of the last 0 by 1, while preserving the degree of all other characters. Fig (c) in Fig 1 illustrates an example of the shifting operation.

 \triangleright Claim 37. For our constructed query set Q, a shifting operation would not reduce the total cost of the matching.

 \triangleright Claim 38. Given input sequence s, query $q^{(i)} \in \mathcal{Q}$ and any optimal matching M_i^* between s and $q^{(i)}$, we can obtain M_i^* by applying a series of shifting operations to M_i .

Combining the above two claims, we can show that the M_i are always optimal, which proves Lemma 36. So far, the constructed query set satisfies three properties – isomorphism, input-uniqueness, and 0/1-uniqueness. Further details of our algorithm to recover the input sequence are given in the full version. For an algorithmic sketch, see Algorithm 1.

6 Recovery Using Fréchet Distance Oracle

Consider two sequences x and y ($x \neq y$) defined on the binary alphabet $\{0, 1\}$. It is not possible to distinguish any x and y under Fréchet distance, because the Fréchet distance between x and y can be 0. To make the problem non-trivial for Fréchet distance, we first define the concept of *equivalent sequences* under Fréchet distance.

Definition 39 (Equivalent Sequences under Fréchet Distance). Given two sequences x and y, we say x and y are equivalent if y is obtained by taking any bit in x and copying this bit contiguously any number of times. For any pair of equivalent sequences, the Fréchet distance between them is 0.

▶ **Theorem 40.** For a binary alphabet $\{0,1\}$, any algorithm to recover an arbitrary input sequence $s \in \{0,1\}^i$, where $0 \le i \le n$, by querying its Fréchet distance to a non-adaptive set of sequences, requires a query complexity of $\Omega(n)$.

Proof. For each length $1 \le i \le n$, there exists two non-equivalent sequences under Fréchet distance, which are 010101... and 101010..., yielding 2n mutually non-equivalent sequences. As the Fréchet distance oracle returns 0 when the input sequence and the query sequence are equivalent and 1 otherwise, we would need at least 2n - 1 queries to exactly recover the input sequence. If the number of queries is less than 2n - 1, we can always select 2 sequences from the 2n mutually non-equivalent sequences which are not covered by the queries, and these two sequences cannot be distinguished by the query sequences. This yields an $\Omega(n)$ lower bound on the query complexity.

▶ Theorem 41 (Extra Characters Are Not Helpful). Given two sequences s and s', if $d_F(s, s') = 0$, then any query q with extra characters cannot distinguish s and s'.

73:18 Recovery from Non-Decomposable Distance Oracles

Algorithm 1 Exact Recovery via Non-adaptive DTW Queries (with $\mathcal{O}(1)$ Extra Chars). **Input:** Non-adaptive query sequences $\mathbf{Q} = \{q^{(1)}, q^{(2)}, \dots, q^{(n+2)}\}$, where $q^{(n+1)} = 0$, $q^{(n+2)} = 1$ and the rest of the queries follow our construction; The DTW query results $\mathbf{R} = \{d_1, d_2, \dots, d_{n+2}\}.$ **Output:** The sequence *s* to be recovered. 1 Function RecoveryDTW(Q, R): if $d_{n+1} = 0$ then $\mathbf{2}$ 3 return s $\coloneqq 0^{d_{n+2}}$ if $d_{n+2} = 0$ then 4 return s := $1^{d_{n+1}}$ 5 $positions := [], coef_1 := 0$ 6 \triangleright Corresponding queries $q^{(i)} = a^{n-i}b^i$ for $i \in [1, n]$ do 7 $\operatorname{coef} \coloneqq d_i * 15 * 2 \mod 5$ 8 if $(coef - coef_1 + 5) \mod 5 = 2$ then 9 positions.append(0)10 else if $(coef - coef_1 + 5) \mod 5 = 3$ then 11 positions.append(1)12 $\operatorname{coef}_1 \coloneqq \operatorname{coef}$ 13 positions.reverse() 14 sequence := [], i := 015 $n_0 \coloneqq d_{n+2}, n_1 \coloneqq d_{n+1}$ $\mathbf{16}$ while positions/i = 1 do $\mathbf{17}$ sequence.append(1) $\mathbf{18}$ $i \, += 1$ 19 $i += n - n_0 - n_1$ 20 while i < n do 21 sequence.append(positions[i]) 22 23 i += 1**return** $s \coloneqq$ sequence $\mathbf{24}$

Proof. We show that, given an optimal matching between s and any query q, we can construct a matching between s' and q with the same cost, and vice versa. In this way, we know that $d_F(s,q) \ge d_F(s',q)$ and $d_F(s',q) \ge d_F(s,q)$, so $d_F(s,q) = d_F(s',q)$ and q cannot distinguish s and s'. Since $d_F(s,s') = 0$, s and s' have the same condensed expression. Suppose s and s'have k runs. Let q_{si} denote the substring in q which is matched to the *i*-th run of s, where $i \in [k]$. We can always match all q_{si} 's to the *i*-th run of s' instead, and obtain a matching between s' and q with a cost of $d_F(s,q)$. This finishes the proof of Theorem 41.

▶ **Theorem 42.** For a binary alphabet $\{0,1\}$ and two input sequences $s, s' \in \{0,1\}^i$ where $0 \le i \le n$ and s and s' are non-equivalent sequences under Fréchet distance, there exists an algorithm to distinguish the input sequences s and s', given $\mathcal{O}(n)$ query sequences **Q** and the Fréchet distance of s and s' to each query sequence $q \in \mathbf{Q}$.

Proof. We first show that for each length $0 \le i \le n$, there are only two non-equivalent sequences under Fréchet distance, which are 010101... and 101010... sequences, viz., we can identify two non-equivalent sequences by specifying the sequence length i and the starting bit. Therefore, for a maximum length n, there are only 2n mutually non-equivalent sequences.

Given any two different sequences from this 2n-sized collection of non-equivalent sequences under Fréchet distance, we can use $\mathcal{O}(n)$ query sequences to distinguish them. That is, we can utilize the exact set of 2n non-equivalent sequences as the query sequences. If the query sequence p is exactly the input sequence q, the Fréchet distance between p and q is $d_F(p,q) = 0$. If the query sequence p is not equivalent to the input sequence q, then the Fréchet distance between p and q is $d_F(p,q) = 1$ because it is impossible to skip over a bit without paying cost 1. Therefore, 2n query sequences suffice to distinguish any two sequences from the non-equivalent sequence set and this finishes the proof.

This theorem shows that, if an input is in the collection of non-equivalent sequences under Fréchet distance, we can use $\mathcal{O}(n)$ queries to exactly recover this sequence given the query results under the Fréchet distance.

Related Work

7

A distance embedding [20] embeds sequences from the original distance metric space to other distance measures (usually l_p norms), such that the distance measurements in the original space can be preserved up to a factor of D, namely the distortion rate. The sequence distance embedding problem is related to our problem in the sense that, in our problem, we intend to recover the input sequence from a list of query results that are in the l_p space, which can be regarded as finding a special distance embedding. Existing works on the sequence distance embedding problem mainly focus on constructing such an embedding which can have a close approximation (viz., low distortion rate) and reduce the computational complexity (i.e., cost) on the new distance space. [5] shows a lower bound of 3/2 on the distortion rate of embedding edit distance into ℓ_p norm spaces. An improvement of $(\log n)^{\frac{1}{2}-o(1)}$ on this lower bound [25] has been further simplified and improved into $\Omega(\log n)$ by [27].

Distance embeddings can be used to estimate the distance on the complex metric space because the evaluation and computations on the new (simpler metric) space can be significantly faster [20]. Under the asymmetric query model (when estimating the edit distance between x and y, the algorithm has unrestricted power accessing x but limited power accessing y), [6] proposes a $(\log n)^{\mathcal{O}(1/\epsilon)}$ approximation algorithm that runs in $n^{1+\epsilon}$ time. [15] considers the alignment problem when estimating the edit distance (finding the sequence of edits between the estimated sequences) and presents an alignment with $(\log n)^{\mathcal{O}(1/\epsilon^2)}$ approximation in time $\tilde{\mathcal{O}}(n^{1+\epsilon})$. The sequence distance embedding problem has been investigated on other distance metrics as well, for example, the block edit distance [20] and the Ulam distance [16]. Existing work also shows embeddings from edit distance to the Hamming space [8, 14]. However, to the best of our knowledge, there is no prior work considering the embedding problem of the *DTW distance* and the *exact* recovery problem based on distance queries.

8 Open Problems and Research Directions

We initiate the study of an exact recovery problem of sequences using queries to a nondecomposable distance oracle. We show recovery algorithms for edit distance, DTW distance, and Fréchet distance, as well as a general adaptive algorithm for a wide class of distance oracles. We envision the following directions for future work.

First, for the edit distance, there is still a quadratic gap between the non-adaptive query complexity upper and lower bounds without extra characters. Closing this gap may require a deeper understanding about the properties of edit distance.

73:20 Recovery from Non-Decomposable Distance Oracles

Second, for the DTW distance, it remains unclear whether 1 extra character suffices for an $\mathcal{O}(n)$ non-adaptive upper bound, or we can have an $\Omega(n^2)$ non-adaptive lower bound with 1 extra character (our proof uses 2 extra characters).

Lastly, it would be interesting to consider the exact sequence recovery problem using the properties of specific distance metrics. For example, the Edit distance with Real Penalty (ERP) distance [17] which supports local time shifting in time series by the marriage of the ℓ_1 norm and edit distance, would be of interest. One can also consider other variants of our problem in terms of adaptive queries or approximate recovery in the presence of noise.

— 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 Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, and Kurt Mehlhorn. The query complexity of a permutation-based variant of Mastermind. *Discrete Applied Mathematics*, 260:28–50, 2019.
- 3 Pankaj K Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete fréchet distance in subquadratic time. SIAM Journal on Computing, 43(2):429–449, 2014.
- 4 M Aldridge, O Johnson, and J Scarlett. Group testing: An information theory perspective. Foundations and Trends in Communications and Information Theory, 15(3-4):196–392, 2019.
- 5 A Andoni, M Deza, A Gupta, P Indyk, and S Raskhodnikova. Lower bounds for embedding edit distance into normed spaces. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pages 523–526, 2003.
- 6 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pages 377–386. IEEE, 2010.
- 7 Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *European symposium on algorithms*, pages 52–63. Springer, 2006.
- 8 Djamal Belazzougui and Qin Zhang. Edit distance: Sketching, streaming, and document exchange. In 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pages 51–60. IEEE, 2016.
- 9 Vladimir Braverman, Moses Charikar, William Kuszmaul, David P Woodruff, and Lin F Yang. The one-way communication complexity of dynamic time warping distance. arXiv preprint, 2019. arXiv:1903.03520.
- 10 Nader H Bshouty. Optimal algorithms for the coin weighing problem with a spring scale. In *COLT*, volume 2009, page 82, 2009.
- 11 Maike Buchin, Anne Driemel, Koen van Greevenbroek, Ioannis Psarros, and Dennis Rohde. Approximating length-restricted means under dynamic time warping. arXiv preprint, 2021. arXiv:2112.00408.
- 12 Xingyu Cai, Tingyang Xu, Jinfeng Yi, Junzhou Huang, and Sanguthevar Rajasekaran. DTWNet: a dynamic time warping network. *Advances in neural information processing* systems, 32, 2019.
- 13 David G Cantor and WH Mills. Determination of a subset from certain combinatorial properties. Canadian Journal of Mathematics, 18:42–48, 1966.
- 14 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucky. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 712–725, 2016.
- 15 Moses Charikar, Ofir Geri, Michael P Kim, and William Kuszmaul. On estimating edit distance: Alignment, dimension reduction, and embeddings. In *ICALP*, 2018.

- 16 Moses Charikar and Robert Krauthgamer. Embedding the Ulam metric into l_1 . Theory of Computing, 2(1):207–224, 2006.
- 17 Lei Chen and Raymond Ng. On the marriage of l_p -norms and edit distance. In *Proceedings* of the Thirtieth international conference on Very large data bases-Volume 30, pages 792–803, 2004.
- 18 Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint*, 2019. arXiv:1902.02918.
- 19 Amin Coja-Oghlan, Oliver Gebhard, Max Hahn-Klimroth, and Philipp Loick. Optimal group testing. In *Conference on Learning Theory*, pages 1374–1388. PMLR, 2020.
- 20 Graham Cormode. Sequence distance embeddings. PhD thesis, Department of Computer Science, 2003.
- 21 Robert Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.
- 22 Manuel Fernández, David P Woodruff, and Taisuke Yasuda. The query complexity of mastermind with l_p distances. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 2019.
- 23 Zhuangfei Hu, Xinda Li, David P Woodruff, Hongyang Zhang, and Shufan Zhang. Recovery from non-decomposable distance oracles. *arXiv preprint*, 2022. arXiv:2209.05676.
- 24 Zilin Jiang and Nikita Polyanskii. On the metric dimension of cartesian powers of a graph. Journal of Combinatorial Theory, Series A, 165:1–14, 2019.
- 25 Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), pages 101–110. IEEE, 2005.
- **26** Donald E Knuth. The computer as Master Mind. *Journal of Recreational Mathematics*, 9(1):1–6, 1976.
- 27 Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into l₁. SIAM Journal on Computing, 38(6):2487–2498, 2009.
- 28 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In Frank Thomson Leighton and Allan Borodin, editors, Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA, pages 596-605. ACM, 1995. doi:10.1145/225058.225277.
- 29 Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy*, pages 656–672, 2019.
- 30 Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady, volume 10, pages 707–710, 1966.
- 31 Ming Li and Paul MB Vitányi. Combinatorics and Kolmogorov complexity. In 1991 Proceedings of the Sixth Annual Structure in Complexity Theory Conference, pages 154–155. IEEE Computer Society, 1991.
- 32 Juan A Rodríguez-Velázquez, Ismael G Yero, Dorota Kuziak, and Ortrud R Oellermann. On the strong metric dimension of Cartesian and direct products of graphs. *Discrete Mathematics*, 335:8–19, 2014.
- 33 Nathan Schaar, Vincent Froese, and Rolf Niedermeier. Faster binary mean computation under dynamic time warping. arXiv preprint, 2020. arXiv:2002.01178.
- 34 Harold S Shapiro and NJ Fine. E1399. *The American Mathematical Monthly*, 67(7):697–698, 1960.
- 35 Staffan Söderberg and Harold S Shapiro. A combinatory detection problem. The American Mathematical Monthly, 70(10):1066–1070, 1963.
- 36 Roman Vershynin. Lectures in geometric functional analysis. Unpublished manuscript. Available at https://www.math.uci.edu/~rvershyn/papers/GFA-book.pdf, 3(3):3-3, 2011.
- 37 Chao Wang, Qing Zhao, and Chen-Nee Chuah. Optimal nested test plan for combinatorial quantitative group testing. *IEEE Transactions on Signal Processing*, 66(4):992–1006, 2017.

73:22 Recovery from Non-Decomposable Distance Oracles

A Coordinate Descent Algorithm Instantiation

Now we briefly discuss how we apply our Coordinate Descent algorithm to all three distances we consider in this paper by justifying the two conditions hold.

Edit distance. For condition 2, we know that $\forall q, q', dist(q, q') \leq n$ since the maximum length of q and q' is n. For condition 1, in each iteration, we consider a set Q that contains all sequences that can be transformed from q' by inserting, deleting or substituting one character in q' (edit operations). Note that |Q| cannot exceed (n + 1) + n + n = 3n + 1. We claim that there exists a q'' in Q such that dist(q, q') > dist(q, q''). Let dist(q', q) = d. By the definition of edit distance, there exists a chain of edit operations of length d that transforms q' to q, resulting in a list of intermediate sequences q_1, \ldots, q_{d-1} . Note that $dist(q', q) > dist(q_1, q)$, otherwise $dist(q_1, q) \geq d$. However, the chain implies we can transform q_1 to q in d - 1edit operations, which leads to a contradiction. Since $q_1 \in Q$, we can find q_1 satisfying the condition in 3n + 1 searches. Therefore, the algorithm is guaranteed to recover the input in $\mathcal{O}(n^2)$ steps.

DTW distance. For DTW distance, condition 2 holds since $\forall q, q', dist(q, q') \leq n$. For condition 1, consider the $\#_{\text{RUNS}}(x)$ in q' and q. If $\#_{\text{RUNS}}(x)$ of q' < q, then either adding an (arbitrary length) run to the start or the end of q' will decrease the DTW distance from q. On the other hand, if $\#_{\text{RUNS}}(x)$ of q' > q, then either deleting a run from the start or the end of q' will decrease the DTW distance from q. If $\#_{\text{RUNS}}(x)$ of q' = q and $dist(q', q) \neq 0$, we can still decrease the distance from q by either adding/deleting a run to the start/end of the sequence. Therefore, the algorithm is guaranteed to recover the input in $\mathcal{O}(n)$ steps.

Fréchet distance. Condition 2 holds since $\forall q, q', dist(q, q') \leq 1$. For condition 1, enumerating 2n non-equivalent sequences, (i.e., 010101... and 101010...) guarantees finding q'' such that dist(q,q') > dist(q,q'') = 0. Therefore, the algorithm terminates in $\mathcal{O}(n)$ steps.