Color Distance Oracles and Snippets: Separation Between Exact and Approximate Solutions

Noam Horowicz
Bar-Ilan University, Ramat Gan, Israel

Tsvi Kopelowitz
Bar-Ilan University, Ramat Gan, Israel

Abstract

In the *snippets* problem, the goal is to preprocess a text T so that given two pattern queries, P_1 and P_2, one can quickly locate the occurrences of the two patterns in T that are closest to each other, or report the distance between these occurrences. Kopelowitz and Krauthgamer [CPM2016] showed upper bound tradeoffs and conditional lower bounds tradeoffs for the snippets problem, by utilizing connections between the snippets problem and the problem of constructing a color distance oracle (CDO), which is a data structure that preprocess a set of points with associated colors so that given two colors c and c' one can quickly find the (distance between the) closest pair of points where one has color c and the other has color c'. However, the existing upper bound and lower bound curves are not tight.

Inspired by recent advances by Kopelowitz and Vassilevska-Williams [ICALP2020] regarding tradeoff curves for Set-disjointness data structures, in this paper we introduce new conditionally optimal algorithms for a $(1+\varepsilon)$ approximation version of the snippets problem and a $(1+\varepsilon)$ approximation version of the CDO problem, by applying fast matrix multiplication. For example, for CDO on n points in an array, if the preprocessing time is $\tilde{O}(n^a)$ and the query time is $\tilde{O}(n^b)$ then, assuming that $\omega = 2$ (where ω is the exponent of n in the runtime of the fastest matrix multiplication algorithm on two squared matrices of size $n \times n$), we show that approximate CDO can be solved with the following tradeoff

$$\begin{cases} a + 2b = 2 & \text{if } 0 \le b \le \frac{1}{3} \\ 2a + b = 3 & \text{if } \frac{1}{3} \le b \le 1. \end{cases}$$

Moreover, we prove that for exact CDO on points in an array, the algorithm of Kopelowitz and Krauthgamer [CPM2016], which obtains a tradeoff of $a + b = 2$, is essentially optimal assuming that the *strong all-pairs shortest paths* hypothesis holds for randomized algorithms. Thus, we demonstrate that the exact version of CDO is strictly harder than the approximate version. Moreover, this separation carries over to the snippets problem.

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

Keywords and phrases data structures, fast matrix multiplication, fine-grained complexity, pattern matching, distance oracles

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.72

Related Version *Full Version*: https://arxiv.org/abs/2507.04578 [12]

Funding Supported by a BSF grant 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).

Acknowledgements The authors thank an anonymous reviewer who suggested some ideas for tightening the lower bounds proved in Theorems 8 and 9.

1 Introduction

In the *snippets* problem, introduced by Kopelowitz and Krauthgamer [17], the goal is to preprocess a text T so that given two pattern queries, P_1 and P_2, one can quickly locate the locations of the two patterns in T that are closest to each other, or report the distance between these locations. The snippets problem is motivated by many common text indexing applications, such as searching a corpus of documents for two query keywords. Often, the relevance of a document to the query is measured by the proximity of the two keywords within the document. The term *snippet* is derived from search engines often providing each result with a snippet of text from the document with the two keywords close to each other.

Kopelowitz and Krauthgamer [17] designed a tradeoff algorithm for the snippets problem based on a connection to a problem on colored points in metric spaces, which we define next.

Colored points and color distances

Let $\mathbf{M}$ be a metric space with distance function $d(\cdot,\cdot)$. Let $S \subseteq \mathbf{M}$ be a set of points where $|S| = n$. Let $\mathcal{C}$ be a set of *colors*. For sake of convenience we assume that $\mathcal{C} = [|\mathcal{C}|]$. Each point $p \in S$ has an associated color $c_p \in \mathcal{C}$. For a color $c \in \mathcal{C}$, let $P_S(c) = \{p \in S | c_p = c\}$. When S is clear from context we abuse notation and denote $P(c) = P_S(c)$.

For sets of points $A, B \subset \mathbf{M}$ and a point $p \in M$, let $d(p, A) = \min_{p' \in A}\{d(p, p')\}$ and let $d(A, B) = \min_{p' \in A}\{d(p', B)\} = \min_{p' \in A, \hat{p} \in B}\{d(p', \hat{p})\}$. We extend the definition of d to inputs of colors as follows. For colors $c, c' \in \mathcal{C}$ and point $p \in \mathbf{M}$, let $d(p, c) = d(p, P(c))$ and let $\delta_{c,c'} = d(c, c') = d(P(c), P(c'))$. We say that $\delta_{c,c'}$ is the *color distance* between c and c'. A natural problem is to construct a *color distance oracle* (CDO), which is defined as follows.

▶ **Problem 1** (The Color Distance Oracle (CDO) problem [17]). *Given a set $S \subseteq \mathbf{M}$ of n colored points with colors from $\mathcal{C}$, preprocess S to support the following query: given $c, c' \in \mathcal{C}$, return $\delta_{c,c'}$.*

In this paper we consider metrics defined by locations in an array of size s, and so the metric is the set[1] $[s]$, and the distance function of two points p and q is $d(p, q) = |p - q|$.

Multi-colored points and color hierarchies

A natural generalization of a colored set is to allow each $p \in S$ to be associated with a nonempty set of colors $c(p) \subseteq \mathcal{C}$, and in such a case S is said to be *multi-colored*. This version of the CDO problem is called the *multi-color distance oracle* (MCDO) problem. In general, representing $c(p)$ costs $\Theta(|c(p)|)$ space by explicitly listing all colors in $c(p)$. Thus, the input size is $n = \sum_{p \in S} |c(p)|$.

Nevertheless, there are interesting cases in which the lists of colors for each point are not required to be given explicitly. One such example, which we focus on, is when for every two colors $c, c' \in \mathcal{C}$, either one of the sets $P(c)$ and $P(c')$ contains the other, or the two sets are disjoint. In such a case, we say that $\mathcal{C}$ has a *color hierarchy* with respect to S (a formal terminology is that $\{P(c)\}_{c \in \mathcal{C}}$ is a laminar family), represented by a rooted forest (i.e., each tree has a root) T_S of size $O(|\mathcal{C}|)$. Each color c is associated with a vertex u_c in T_S, such that the descendants of u_c are exactly all the vertices $u_{c'}$ whose color c' satisfies $P(c') \subseteq P(c)$. We convert the forest T_S to a rooted tree by adding a dummy root vertex and making the dummy root the parent of all of the roots of the trees in the forest.

[1] Throughout this paper for a positive integer k we denote $[k] = \{1, 2, , \ldots, k\}$.

With the aid of T_S, a multi-colored set S that admits a color hierarchy can be represented using only $O(n + |\mathcal{C}|)$ machine words, because it suffices to store T_S and associate with each point p just one color c (the color with the lowest corresponding vertex in T_S); the other colors of p are implicit from T_S (the colors on the path from u_c to the root of T_S). Notice that $|\mathcal{C}| > n$ implies that there are at least two colors with the same point set. Thus, we make the simplifying assumption that all colors have different point sets[2], and so the input size is $O(n)$.

Thus, a natural extension of Problem 1 is the following.

▶ **Problem 2** (The Multi-Color Distance Oracle with a Color Hierarchy (MCDOCH) problem [17]). *Given a set $S \subseteq \mathbf{M}$ of n multi-colored points with colors from $\mathcal{C}$ which form a color hierarchy with respect to S, preprocess S to support the following query: given $c, c' \in \mathcal{C}$, return $\delta_{c,c'}$.*

Let a and b be real numbers such that the preprocessing and query times of a MCDOCH algorithm are $\tilde{O}(n^a)$[3] and $\tilde{O}(n^b)$, respectively. A tradeoff algorithm for the MCDOCH problem was presented in [17], where given parameter $1 \leq \tau \leq n$, the query time is $\tilde{O}(n^b) = \tilde{O}(\tau)$ and the preprocessing time is $\tilde{O}(n^a) = \tilde{O}(n^2/\tau) = \tilde{O}(n^{2-b})$. When ignoring sub-polynomial factors, this upper bound tradeoff is given by $a + b = 2$ where $0 \leq b \leq 1$ and $1 \leq a \leq 2$. Notice that the same tradeoff applies to the CDO problem.

In addition to the tradeoff algorithms, [17] proved a 3SUM based conditional lower bound (CLB) tradeoff[4] of $a + 2b \geq 2$ for the CDO problem (and hence for the MCDOCH problem), even when $\mathbf{M}$ is defined by locations in an array. Their CLB holds also for approximate versions of the CDO problem (see Theorem 8 in [17]), where for a fixed $\alpha > 1$, the answer to a color distance query between c and c' is required to be between $\delta_{c,c'}$ and $\alpha \cdot \delta_{c,c'}$.

Solving the Snippets problem

By designing a reduction from the snippets problem to the MCDOCH problem, [17] were able to design an algorithm for the snippets problem with essentially the same tradeoff: for a chosen parameter $1 \leq \tau \leq |T|$ their snippets algorithm uses $\tilde{O}(|T|^2/\tau)$ preprocessing time and answers queries in $\tilde{O}(|P_1| + |P_2| + \tau)$ time. The main idea in the reduction is to construct the suffix tree of T, assign a color to each vertex in the suffix tree, and for each leaf ℓ in the suffix tree, the corresponding location in T is assigned all of the colors on the path from ℓ to the suffix tree root. It is straightforward to see that the colors define a hierarchy and that T_S is exactly the suffix tree.

A complexity gap

The upper bound curve of $a+b = 2$ and the CLB tradeoff of $a+2b \geq 2$ form a complexity gap for the snippets and (approximate) CDO problems, which was left open by [17]. Moreover, based on the combinatorial Boolean Matrix Multiplication (BMM) hypothesis, [17, Theorem 8] implies that any algorithm that beats the $a + b = 2$ curve, even for approximate versions, must use non-combinatorial techniques, such as fast matrix multiplication (FMM) algorithms.

We remark that the CLB tradeoff given in [17] is based on a reduction from a SetDisjointness problem to the (approximate) CDO problem, and at the time [17] was published, the SetDisjointness problem was known to exhibit the same upper bound tradeoff and 3SUM

[2] If we happen to have several colors with the same point set, we ignore all of them but one during preprocessing, and use a straightforward mapping from the original coloring to the reduced coloring during query time.

[3] Throughout this paper we use the notation $\tilde{O}()$ to suppress sub-polynomial factors.

[4] The CLB tradeoff stated here is straightforward to derive from the statement of Theorem 4 in [17].

based CLB tradeoff [18]. However, recently Kopelowitz and Vassilevska-Williams [19] closed this complexity gap by, among other ideas, using FMM techniques. Thus, a natural question to ask is whether FMM can assist in obtaining improved algorithms for CDO problems.

Our results

We close the complexity gap for the $(1+\varepsilon)$ approximation versions of CDO and $MCDOCH$, where $\mathbf{M}$ is defined by locations in an array. Formally, the problems are defined as follows.

▶ **Problem 3** (The $(1+\varepsilon)$-Approximate Color Distance Oracle (ACDO) problem on an array)**.** *Let $\mathbf{M}$ be a metric defined by locations in an array of size n. Given a set $S \subset \mathbf{M}$ of n colored points with colors from $\mathcal{C}$ and a fixed real $0 \leq \varepsilon \leq 1$, preprocess S to support the following query: given two colors $c, c' \in \mathcal{C}$, return a value $\hat{\delta}$ such that $\delta_{c,c'} \leq \hat{\delta} \leq (1+\varepsilon)\delta_{c,c'}$.*

▶ **Problem 4** (The $(1+\varepsilon)$-Approximate Multi-Color Distance Oracle problem with a Color Hierarchy (AMCDOCH) on an array)**.** *Let $\mathbf{M}$ be a metric defined by locations in an array of size $\Theta(n)$. Given a set $S \subseteq \mathbf{M}$ of n multi-colored points with colors from $\mathcal{C}$ which form a color hierarchy with respect to S, and a fixed real $0 \leq \varepsilon \leq 1$, preprocess S to support the following query: given two colors $c, c' \in \mathcal{C}$, return a value $\hat{\delta}$ such that $\delta_{c,c'} \leq \hat{\delta} \leq (1+\varepsilon)\delta_{c,c'}$.*

Our first main result is a new tradeoff algorithm for ACDO, which is stated by the following theorem. Notice that the time complexities presented are dependent on $\omega \geq 2$, which is the exponent of n in the runtime of the fastest FMM algorithm on two squared matrices of size $n \times n$. The currently best upper bound on ω is $\omega < 2.371339$ given by [2]. However, for clarity, we choose to express our tradeoffs in terms of ω. We remark that ACDO is a special case of AMCDOCH, and so Theorem 6 implies Theorem 5. However, Theorem 5 is used in the proof of Theorem 6, so we explicitly state both theorems.

▶ **Theorem 5.** *For any real $0 \leq b \leq 1$, there exists an ACDO algorithm with preprocessing time $\tilde{O}(n^a)$ and query time $\tilde{O}(n^b)$ where*

$$\begin{cases} a + \dfrac{2}{\omega-1}b = 2 & \text{if } 0 \leq b \leq \dfrac{\omega-1}{\omega+1} \\ \dfrac{2}{\omega-1}a + b = \dfrac{\omega+1}{\omega-1} & \text{if } \dfrac{\omega-1}{\omega+1} \leq b \leq 1. \end{cases}$$

▶ **Theorem 6.** *For any real $0 \leq b \leq 1$, there exists an AMCDOCH algorithm with preprocessing time $\tilde{O}(n^a)$ and query time $\tilde{O}(n^b)$ where*

$$\begin{cases} a + \dfrac{2}{\omega-1}b = 2 & \text{if } 0 \leq b \leq \dfrac{\omega-1}{\omega+1} \\ \dfrac{2}{\omega-1}a + b = \dfrac{\omega+1}{\omega-1} & \text{if } \dfrac{\omega-1}{\omega+1} \leq b \leq 1. \end{cases}$$

Combining Theorem 6 with the reduction of the snippets problem to the MCDOCH problem given in [17], we obtain the following tradeoff for a $1+\varepsilon$ approximation version of the snippets problem.

▶ **Theorem 7.** *For any fixed $0 < \varepsilon \leq 1$, and $1 \leq a \leq 2$, there exists an algorithm that preprocesses a text T in $\tilde{O}(|T|^a)$ time such that given two pattern strings P_1 and P_2 where the distance between the closest occurrences of the two patterns in T is δ, the algorithm returns a value $\hat{\delta}$ such that $\delta \leq \hat{\delta} \leq (1+\varepsilon)\delta$ in $\tilde{O}(|P_1| + |P_2| + |T|^b)$ time, where*

$$\begin{cases} a + \dfrac{2}{\omega-1}b = 2 & \text{if } 0 \leq b \leq \dfrac{\omega-1}{\omega+1} \\ \dfrac{2}{\omega-1}a + b = \dfrac{\omega+1}{\omega-1} & \text{if } \dfrac{\omega-1}{\omega+1} \leq b \leq 1. \end{cases}$$

We remark that it is straightforward to adapt our ACDO and AMCDOCH algorithms (and hence our new approximate snippets algorithm) to return the two points (one of each color) that define the distance being returned (or the locations of the two patterns in the approximate snippets problem). For details, see the full version [12].

A complexity separation between exact and approximate solutions

We remark that by combining the reduction given in [17, Theorem 8] from SetDisjointness to the CDO problem with the CLBs for SetDisjointness given in [19, Theorem 6] for the case of $\omega = 2$ (which some researchers believe should be obtainable), Theorems 5, 6 and 7 are all conditionally optimal[5] (up to subpolynomial factors). A natural question to ask is whether one can design exact algorithms with the same tradeoff bounds. We provide evidence that this is not possible, based on a *strong* version of the popular all-pairs shortest paths (APSP) conjecture [7], thereby demonstrating a separation between CDO and ACDO.

The Strong-APSP hypothesis, introduced by Chan, Vassilevska-Williams and Xu [7], states that for a graph with $\hat{n}$ vertices[6], even if all of the edge weights are in[7] $[\hat{n}]$, APSP still does not have a truly subcubic algorithm. A closely related problem to APSP is the $(min, +)$-*matrix product* problem, where the input is two $\hat{n}$ by $\hat{n}$ matrices $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$, and the output is the matrix $D = \{d_{i,j}\}$ where $d_{i,j} = \min_{1 \leq k \leq \hat{n}}(a_{i,k} + b_{k,j})$. Shoshan and Zwick [24] showed that solving APSP with weights $[M]$ is equivalent (in the sense of having the same runtime) to solving $(min, +)$-matrix product with entries in $[O(M)]$. Thus, the Strong-APSP hypothesis implies that there is no truly subcubic time $(min, +)$-matrix product algorithm even when all of the entries are in $[\hat{n}]$.

By reducing $(min, +)$-matrix product with entries in $[\hat{n}]$ to MCDO, we are able to prove the following CLB in Section 6, which matches the upper bound given in [17].

▶ **Theorem 8.** *Assuming the Strong-APSP conjecture, any algorithm for MCDO on n points in an array of size $O(n)$ with $\tilde{O}(n^a)$ preprocessing time and $\tilde{O}(n^b)$ query time, respectively, must obey $a + b \geq 2$.*

Moreover, we are able to leverage the ideas used in the proof of Theorem 8 to reduce $(min, +)$-matrix product with entries in $[\hat{n}]$ to CDO via a randomized reduction, thereby obtaining a CLB under the assumption that the Strong-APSP hypothesis holds even for randomized algorithms (either in expectation or with high probability).

▶ **Theorem 9.** *Assuming the Strong-APSP hypothesis for randomized algorithms, any algorithm for CDO on n points in an array of size $O(n)$ with $\tilde{O}(n^a)$ preprocessing time and $\tilde{O}(n^b)$ query time, respectively, must obey $a + b \geq 2$.*

The proof of Theorem 9 is based on the proof of Theorem 8, combined with probabilistic techniques in order to create instances that are not multi-colored. Due to space considerations, the proof is given in the full version [12]

[5] For Theorems 5 and 6, the optimality follows directly from applying [17, Theorem 8] and [19, Theorem 6]. For Theorem 7, the optimality follows since an approximate solution for the snippets problem can be used to solve ACDO on an array as follows: treat each color as a character, and then the array becomes a string. Preprocess the string using the snippets algorithm, so that given an ACDO query on two colors $c, \hat{c}$, query the snippets algorithm with patterns $P = c$ and $\hat{P} = \hat{c}$. Thus, the ACDO lower bound applies to approximate snippets.

[6] We use $\hat{n}$ to differentiate from $n = |S|$ in the various CDO problems.

[7] Actually, in [7] the weights are bounded by $\hat{n}^{3-\omega}$, but since $\omega \geq 2$ we can bound the weights by $\hat{n}$.

Theorem 9 combined with Theorem 5 implies that the exact version of CDO is strictly harder than the approximate version. Moreover, we remark that the CLBs for CDO apply also to the snippets problem, since the special case of $\mathbf{M}$ defined by locations in an array is captured by the snippets problem when P_1 and P_2 are each a single character. Thus, the CLB of Theorem 9 applies also to the snippets problem, and so the exact version of the snippets problem is also strictly harder than the approximate version.

Additional Related Work

A problem closely related to the CDO is the (approximate) vertex-labeled distance oracles for graphs (VLDO) problem, where the goal is to preprocess a colored graph G with n vertices, so that given a vertex query v and a color c, one can quickly return (an approximation of) $d(v, c)$. Hermelin, Levy, Weimann, and Yuster [11] introduced the VLDO problem and designed a solution using $O(kn^{1+1/k})$ expected space, with stretch factor $4k-5$ and $O(k)$ query time. They also showed how to reduce the space usage to $O(kN^{1/k})$ but the stretch factor is exponential in $2k-21$. Chechik [8] later showed how to lower the stretch back to $4k-5$. For planar graphs, Evald, Fredslund-Hansen, and Wulff-Nilsen [10] designed a near-optimal exact tradeoff using $n^{1+o(1)}$ space with $\tilde{O}(1)$ query time, or $\tilde{O}(n)$ space with $n^{o(1)}$ query time. Li, Ma, and Ning [21] designed a $1+\varepsilon$ approximation for planar graphs.

2 Preliminaries and Algorithmic Overview

Let $[n] = \{1, 2, \ldots, n\}$. Let $S \subset \mathbb{Z}$ be a set of n integers. Let $\max(S)$ be the largest integer in S and let $\min(S)$ be the smallest integer in S. For integers $i, j \in [n]$ where $i < j$, let $S[i, j] = \{p \in S : i \leq p \leq j\}$.

Our algorithms make use of the following Nearest Neighbor Search (NNS) data structures.

▶ **Problem 10** (The Nearest Neighbour Search problem [1, 4, 16, 25]). *Given a set $S \subseteq \mathbf{M}$ of size n, preprocess S to support the following query: given an integer p, return* $\operatorname{argmin}_{p' \in S}\{d(p, p')\}$.

▶ **Problem 11** (The Range Nearest Neighbour Search (RNNS) problem [5, 6, 9, 14, 15, 20, 22, 23]). *Given a set S of n integers, preprocess S to support the following query: given $i, j \in [n]$ and an integer p, return* $\operatorname{argmin}_{p' \in S[i,j]}\{d(p, p')\}$.

2.1 Algorithmic Overview

Generic (Approximate) CDO Algorithm

Our algorithm for ACDO is based on a generic algorithm whose structure follows the structure of the algorithm in [17] for the exact CDO problem. The generic algorithm is a classic heavy-light algorithm; our algorithms are a new *implementation* of the generic algorithm.

For an integer parameter $0 \leq \tau \leq n$, a color $c \in \mathcal{C}$ is said to be *heavy* if $\left|P(c)\right| \geq \tau$ and *light* otherwise. Let $\mathcal{H} = \{h_1, h_2, \ldots, h_{|\mathcal{H}|}\}$ be the set of heavy colors, and let $\mathcal{L}$ be the set of light colors. Notice that $|\mathcal{H}| \leq \frac{n}{\tau}$. In the preprocessing phase, for each color $c \in \mathcal{C}$, the algorithm stores $P(c)$ in an NNS data structure, denoted by NNS_c. In addition, the algorithm pre-computes a matrix $E^* = \{e^*_{i,j}\}$ of size $|\mathcal{H}| \times |\mathcal{H}|$, such that $\delta_{h_i,h_j} \leq e^*_{i,j} \leq (1+\varepsilon)\delta_{h_i,h_j}$. An ACDO query on $c, c' \in \mathcal{C}$ is processed as follows. If both $c, c' \in \mathcal{H}$, then, without loss of generality, $c = h_i$ and $c' = h_j$. In such a case the algorithm returns $e^*_{i,j}$, which is a $(1+\varepsilon)$ approximation of δ_{h_i,h_j}. Otherwise, without loss of generality, C is a light color, and the algorithm returns $\min_{\hat{p} \in P(c)}\{d(\hat{p}, c')\} = \delta_{c,c'}$, by performing $\left|P(c)\right| \leq \tau$ NNS queries on $\mathrm{NNS}_{c'}$, one for each point in C.

Time complexity

The preprocessing and query time costs of the generic algorithm depend on the implementation of the NNS data structure, and the time used for computing matrix E^*. Thus, we express the time complexity of the generic algorithm as a function of $T_{p,\mathrm{NNS}}(t)$, $T_{q,\mathrm{NNS}}(t)$, and $T_{E^*}(\mathcal{H})$, which are the preprocessing time and query time of the NNS data structure on a set of size t, and the time used to compute E^*, respectively. In the preprocessing phase, the algorithm computes E^* and creates an NNS data structure for each color in $\mathcal{C}$. Thus, the preprocessing time cost is

$$O(T_{E^*}(\mathcal{H}) + \sum_{c\in\mathcal{C}} T_{p,\mathrm{NNS}}(P(c))).$$

In the query phase, the time cost is dominated by executing at most τ NNS queries on a set of size at most n, and so the time cost is $O(\tau \cdot T_{q,\mathrm{NNS}}(n))$.

Constructing E^* and solving ACDO on an array

In Section 3 we describe an efficient algorithm for constructing E^* when $\mathbf{M}$ is the metric of integers. Our algorithm is structured as follows. For pairs of heavy colors whose color distance is smaller than some threshold, the algorithm computes their distance via a straightforward brute-force computation. The more challenging part is dealing with pairs of heavy colors with larger color distance. Our approach is to compute a series of Boolean matrices with specially designed properties, so that after the computation of the matrices we are able scan the matrices and deduce a $(1+\varepsilon)$ approximation for every pair of colors with a rather large distance color. We remark that to compute the matrices used for the approximations, our algorithm makes use of FMM, which, as noted in Section 1, is required for obtaining improvements in runtime compared to the exact solution of [17].

To complete the proof of Theorem 5, in Section 4 we analyze the runtime of the generic algorithm with the runtime of constructing E^* when $\mathbf{M}$ is defined by locations in an array, and by plugging in known NNS data structures.

Solving AMCDOCH on an array

In Section 5 we prove Theorem 6. Our algorithm is a reduction from AMCDOCH on an array to ACDO on an array. The structure of our reduction somewhat resembles the structure of the MCDOCH algorithm of [17]. In particular, in the algorithm of [17] they partition the points into sets of size τ, preprocess each set into an RNNS data structure, and for each pair of sets apply τ RNNS queries to compute the distance between the sets. To obtain a faster runtime, we compute the distance between every pair of sets by recoloring the input points using the sets as a new color scheme and then applying our ACDO algorithm with the new color set. One important property of our ACDO algorithm is that when a query takes place between two heavy colors, the query time is constant since the algorithm just looks up a single value in E^*. It turns out that essentially all of the colors in the new color set are heavy, and so after the preprocessing phase of our ACDO algorithm, we are able to compute the distance between a pair of sets in $O(1)$ time. This property turns out to be helpful for our AMCDOCH algorithm, which is described in Section 5.

3 Computing E^* when $\mathbf{M}$ is integers

In this section, we show how to compute matrix E^* when our metric $\mathbf{M}$ is one dimensional, and for $p, p' \in \mathbf{M}$ we have $d(p, p') = |p - p'|$. We make the simplifying assumption that $\min(S) = 1$ since, otherwise, one could shift S by subtracting $\min(S) - 1$ from each point.

Let W be a positive integer parameter that will be determined later. The construction algorithm for E^* has two conceptual parts. The first part deals with pairs of heavy colors with distance at most $\frac{1+2\varepsilon}{\varepsilon}W$, by computing the distances exactly using a brute-force method. The second part deals with pairs of heavy colors with distance greater than $\frac{1+2\varepsilon}{\varepsilon}W$. In this case, the algorithm computes an approximation of the distances by utilizing a series of matrices defined as follows.

Let $\ell_0 = \lfloor \log_{1+\varepsilon}(\frac{1}{\varepsilon}) \rfloor + 1$ and let $\ell_{max} = \lceil \log_{1+\varepsilon}(\max(S)) \rceil$. Let $A = \{a_{i,j}\}$ be a Boolean $|\mathcal{H}| \times \lceil \frac{\max(S)}{W} \rceil$ matrix where

$$a_{i,j} = \begin{cases} 1 & P(h_i) \cap S[(j-1)W + 1, jW] \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Thus, matrix A indicates for each heavy color and each block of size W in S which starts at a location following an integer multiple of W, whether the heavy color appears in the block or not.

For $\ell_0 \leq \ell \leq \ell_{max}$, let $B^{(\ell)} = \{b^{(\ell)}_{i,j}\}$ be a Boolean $\lceil \frac{\max(S)}{W} \rceil \times |\mathcal{H}|$ matrix where

$$b^{(\ell)}_{i,j} = \begin{cases} 1 & P(h_j) \cap S[(i-1)W + 1, iW + (1+\varepsilon)^\ell W] \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Matrix $B^{(\ell)}$ indicates for each block of size $W + (1+\varepsilon)^\ell W$ in S which starts at a location following an integer multiple of W, and for each heavy color, whether the heavy color appears in the block or not.

Let $E^{(\ell)} = A \cdot B^{(\ell)} = \{e^{(\ell)}_{i,j}\}$, where the product is a Boolean matrix product. Thus, $E^{(\ell)}$ roughly indicates for each pair of heavy colors whether their color distance is at most $W + (1+\varepsilon)^\ell W$.

In Section 3.1 we state and prove lemmas regarding the connections between the matrices $E^{(\ell)}$ and approximating color distances. In Section 3.2 we describe the algorithm for constructing E^*, and prove its correctness based on the lemmas of Section 3.1.

3.1 Properties of The $E^{(\ell)}$ Matrices

The following lemma describes a connection between entries of 0 in E^ℓ and lower bounds on color distances.

▶ **Lemma 12.** *For $h_i, h_j \in \mathcal{H}$ and integer $\ell_0 \leq \ell \leq \ell_{max}$, if $e^{(\ell)}_{i,j} = 0$ and $e^{(\ell)}_{j,i} = 0$, then $\delta_{h_i,h_j} > (1+\varepsilon)^\ell W$.*

Proof. Let $p \in P(h_i)$ and $p' \in P(h_j)$ be chosen such that $\delta_{h_i,h_j} = d(p, p')$. Assume towards a contradiction that $\delta_{h_i,h_j} \leq (1+\varepsilon)^\ell W$. We focus on the case of $p' \geq p$ and so $\delta_{h_i,h_j} = p' - p$; the proof for the case $p > p'$ is symmetrical. Thus, there exists an integer $1 \leq k \leq \lceil \frac{\max(S)}{W} \rceil$ such that $p \in S[(k-1)W + 1, kW]$ and so $a_{i,k} = 1$. Therefore,

$$p' = p + \delta_{h_i,h_j} \leq p + (1+\varepsilon)^\ell W \leq kW + (1+\varepsilon)^\ell W.$$

We conclude that $p' \in S[(k-1)W + 1, kW + (1+\varepsilon)^\ell W]$, implying that $b^{(\ell)}_{k,j} = 1$, and so $e^{(\ell)}_{i,j} = 1$, which is a contradiction. ◀

The following lemma shows a connection between colors with distance greater than $(\frac{1+2\varepsilon}{\varepsilon})W$ and entries in $E^{(\ell_0)}$ whose value is 0.

▶ **Lemma 13.** *For $h_i, h_j \in \mathcal{H}$, if $\delta_{h_i,h_j} > (\frac{1+2\varepsilon}{\varepsilon})W$ then $e^{(\ell_0)}_{i,j} = 0$ and $e^{(\ell_0)}_{j,i} = 0$.*

Proof. Assume by contradiction that $e^{(\ell_0)}_{i,j} = 1$ or $e^{(\ell_0)}_{j,i} = 1$. We focus on the case of $e^{(\ell_0)}_{i,j} = 1$; the proof for the case $e^{(\ell_0)}_{j,i} = 1$ is symmetrical. Since $e^{(\ell_0)}_{i,j} = 1$, there exists an integer $1 \leq k \leq \lceil \frac{\max(S)}{W} \rceil$ such that $a_{i,k} = 1$ and $b^{(\ell_0)}_{k,j} = 1$. Therefore, there exist points $p \in P(h_i)$ and $p' \in P(h_j)$ such that $p \in S[(k-1)W+1, kW]$ and $p' \in S[(k-1)W+1, kW+(1+\varepsilon)^{(\ell_0)}]$. Thus,

$$\begin{aligned}
\delta_{h_i,h_j} &= \min_{\hat{p}\in P(c), \tilde{p}\in P(c')} \{d(\hat{p},\tilde{p})\} \leq d(p,p') = \max\left(p-p', p'-p\right) \\
&= \max\left(W-1, (1+\varepsilon)^{(\ell_0)}W + W - 1\right) = (1+\varepsilon)^{(\ell_0)}W + W - 1 \\
&= (1+\varepsilon)^{\lfloor \log_{1+\varepsilon}(\frac{1}{\varepsilon})\rfloor+1}W + W - 1 \leq ((1+\varepsilon)^{\log_{1+\varepsilon}(\frac{1}{\varepsilon})+1} + 1)W - 1 \\
&< (\frac{1+2\varepsilon}{\varepsilon})W.
\end{aligned}$$

◀

The following lemma describes a connection between entries of 1 in E^ℓ and upper bounds on color distances.

▶ **Lemma 14.** *For $h_i, h_j \in \mathcal{H}$ and integer $\ell_0 \leq \ell \leq \ell_{\max}$, if $e^{(\ell)}_{i,j} = 1$, then $\delta_{h_i,h_j} < (1+\varepsilon)^{\ell+1}W$.*

Proof. If $e^{(\ell)}_{i,j} = 1$, then there exists an integer $1 \leq k \leq \lceil \frac{\max(S)}{W} \rceil$ such that $a_{i,k} = 1$ and $b^{(\ell)}_{k,j} = 1$. Hence, there exist $p \in P(h_i)$ and $p' \in P(h_j)$ such that $p \in S[(k-1)W+1, kW]$ and $p' \in S[(k-1)W+1, kW+(1+\varepsilon)^{\ell}W]$. Thus,

$$\begin{aligned}
\delta_{h_i,h_j} &\leq \left|p-p'\right| = \max(p-p', p'-p) \leq \max\left(W-1, (1+\varepsilon)^{\ell}W + W - 1\right) \\
&= (1+\varepsilon)^{\ell}W + W - 1 < ((1+\varepsilon)^{\ell} + 1)W.
\end{aligned}$$

Notice that $\lfloor \log_{1+\varepsilon}(\frac{1}{\varepsilon})\rfloor + 1 > \log_{1+\varepsilon}(\frac{1}{\varepsilon})$. Finally, since $\lfloor \log_{1+\varepsilon}(\frac{1}{\varepsilon})\rfloor + 1 = \ell_0 \leq \ell$, we have

$$(1+\varepsilon)^{\ell} \geq (1+\varepsilon)^{\lfloor \log_{1+\varepsilon}(\frac{1}{\varepsilon})\rfloor+1} > (1+\varepsilon)^{\log_{1+\varepsilon}(\frac{1}{\varepsilon})} = \frac{1}{\varepsilon},$$

and so $1 < \varepsilon(1+\varepsilon)^{\ell}$. Thus,

$$((1+\varepsilon)^{\ell} + 1)W < ((1+\varepsilon)^{\ell} + \varepsilon(1+\varepsilon)^{\ell})W = ((1+\varepsilon)^{\ell+1})W,$$

completing the proof. ◀

The following lemma shows that $e^{(\ell)}_{i,j}$ is weakly monotone increasing with respect to ℓ.

▶ **Lemma 15.** *For $h_i, h_j \in \mathcal{H}$, and $\ell_0 \leq \ell \leq \ell_{max}$, if $e^{(\ell)}_{i,j} = 1$, then for any $\ell < \ell' \leq \ell_{max}$ we have $e^{(\ell')}_{i,j} = 1$.*

Proof. If $e^{(\ell)}_{i,j} = 1$, then there exists an integer $1 \leq k \leq \lceil \frac{\max(S)}{W} \rceil$ such that $a_{i,k} = 1$ and $b^{(\ell)}_{k,j} = 1$. Thus, there exists $p \in P(h_i)$ such that $p \in S[(k-1)W+1, kW+(1+\varepsilon)^{\ell}W]$. Moreover, since $\ell < \ell'$, we have $S[(k-1)W+1, kW+(1+\varepsilon)^{\ell}W] \subseteq S[(k-1)W+1, kW+(1+\varepsilon)^{\ell'}W]$, and so $b^{(\ell')}_{k,j} = 1$. Finally, since $a_{i,k} = 1$, we conclude that $e^{(\ell')}_{i,j} = 1$. ◀

The following lemma shows that in $E^{(\ell_{max})}$, for each pair of heavy colors, there exists at least one corresponding 1 entry.

▶ **Lemma 16.** *For $h_i, h_j \in \mathcal{H}$, either $e_{i,j}^{(\ell_{max})} = 1$ or $e_{j,i}^{(\ell_{max})} = 1$.*

Proof. Assume by contradiction that $e_{i,j}^{(\ell_{max})} = 0$ and $e_{j,i}^{(\ell_{max})} = 0$. Let $p \in P(h_i)$ and $p' \in P(h_j)$ be chosen such that $\delta_{h_i,h_j} = d(p,p')$. Hence by Lemma 12, we have $\delta_{h_i,h_j} > (1+\varepsilon)^{\ell_{max}} W = (1+\varepsilon)^{\lceil \log_{1+\varepsilon} \frac{\max(S)}{W} \rceil} W$. Notice that $\max(S) \geq \delta_{h_i,h_j}$. Therefore,

$$\max(S) \geq \delta_{h_i,h_j} > (1+\varepsilon)^{(\lceil \log_{1+\varepsilon} \frac{\max(S)}{W} \rceil)} W \geq (1+\varepsilon)^{\log_{1+\varepsilon} \frac{\max(S)}{W}} W \geq \max(S),$$

which is a contradiction. ◀

In the following lemma we show the main property of the matrices $E^{(\ell)}$ used in the algorithm of Section 3.2 when dealing with pairs of heavy colors with color distance greater than $\frac{1+2\varepsilon}{\varepsilon} W$.

▶ **Lemma 17.** *For $h_i, h_j \in \mathcal{H}$, if $\delta_{h_i,h_j} > \frac{1+2\varepsilon}{\varepsilon} W$, then there exists a unique integer $\ell_0 \leq \ell \leq \ell_{max} - 1$ such that $e_{i,j}^{(\ell)} = e_{j,i}^{(\ell)} = 0$ and either $e_{i,j}^{(\ell+1)} = 1$ or $e_{j,i}^{(\ell+1)} = 1$.*

Proof. By Lemma 16, at least one of $e_{i,j}^{(\ell_{max})}$ and $e_{j,i}^{(\ell_{max})}$ must be 1. So assume without loss of generality that $e_{i,j}^{(\ell_{max})} = 1$. Since $\delta_{h_i,h_j} > \frac{1+2\varepsilon}{\varepsilon} W$, by Lemma 13 $e_{i,j}^{(\ell_0)} = 0$. Thus, by Lemma 15, there must exist exactly one $\ell_0 \leq \ell < \ell_{max}$ such that $e_{i,j}^{(\ell)} = 0$ and $e_{i,j}^{(\ell+1)} = 1$.

To complete the proof we show uniqueness. If $e_{i,j}^{(\ell_{max})} = e_{j,i}^{(\ell_{max})} = 1$, then assume towards a contradiction that there exist two different integers $\ell' \neq \hat{\ell}$ such that $e_{i,j}^{(\ell')} = 0$, $e_{j,i}^{(\ell')} = 0$, $e_{i,j}^{(\ell'+1)} = 1$, $e_{i,j}^{(\hat{\ell})} = 0$, $e_{j,i}^{(\hat{\ell})} = 0$, and $e_{j,i}^{(\hat{\ell}+1)} = 1$. Without loss of generality, $\ell' < \hat{\ell}$, and so by Lemma 15, $e_{i,j}^{(\ell'+1)} = 1$ and $\ell' < \hat{\ell}$ implies $e_{i,j}^{(\hat{\ell})} = 1$ which is a contradiction.

Otherwise, if at least one of $e_{i,j}^{(\ell_{max})}$ or $e_{j,i}^{(\ell_{max})}$ is 0, then since we assumed without loss of generality that $e_{i,j}^{(\ell_{max})} = 1$, we have $e_{j,i}^{(\ell_{max})} = 0$, which together with Lemma 15 implies that for all $\ell_0 \leq \ell' < \ell_{max}$, $e_{j,i}^{(\ell')} = 0$. Thus, ℓ is unique. ◀

3.2 Algorithm for Computing E^*

In this section, we describe the algorithm for computing E^*, prove its correctness and analyze its time cost. Pseudocode for the algorithm is given in Algorithm 1. The algorithm begins by a brute-force exact computation of color distances for all heavy colors h_i and h_i where $\delta_{h_i,h_j} \leq \frac{1+2\varepsilon}{\varepsilon} W$. The brute-force computation costs $O((\frac{1+2\varepsilon}{\varepsilon})Wn) = O(\frac{Wn}{\varepsilon})$ time.

The rest of the algorithm deals with the case of $\delta_{h_i,h_j} > \frac{1+2\varepsilon}{\varepsilon} W$. First, the algorithm computes the matrices $E^{(\ell)}$ from matrices A and $B^{(\ell)}$. The time cost of computing all $E^{(\ell)}$ is dominated by the cost of $\log_{1+\varepsilon}(\max(S)) = \Theta(\frac{\log \max(S)}{\varepsilon})$ matrix multiplications. Each multiplication is between a matrix of size $\frac{n}{\tau} \times \frac{\max(S)}{W}$ and a matrix of size $\frac{\max(S)}{W} \times \frac{n}{\tau}$. It is folklore knowledge that the matrix multiplication of two matrices of size $x \times y$ and $y \times z$ can be computed in $O\left(\frac{x \cdot y \cdot z}{\min(x,y,z)^{3-\omega}}\right)$. Thus, the time cost of computing the matrix multiplications is $O\left(\frac{n^2 \cdot \max(S)}{\tau^2 W \cdot \min(\frac{n}{\tau}, \frac{\max(S)}{W})^{3-\omega}} \frac{\log \max(S)}{\varepsilon}\right)$ time.

Finally, the algorithm utilizes the matrices $E^{(\ell)}$ to complete entries in E^* which correspond to heavy pairs of colors that were not covered by the brute-force computation. To do so, for each such pair $(h_i, h_j) \in \mathcal{H} \times \mathcal{H}$, the algorithm scans all $e_{i,j}^{(\ell)}$ to find the unique ℓ from

Algorithm 1 CONSTRUCT$E^*(S, \mathcal{H}, W, \varepsilon)$.

```
E* ← ∞
ℓ0 ← ⌊log_{1+ε}(1/ε)⌋ + 1; ℓmax ← ⌈log_{1+ε}(max(S))⌉
for p ∈ S and p̂ ∈ S[p, p + ((1+2ε)/ε)W] do
    if C_p = h_i and C_p̂ = h_i and e*_{i,j} > p̂ − p then
        e*_{i,j} ← p̂ − p; e*_{j,i} ← p̂ − p
    end if
end for
Construct matrix A
for ℓ ← ℓ0 to ℓmax do
    Construct matrix B^(ℓ)
    E^(ℓ) ← A · B^(ℓ)
end for
for ℓ ← ℓ0 to ℓmax do
    for (h_i, h_j) ∈ H × H do
        if e^(ℓ)_{i,j} = e^(ℓ)_{j,i} = 0 and (e^(ℓ+1)_{i,j} = 1 or e^(ℓ+1)_{j,i} = 1) then
            e*_{i,j} ← (1 + ε)^(ℓ+2) W
        end if
    end for
end for
```

Lemma 17 such that $e^{(\ell)}_{i,j} = e^{(\ell)}_{j,i} = 0$ and either $e^{(\ell+1)}_{i,j} = 1$ or $e^{(\ell+1)}_{j,i} = 1$, and sets $e^*_{i,j}$ to be $(1+\varepsilon)^{\ell+2}W$. Computing the entries in E^* for all such pairs costs $O(|\mathcal{H}|^2 \frac{\log \max(s)}{\varepsilon})) = O((\frac{n}{\tau})^2 \frac{\log \max(s)}{\varepsilon}))$, which is dominated by the cost of the matrix multiplications performed during the computation of all of the $E^{(\ell)}$ matrices.

▶ **Lemma 18.** *There exists an algorithm that computes E^* in $\tilde{O}\left(\frac{Wn}{\varepsilon} + \frac{n^2 \cdot \max(S)}{\tau^2 W \cdot \min(\frac{n}{\tau}, \frac{\max(S)}{W})^{3-\omega}}\right)$ time, such that for $h_i, h_j \in \mathcal{H}$ we have $\delta_{h_i,h_j} \leq e^*_{i,j} \leq (1+\varepsilon)\delta_{h_i,h_j}$.*

Proof. The runtime of the algorithm follows from the discussion above.

If $\delta_{h_i,h_j} \leq \frac{1+2\varepsilon}{\varepsilon}W$, then there exist $p \in P(h_1)$ and $\hat{p} \in P(h_i)$ such that $\delta_{h_i,h_j} = |\hat{p} - p| \leq \frac{1+2\varepsilon}{\varepsilon}W$, and so, after the brute-force computation, we have $e^*_{i,j} = \delta_{h_i,h_j}$.

Otherwise, $\delta_{h_i,h_j} > \frac{1+2\varepsilon}{\varepsilon}W$, and so the algorithm sets $e^*_{i,j}$ to be $(1+\varepsilon)^{\ell+2}W$, where ℓ is the unique integer from Lemma 17, and in particular, $e^{(\ell)}_{i,j} = e^{(\ell)}_{j,i} = 0$ and either $e^{(\ell+1)}_{i,j} = 1$ or $e^{(\ell+1)}_{j,i} = 1$. By Lemmas 14 and 12, $\delta_{h_i,h_j} < (1+\varepsilon)^{\ell+2}W < (1+\varepsilon)^2\delta_{h_i,h_j}$.

Finally, to obtain a $(1+\varepsilon)$ approximation one can run the algorithm with approximation parameter $\varepsilon' = \frac{\varepsilon}{3}$, which does not affect the asymptotic time complexity. ◀

4 Proof of Theorem 5

In this section, we analyze the combination of the construction algorithm for E^* given in Section 3.2 together with the generic ACDO algorithm described in Section 2.1. However, we focus on the type of instances of ACDO which are used in Section 5 for solving the approximate snippets problem. In particular, our metric is defined by locations in an array of size n, and so we have $\max(S) = n$. Thus, in our case,

$$T_{E^*}(\mathcal{H}) = O\left(\frac{Wn}{\varepsilon} + \frac{n^\omega \max(\tau, W)^{3-\omega}}{\tau^2 W} \cdot \frac{\log n}{\varepsilon}\right).$$

If $n^{\frac{\omega-1}{\omega+1}} \log^{\frac{1}{2}} n \leq \tau \leq n$, then we choose $W = (\frac{n}{\tau})^{\frac{\omega-1}{2}} \log^{\frac{1}{2}} n$. In such a case, we have $W = (\frac{n}{\tau})^{\frac{\omega-1}{2}} \log^{\frac{1}{2}} n \leq \left(\frac{n}{n^{\frac{\omega-1}{\omega+1}}}\right)^{\frac{\omega-1}{2}} \log^{\frac{1}{2}} n = n^{\frac{\omega-1}{\omega+1}} \log^{\frac{1}{2}} n \leq \tau$, and so

$$T_{E^*}(\mathcal{H}) = O\left(\frac{(\frac{n}{\tau})^{\frac{\omega-1}{2}} n}{\varepsilon} \log^{\frac{1}{2}} n + \frac{n^\omega \tau^{3-\omega}}{\tau^2 (\frac{n}{\tau})^{\frac{\omega-1}{2}} \cdot \log^{\frac{1}{2}} n} \cdot \frac{\log n}{\varepsilon}\right) = O\left(\frac{n^{\frac{\omega+1}{2}}}{\varepsilon \tau^{\frac{\omega-1}{2}}} \log^{\frac{1}{2}} n\right).$$

If $1 \leq \tau \leq n^{\frac{\omega-1}{\omega+1}} \log^{\frac{1}{2}} n$, then we choose $W = \frac{n}{\tau^{\frac{2}{\omega-1}}} \cdot \log^{\frac{1}{\omega-1}} n$. In such a case we have $W = \frac{n}{\tau^{\frac{2}{\omega-1}}} \log^{\frac{1}{\omega-1}} n \geq \frac{n}{n^{\frac{\omega-1}{\omega+1} \cdot \frac{2}{\omega-1}}} \log^{\frac{1}{\omega-1}} n = n^{\frac{\omega-1}{\omega+1}} \log^{\frac{1}{\omega-1}} n \geq \tau$, and so

$$\begin{aligned} T_{E^*}(\mathcal{H}) &= O\left(\frac{n^2}{\varepsilon \tau^{\frac{2}{\omega-1}}} \log^{\frac{1}{\omega-1}} n + \frac{n^\omega (\frac{n}{\tau^{\frac{2}{\omega-1}}})^{2-\omega} \cdot \log(n)^{\frac{2-\omega}{\omega-1}}}{\tau^2} \cdot \frac{\log n}{\varepsilon}\right) \\ &= O\left(\frac{n^2}{\varepsilon \tau^{\frac{2}{\omega-1}}} \log^{\frac{1}{\omega-1}} n\right). \end{aligned}$$

Thus, to summarize we have shown

$$T_{E^*}(\mathcal{H}) = \begin{cases} \tilde{O}\left(\frac{n^2}{\varepsilon \tau^{\frac{2}{\omega-1}}}\right) & \text{for } 1 \leq \tau \leq n^{\frac{\omega-1}{\omega+1}} \\ \tilde{O}\left(\frac{n^{\frac{\omega+1}{2}}}{\varepsilon \tau^{\frac{\omega-1}{2}}}\right) & \text{for } n^{\frac{\omega-1}{\omega+1}} \leq \tau \leq n. \end{cases}$$

Notice that in either case, $T_{E*}(\mathcal{H}) = \Omega(\max(n^2/\tau^2), n)$.

For the NNS data structure we use the van Emde Boas [25] data structure which for a set of m points from integer universe $\{1, 2, \ldots, u\}$ has a preprocessing cost of $O(m \cdot \log\log u)$ and query time $O(\log\log u)$. In our setting, $u = n$, and so $T_{p,NNS}(P(c)) = O(|P(c)| \log\log n)$ and $T_{q,NNS}(n) = O(\log\log n)$.

Thus, the construction time of the ACDO algorithm is

$$\begin{aligned} O(T_{E^*}(\mathcal{H}) + \sum_{c \in \mathcal{C}} T_{p,\mathrm{NNS}}(P(c))) &= O(T_{E^*}(\mathcal{H}) + \sum_{c \in \mathcal{C}} |P(c)| \log\log n) \\ &= O(T_{E^*}(\mathcal{H}) + n \log\log n) \\ &= \tilde{O}(T_{E^*}), \end{aligned}$$

and the query time is $O(\tau \cdot T_{q,\mathrm{NNS}}(n)) = O(\tau \log\log n) = \tilde{O}(\tau)$. To complete the proof of Theorem 5, we plug $\tau = \tilde{O}(n^b)$ into $n^a = T_{E^*}(\mathcal{H})$, to obtain the following tradeoff curve for a fixed ε.

$$\begin{cases} a + \frac{2}{\omega-1} b = 2 & \text{if } 0 \leq b \leq \frac{\omega-1}{\omega+1} \\ \frac{2}{\omega-1} a + b = \frac{\omega+1}{\omega-1} & \text{if } \frac{\omega-1}{\omega+1} \leq b \leq 1. \end{cases}$$

▶ Remark 19. One feature of Theorem 5 algorithm, which is used in the proof of Theorem 6, is that the query cost for ACDO queries when both C and C' are heavy is $O(1)$ time since all the algorithm does is looking up the answer in E^*.

5 Proof Sketch of Theorem 6

The proof of Theorem 6 is inspired by the exact MCDOCH algorithm of [17], combined with a new observation which allows to leverage Theorem 5. Due to space considerations, in this section, we only highlight the differences between our algorithm for AMCDOCH and the exact MCDOCH algorithm of [17]. The complete proof of Theorem 6 is given in the full version [12].

The relevant part of the algorithm of [17] creates an array A of size n which is a permutation of the points in S, and preprocesses A into an RNNS data structure. Then, the algorithm partitions A into blocks of size τ, for an integer parameter $1 \leq \tau \leq n$. Let $\mathcal{I} = \{I_1, I_2 \dots I_{\lceil \frac{n}{\tau} \rceil}\}$ denote the set of blocks.

Following [17], the algorithm constructs a matrix $B = \{b_{i,j}\}$ of size $\lceil \frac{n}{\tau} \rceil \times \lceil \frac{n}{\tau} \rceil$, such that $b_{i,j} = d(I_i, I_j)$. To construct B, the algorithm of [17] performs τ RNNS queries for the computation of each entry in B. Computing B turns out to be the dominating component in the preprocessing runtime of [17].

To obtain a faster preprocessing time for the approximate setting, instead of constructing B, our algorithm constructs an *approximate* matrix $\hat{B} = \{\hat{b}_{i,j}\}$ where $d(I_i, I_j) \leq \hat{b}_{i,j} \leq (1+\varepsilon)d(I_i, I_j)$. The construction of $\hat{B}$ utilizes the algorithm of Theorem 5 as follows. Define a new coloring set $\hat{\mathcal{C}} = \{\hat{c_1}, \hat{c_2}, \dots \hat{c_{|\mathcal{I}|}}\}$ over the points in S, such that for each $\hat{c_i} \in \hat{\mathcal{C}}$ we have $P(\hat{c_i}) = I_i$. Notice that for every $i \neq j$, we have $I_i \bigcap I_j = \emptyset$, and so using the colors of $\hat{\mathcal{C}}$ on S, every point in S has only one color. Thus, the algorithm uses the algorithm of Theorem 5 on S, but with the colors of $\hat{\mathcal{C}}$, and the query time designed to be $\tilde{O}(n^b) = \tilde{O}(\tau)$. Now, $\hat{b}_{i,j}$ is set to be the answer of the ACDO query on colors $\hat{c}_i$ and $\hat{c}_j$, and so $d(I_i, I_j) = \delta_{\hat{c}_i, \hat{c}_j} \leq \hat{b}_{i,j} \leq (1+\varepsilon)\delta_{\hat{c}_i, \hat{c}_j} = (1+\varepsilon)d(I_i, I_j)$.

In the last step of the preprocessing phase, the algorithm preprocesses $\hat{B}$ (instead of B in [13]) using a $2D$ Range Minimum Query (2DRMQ) data structure [3] so that given a rectangle in $\hat{B}$, defined by its corners, the algorithm returns in $O(1)$ time the smallest value entry in the rectangle.

Answering queries and correctness

The process for answering a query is the same as in [17], but using $\hat{B}$ instead of B. Thus, the correctness of the algorithm needs to be proven given the use of $\hat{B}$.

As shown in [17], for each $c \in \mathcal{C}$, $P(c)$ is exactly the points in some interval $A[x_c, y_c]$. The challenging part is answering an AMCDOCH query between C and C', when $P(c)$ and $P(c')$ are disjoint. Thus, assume without loss of generality that $x_c \leq y_c < x_{c'} \leq y_{c'}$. For the description here, we assume that x_c (and $x_{c'}$) is the first location of some block, and y_c (and $y_{c'}$) is the last location of some block. The more general cases are covered in the full version of this paper [12], with an additional execution of $\tilde{O}(\tau)$ RNNS queries.

The query algorithm executes a 2DRMQ data structure query on the rectangle in $\hat{B}$ defined by corners $(\frac{x_c - 1}{\tau} + 1, \frac{x_{c'} - 1}{\tau} + 1)$ and $(\frac{y_c}{\tau}, \frac{y_{c'}}{\tau})$. Let $\hat{b}_{i,j}$ be the answer returned by the 2DRMQ query. Notice that there exist $p \in A[x_c, y_c]$ and $p' \in A[x_{c'}, y_{c'}]$ such that $\delta_{c,c'} = d(p, p')$. Thus, there exist integers $\frac{x_c - 1}{\tau} + 1 \leq i^* \leq \frac{y_c}{\tau}$, $\frac{x_{c'} - 1}{\tau} + 1 \leq j^* \leq \frac{y_{c'}}{\tau}$ such that $p \in I_{i^*}$ and $p' \in I_{j^*}$. Thus, $\delta_{c,c'} = d(p, p') = d(I_{i^*}, I_{j^*})$, and so

$$\begin{aligned}\hat{b}_{i,j} &= \min_{\frac{x_c-1}{\tau}+1 \leq \hat{i} \leq \frac{y_c}{\tau}, \frac{x_{c'}-1}{\tau}+1 \leq \hat{j} \leq \frac{y_{c'}}{\tau}} \{\hat{b}_{\hat{i},\hat{j}}\} \leq \min_{\frac{x_c-1}{\tau}+1 \leq \hat{i} \leq \frac{y_c}{\tau}, \frac{x_{c'}-1}{\tau}+1 \leq \hat{j} \leq \frac{y_{c'}}{\tau}} \{(1+\varepsilon)d(I_{\hat{i}}, I_{\hat{j}})\} \\ &\leq (1+\varepsilon)d(I_{i^*}, I_{j^*}) = (1+\varepsilon)\delta_{c,c'}.\end{aligned}$$

Time Complexity

For the RNNS data structure we use the solution of [23] which on n points (which is the size of A) has preprocessing time $O(n \log n)$ and query time $O(\log^{\varepsilon} n) = \tilde{O}(1)$.

By Remark 19 and the fact that each block has size $\tau = \tilde{O}(n^b)$, each ACDO query used for constructing $\hat{B}$ costs $O(1)$ time. So, after preprocessing the ACDO data structure, constructing $\hat{B}$ costs $O((\frac{n}{\tau})^2)$ time. The preprocessing time of the 2DRMQ data structure of [3] when applied to $\hat{B}$ is $\tilde{O}((\frac{n}{\tau})^2)$. Thus, since $T_{E^*}(n) = \Omega(\max(n^2/\tau^2), n))$, the total time cost of the preprocessing phase, which is composed of constructing the ACDO and RNNS data structures on A, computing $\hat{B}$, and preprocessing a 2DRMQ data structure, is

$$\tilde{O}\left(T_{E^*}(n) + (n/\tau)^2 + n\right) = \tilde{O}\left(T_{E^*}(n)\right) = \begin{cases} \tilde{O}\left(\dfrac{n^2}{\varepsilon \tau^{\frac{2}{\omega-1}}}\right) & \text{for } 1 \leq \tau \leq n^{\frac{\omega-1}{\omega+1}} \\ \tilde{O}\left(\dfrac{n^{\frac{\omega+1}{2}}}{\varepsilon \tau^{\frac{\omega-1}{2}}}\right) & \text{for } n^{\frac{\omega-1}{\omega+1}} \leq \tau \leq n \end{cases}$$

The query process consists of $O(\tau)$ RNNS queries, and a single 2DRMQ query. Thus, the query time cost is $O(\tau)$. Finally, similar to the proof of Theorem 5, we obtained the following tradeoff curve for a fixed ε.

$$\begin{cases} a + \dfrac{2}{\omega-1} b = 2 & \text{if } 0 \leq b \leq \dfrac{\omega-1}{\omega+1} \\ \dfrac{2}{\omega-1} a + b = \dfrac{\omega+1}{\omega-1} & \text{if } \dfrac{\omega-1}{\omega+1} \leq b \leq 1. \end{cases}$$

6 Proof of Theorem 8: CLB for Exact MCDO on an Array

To prove Theorem 8, we design a reduction from $(min, +)$-matrix product on values in $[\hat{n}]$ to MCDO. To do so, consider an *unbalanced* version of $(min, +)$-matrix product where A is of size $\hat{n} \times \hat{m}$, B is of size $\hat{m} \times \hat{n}$, and all of the values are bounded by some positive integer M. It is straightforward to show that for any $x \geq 0$ such that $\hat{m} = \hat{n}^x$, solving unbalanced $(min, +)$-matrix product with any polynomial time improvement over $(\hat{n}^2\hat{m})^{1-o(1)}$ time is equivalent to solving $(min, +)$-matrix product on balanced matrices with $\hat{m} = \hat{n}$ in $\hat{n}^{3-\Omega(1)}$ time. Thus, an algorithm for unbalanced $(min, +)$-matrix product with values bounded by $M = [\min(\hat{n}, \hat{m})] = \hat{n}$ in time less than $(\hat{n}^2\hat{m})^{1-o(1)}$ would refute the Strong-APSP hypothesis.

The reduction

Our goal is to design a reduction from the unbalanced $(min, +)$-matrix product problem with values bounded by $\hat{n} \geq \min(\hat{n}, \hat{m})$ to MCDO. In preparation for the proof of Theorem 9, we first describe the reduction using positive integer parameter M as the upper bound on the values in the input matrices, and during the analysis we set $M = \hat{n}$.

For each $a_{i,j}$, the algorithm defines a point $a'_{i,j} = (M - a_{i,j}) + 9Mj$ with color i. For each $b_{i,j}$, the algorithm defines a point $b'_{i,j} = b_{i,j} + 3M + 9Mi$ with color $\hat{n} + j$. Thus, the largest point defined is at most $O(M \cdot \hat{m})$, and there are $2\hat{n}$ colors. Notice that there cannot be two entries in the same row (column) of A (B) that define the same point. However, it is possible that $a_{i,j} = a_{i',j}$ for $i \neq i'$, and so $a'_{i,j} = a'_{i',j}$, but the two points have different colors. A similar phenomenon can happen to points defined by entries of B which share the same row. Thus, we merge all occurrences of the same point into a single occurrence but colored

with all of the colors of the pre-merge occurrences. The set of points, denoted by S, contains at most $2\hat{n}\hat{m}$ multi-colored points in a metric defined by an array of size $O(M\hat{m})$, and so the algorithm uses the MCDO algorithm to preprocess S. Finally, for each entry $d_{i,j}$ in D, the algorithm sets $d_{i,j} \leftarrow \delta_{i,\hat{n}+j} - 2M$ by performing a query on the MCDO data structure.

Correctness

Suppose $d_{i,j} = a_{i,k^*} + b_{k^*,j}$ for some $k^* \in [\hat{m}]$. For every $k \in [\hat{m}]$, by definition we have $b'_{k,j} > a'_{i,k}$ and so $\left|a'_{i,k} - b'_{k,j}\right| = b'_{k,j} - a'_{i,k} = b_{k,j} + a_{i,k} + 2M$. Since a'_{i,k^*} is colored with color i and $b'_{k^*,j}$ is colored with color $\hat{n} + j$, we have

$$\delta_{i,\hat{n}+j} - 2M \leq \left|a'_{i,k^*} - b'_{k^*,j}\right| - 2M = a_{i,k^*} + b_{k^*,j} = d_{i,j}. \tag{1}$$

Let $k', k'' \in [\hat{m}]$. If $k' \neq k''$, then

$$\begin{aligned}\left|a'_{i,k'} - b'_{k'',j}\right| &= \max\left(a'_{i,k'} - b'_{k'',j}, b'_{k'',j} - a'_{i,k'}\right) \\ &= \max\left(9M(k' - k'') - 2M - a_{i,k'} - b_{k'',j}, b_{k'',j} + a_{i,k'} + 9M(k'' - k') + 2M\right) \\ &\geq 5M.\end{aligned}$$

However, if $k' = k''$ then $a_{i,k'} + b_{k',j} + 2M \leq 4M$. Since $4M < 5M$, we have

$$\delta_{i,\hat{n}+j} = \min_{p \in P(i), p' \in P(\hat{n}+j)} \left\{\left|p - p'\right|\right\} = \min_{1 \leq \hat{k} \leq \hat{n}} \left\{\left|a'_{i,\hat{k}} - b'_{\hat{k},j}\right|\right\}.$$

Thus, there exists some $\hat{k} \in [\hat{m}]$ such that $\delta_{i,\hat{n}+j} = \left|a'_{i,\hat{k}} - b'_{\hat{k},j}\right| = a_{i,\hat{k}} + b_{\hat{k},j} + 2M \geq d_{i,j} + 2M$, and combined with Equation 1 we have $d_{i,j} = \delta_{i,\hat{n}+j} - 2M$.

Lower bound

Recall that in our setting $M = \hat{n}$. The reduction preprocesses an MCDO data structure on $n \leq 2\hat{n}\hat{m}$ points in an array metric of size $O(M\hat{m}) = O(n)$, and then answers $\hat{n}^2$ MCDO queries. If $t_p(n)$ and $t_q(n)$ are preprocessing and query times, respectively, of the MCDO data structure, then the Strong-APSP conjecture implies that $t_p(\hat{n}\hat{m}) + \hat{n}^2 \cdot t_q(\hat{n}\hat{m}) = (\hat{n}^2\hat{m})^{1-o(1)}$.

Recall that our goal is to prove that $a+b \geq 2-o(1)$. Thus, assume towards a contradiction that $a + b = 2 - \Omega(1)$, which, by straightforward algebraic manipulation, implies that $\frac{2a}{a-b} = 1 + \frac{2}{a-b} - \Omega(1)$. Moreover, since the unbalanced $(min, +)$-matrix product is hard for any choice of $x \geq 0$, we can choose $x = \frac{2}{a-b} - 1$ and so $\hat{n}\hat{m} = \hat{n}^{1+x} = \hat{n}^{\frac{2}{a-b}}$. Notice that $2 + \frac{2b}{a-b} = \frac{2a}{a-b}$ Thus, we have $t_p(\hat{n}\hat{m}) + \hat{n}^2 \cdot t_q(\hat{n}\hat{m}) = O(\hat{n}^{\frac{2a}{a-b}} + \hat{n}^{2+\frac{2b}{a-b}}) = O(\hat{n}^{\frac{2a}{a-b}}) = O(\hat{n}^{1+\frac{2}{a-b}-\Omega(1)}) < (\hat{n}^2\hat{m})^{1-o(1)}$, which contradicts the Strong-APSP hypothesis.

7 Conclusions and Open Problems

We have shown the existence of FMM based algorithms for both ACDO and the $(1+\varepsilon)$-approximate snippets problem which, assuming $\omega = 2$, are essentially optimal. Moreover, we proved CLBs for exact version of CDO, implying that the exact versions of CDO and the snippets problem are strictly harder than their approximate versions.

We remark that one immediate and straightforward way to improve our algorithms if $\omega > 2$ is to apply fast rectangular matrix multiplication ([2]). However, we chose not to describe such improvements since they are mostly technical and do not add any additional

insight to the problems that we address. Moreover, we remark that it is straightforward to adapt our CDO algorithms to return the two points (one of each color) that define the distance in addition to the actual distance (see the full version [12]).

Our work leaves open the task of designing improved algorithms for more general metrics such as higher dimensional Euclidean space.

References

1 Mohammad Reza Abbasifard, Bijan Ghahremani, and Hassan Naderi. A survey on nearest neighbor search methods. *International Journal of Computer Applications*, 2014.

2 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.

3 Amihood Amir, Johannes Fischer, and Moshe Lewenstein. Two-dimensional range minimum queries. In *Combinatorial Pattern Matching: 18th Annual Symposium, CPM 2007, London, Canada, July 9-11, 2007. Proceedings 18*, pages 286–294. Springer, 2007. doi:10.1007/978-3-540-73437-6_29.

4 Alexandr Andoni. *Nearest neighbor search: The old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

5 Maxim Babenko, Pawel Gawrychowski, Tomasz Kociumaka, and Tatiana Starikovskaya. Wavelet trees meet suffix trees. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 572–591. SIAM, 2014.

6 Djamal Belazzougui and Simon J Puglisi. Range predecessor and lempel-ziv parsing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2053–2071. SIAM, 2016. doi:10.1137/1.9781611974331.CH143.

7 Timothy M Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Fredman's trick meets dominance product: Fine-grained complexity of unweighted apsp, 3sum counting, and more. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, 2023.

8 Shiri Chechik. Improved distance oracles and spanners for vertex-labeled graphs. In *Algorithms–ESA 2012: 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings 20*, pages 325–336. Springer, 2012. doi:10.1007/978-3-642-33090-2_29.

9 Maxime Crochemore, Costas S Iliopoulos, Marcin Kubica, M Sohel Rahman, German Tischler, and Tomasz Waleń. Improved algorithms for the range next value problem and applications. *Theoretical Computer Science*, 434:23–34, 2012. doi:10.1016/J.TCS.2012.02.015.

10 Jacob Evald, Viktor Fredslund-Hansen, and Christian Wulff-Nilsen. Near-optimal distance oracles for vertex-labeled planar graphs. In *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

11 Danny Hermelin, Avivit Levy, Oren Weimann, and Raphael Yuster. Distance oracles for vertex-labeled graphs. In *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II 38*, pages 490–501. Springer, 2011. doi:10.1007/978-3-642-22012-8_39.

12 Noam Horowicz and Tsvi Kopelowitz. Color distance oracles and snippets: Separation between exact and approximate solutions. *arXiv preprint*, 2025. arXiv:2507.04578.

13 Matti Karppa and Petteri Kaski. Probabilistic tensors and opportunistic boolean matrix multiplication. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 496–515. SIAM, 2019. doi:10.1137/1.9781611975482.31.

14 Orgad Keller, Tsvi Kopelowitz, Shir Landau Feibish, and Moshe Lewenstein. Generalized substring compression. *Theoretical Computer Science*, 525:42–54, 2014. doi:10.1016/J.TCS.2013.10.010.

15 Orgad Keller, Tsvi Kopelowitz, and Moshe Lewenstein. Range non-overlapping indexing and successive list indexing. In *Algorithms and Data Structures: 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007. Proceedings 10*, pages 625–636. Springer, 2007. doi:10.1007/978-3-540-73951-7_54.

16 Jon M Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997.

17 Tsvi Kopelowitz and Robert Krauthgamer. Color-distance oracles and snippets. In *27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CPM.2016.24.

18 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1272–1287. SIAM, 2016. doi:10.1137/1.9781611974331.CH89.

19 Tsvi Kopelowitz and Virginia Vassilevska Williams. Towards optimal set-disjointness and set-intersection data structures. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.74.

20 Robert Krauthgamer and James R Lee. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Citeseer, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982913.

21 Mingfei Li, Chu Chung Christopher Ma, and Li Ning. $(1+\varepsilon)$-distance oracles for vertex-labeled planar graphs. In *International Conference on Theory and Applications of Models of Computation*, pages 42–51. Springer, 2013. doi:10.1007/978-3-642-38236-9_5.

22 J Ian Munro, Yakov Nekrich, and Jeffrey S Vitter. Fast construction of wavelet trees. *Theoretical Computer Science*, 638:91–97, 2016. doi:10.1016/J.TCS.2015.11.011.

23 Yakov Nekrich and Gonzalo Navarro. Sorted range reporting. In *Algorithm Theory – SWAT 2012: 13th Scandinavian Symposium and Workshops, Helsinki, Finland, July 4-6, 2012. Proceedings 13*, pages 271–282. Springer, 2012. doi:10.1007/978-3-642-31155-0_24.

24 Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 605–614. IEEE, 1999. doi:10.1109/SFFCS.1999.814635.

25 Peter van Emde Boas. Preserving order in a forest in less than logarithmic time. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 75–84. IEEE, 1975. doi:10.1109/SFCS.1975.26.