Gap Edit Distance via Non-Adaptive Queries: Simple and Optimal

Elazar Goldenberg[1], Tomasz Kociumaka[*2], Robert Krauthgamer[†3], and Barna Saha[*4]

[1]The Academic College of Tel Aviv-Yaffo, Israel
elazargo@mta.ac.il
[2]Max Planck Institute for Informatics, Germany
tomasz.kociumaka@mpi-inf.mpg.de
[3]Weizmann Institute of Science, Israel
robert.krauthgamer@weizmann.ac.il
[4]University of California, San Diego, United States
barnas@ucsd.edu

Abstract

We study the problem of approximating edit distance in sublinear time. This is formalized as the (k, k^c)-Gap Edit Distance problem, where the input is a pair of strings X, Y and parameters $k, c > 1$, and the goal is to return YES if $\mathsf{ED}(X, Y) \le k$, NO if $\mathsf{ED}(X, Y) > k^c$, and an arbitrary answer when $k < \mathsf{ED}(X, Y) \le k^c$. Recent years have witnessed significant interest in designing sublinear-time algorithms for Gap Edit Distance.

In this work, we resolve the non-adaptive query complexity of Gap Edit Distance for the entire range of parameters, improving over a sequence of previous results. Specifically, we design a non-adaptive algorithm with query complexity $\tilde{\mathcal{O}}(n/k^{c-0.5})$, and we further prove that this bound is optimal up to polylogarithmic factors.

Our algorithm also achieves optimal time complexity $\tilde{\mathcal{O}}(n/k^{c-0.5})$ whenever $c \ge 1.5$. For $1 < c < 1.5$, the running time of our algorithm is $\tilde{\mathcal{O}}(n/k^{2c-2})$. In the restricted case of $k^c = \Omega(n)$, this matches a known result [Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, and Sami; STOC 2003], and in all other (nontrivial) cases, our running time is strictly better than all previous algorithms, including the adaptive ones. However, independent work of Bringmann, Cassis, Fischer, and Nakos [STOC 2022] provides an adaptive algorithm that bypasses the non-adaptive lower bound, but only for small enough k and c.

1 Introduction

The *edit distance* is a ubiquitous distance measure on strings. It finds applications in various fields including computational biology, pattern recognition, text processing, information retrieval, and many more. The edit distance between strings X and Y, denoted by $\mathsf{ED}(X, Y)$, is defined as the minimum number of character insertions, deletions, and substitutions needed to convert X into Y. A simple textbook dynamic programming computes edit distance in quadratic time. Moreover, under reasonable hardness assumptions, such as the Strong Exponential-Time Hypothesis, no truly subquadratic-time algorithm for this problem exists [ABW15, BK15, AHWW16, BI18].

[*]Partly supported by NSF CCF grants 1652303 and 1909046, and a HDR TRIPODS Phase II grant 2217058.
[†]Partly supported by ONR Award N00014-18-1-2364, the Israel Science Foundation grant #1086/18, the Weizmann Data Science Research Center, and a Minerva Foundation grant.

When dealing with enormous amounts of data, such as DNA strings, big data storage, etc., quadratic running time might be prohibitive, leading a quest for faster algorithms that find an approximate solution. A long line of research towards that goal [BEK+03, BJKK04, BES06, AKO10, AO12, CDG+20, BEG+21] recently culminated with an almost-linear-time approximation algorithm by Andoni and Nosatzki [AN20] that, for any desired $\varepsilon > 0$, runs in $O(n^{1+\varepsilon})$ time and achieves an approximation factor that depends only on ε, that is, a constant-factor approximation for any fixed $\varepsilon > 0$.

The growing interest in modern computational paradigms, like streaming and sketching (sublinear space), sampling and property testing (sublinear time), and massively parallel computation, sparked interest in *sublinear-time* algorithms. It started with a seminal work of Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, and Sami [BEK+03], and developed into an exciting sequence of results on approximating ED in sublinear time [AO12, GKS19, BCR20, KS20, BCFN22a]. (These are sometimes called estimation algorithms to emphasize that they approximate ED without necessarily constructing a witness alignment.)

A sublinear-time algorithm for ED cannot be expected to attain constant-factor approximation, since even in the case where the edit distance is $O(1)$, a linear fraction of the input strings must be queried. Hence, the aim here is to solve the promise problem (k, k^c)-Gap Edit Distance, which asks to return YES if $\mathsf{ED}(X, Y) \le k$, NO if $\mathsf{ED}(X, Y) > k^c$, and an arbitrary answer otherwise. The accuracy of the aforementioned results depends on gap "size" c and the gap "location" k; their performance is measured in terms of their query and time complexity, and also qualitatively whether they query the input strings adaptively (i.e., each query may depend on the results of earlier queries).

The first contribution [BEK+03] addressed the case $k^c = \Omega(n)$. Under this restriction, they obtained a sublinear-time algorithm that runs in $\tilde{\mathcal{O}}(k^2/n + \sqrt{k})$ time.[1] Moreover, they showed a query-complexity lower bound of $\Omega(\sqrt{k})$, rendering their result optimal for $c \ge 1.5$. Andoni and Onak [AO12] were the first to overcome the limitation that $k^c = \Omega(n)$. However, their query complexity $\hat{\mathcal{O}}(n^2/k^{2c-1})$ reduces to $\hat{\mathcal{O}}(k)$ when $k^c = \Omega(n)$, far above that of [BEK+03].[2] Interestingly, both these algorithms are non-adaptive.

In recent years, further progress has been achieved, mostly by exploiting adaptive queries, particularly by Goldenberg, Krauthgamer, and Saha [GKS19], and subsequently by Kociumaka and Saha [KS20], who improved over [AO12] when k is small. The running time $\tilde{\mathcal{O}}(n/k^{c-1} + k^3)$ of [GKS19] had an undesirable cubic dependency on k, which was improved to quadratic in [KS20] at the price of an extra $\tilde{\mathcal{O}}(k^{2.5-c}n^{0.5})$ term appearing for $c < 2$. Non-adaptive algorithms often tend to be simple, but they are generally less powerful. So far, the best results in the regime of small k came through carefully using adaptive queries [GKS19, KS20]. Hence, it seemed plausible that adaptivity would be crucial to improving beyond [AO12] for large k as well.

Technical Contributions In light of prior work, the following main questions remained open.

- Can we remove/reduce the polynomial dependency on k from [GKS19, KS20] without degrading the dependency on n?
- Is adaptivity needed to achieve complexity $\tilde{\mathcal{O}}(n/k^{c-1})$ for small k?
- Can we obtain tight query-complexity lower bounds?

In Section 3, we present a simple *non-adaptive* algorithm that removes the polynomial dependency on k entirely, thus answering the first two questions. The algorithm solves the (k, k^c)-Gap Edit Distance problem with time complexity $\hat{\mathcal{O}}(n/k^{c-1})$, as follows.

[1] The $\tilde{\mathcal{O}}(\cdot)$ notation hides factors polylogarithmic in n.

[2] The $\hat{\mathcal{O}}(\cdot)$ notation hides factors subpolynomial in n.

Time Complexity	Non-Adaptive	Restrictions	Reference
$\tilde{\mathcal{O}}\left(k^{0.5}\right) = \tilde{\mathcal{O}}\left(n/k^{c-0.5}\right)$	Yes	$k^c = \Omega(n)$, $c \geq 1.5$	[BEK+03]
$\tilde{\mathcal{O}}\left(k^2/n\right) = \tilde{\mathcal{O}}\left(n/k^{2c-2}\right)$	Yes	$k^c = \Omega(n)$, $c < 1.5$	[BEK+03]
$\hat{\mathcal{O}}(n^2/k^{2c-1})$	Yes		[AO12]
$\tilde{\mathcal{O}}(n/k^{c-1} + k^3)$	No		[GKS19]
$\tilde{\mathcal{O}}(n/k^{c-1.5} + k^{2.5-c})$	Yes	$c \geq 1.5$	[BCR20]
$\tilde{\mathcal{O}}(n/k^{c-1} + k^2 + \sqrt{n} \cdot k^{2.5-c})$	No		[KS20]
$\hat{\mathcal{O}}(n/k^c + n^{0.8} + k^4)$	No		[BCFN22a]
$\hat{\mathcal{O}}(n/k^{c-1})$	Yes		Corollary 3.5
$\tilde{\mathcal{O}}(n/k^{c-0.5})$	Yes	$c \geq 1.5$	Theorem 5.6
$\tilde{\mathcal{O}}(n/k^{2c-2})^*$	Yes	$c < 1.5$	Theorem 5.6

*Query complexity is $\tilde{\mathcal{O}}(n/k^{c-0.5})$, lower than the time complexity.

Table 1: Sublinear-time algorithms for (k, k^c)-Gap Edit Distance.

Theorem 1.1 (Simplified version of Corollary 3.5)**.** *For every constant $c > 1$, there is a non-adaptive randomized algorithm that solves (k, k^c)-Gap Edit Distance in time $\hat{\mathcal{O}}(n/k^{c-1})$.*

This result already improves upon all prior results [AO12, GKS19, KS20, BCR20], except for the earliest algorithm of [BEK+03] that applies only for $k^c = \Omega(n)$ (see below for a comparison with independent work [BCFN22a]). The algorithm abandons the recent approach of [GKS19, KS20] of scanning the two input strings and tracking their periodicity structure using adaptive queries. Instead, our baseline is Andoni and Onak's algorithm [AO12], which samples a few blocks of predetermined length from each string, and computes only a "local" alignment (between the ith block of X and the ith block of Y). In [AO12], the block length is optimized according to the gap parameters k and c. Somewhat surprisingly, just by sampling blocks of different lengths and using all of them simultaneously (instead of choosing a single block length), we achieve a significantly better result. The details, including a technical overview, appear in Section 3.

Our main result still uses non-adaptive sampling and achieves a significant improvement for the entire range of c, and in particular generalizes or improves upon all previous bounds. By building on the above simple algorithm, we first improve the query complexity (in Section 4) and then also the time complexity (in Section 5), both in the polynomial dependency on k and in the $n^{o(1)}$-factor.

Theorem 1.2 (Simplified version[3] of Theorem 5.6)**.** *For every constant $c > 1$, there is a non-adaptive randomized algorithm that solves (k, k^c)-Gap Edit Distance using $\tilde{\mathcal{O}}(n/k^{c-0.5})$ queries and $\tilde{\mathcal{O}}(n/k^{\min(c-0.5, 2c-2)})$ time.*

Our final contribution is a new lower bound for non-adaptive algorithms that applies for all values of k and c (see Section 6). It extends a previous lower bound of [BEK+03], which handles only the very special case $k^c = \Omega(n)$.

Theorem 1.3 (Simplified version of Theorem 6.2)**.** *For every constant $c > 1$ and parameters $n \geq k \geq 1$, every non-adaptive algorithm solving the (k, k^c)-Gap Edit Distance problem has expected query complexity $\Omega(n/k^{c-0.5})$.*

[3] It suffices to use Theorem 5.6 with any constant $h > \frac{1}{1-c}$, and resort to an exact algorithm if $k = \tilde{\mathcal{O}}(1)$. We remark that Theorem 5.6 additionally has some limited applicability to $c = 1 + o(1)$ and can solve $(k, k \cdot 2^{\mathcal{O}(\sqrt{\log n})})$-Gap Edit Distance using $\hat{\mathcal{O}}(n/\sqrt{k})$ queries.

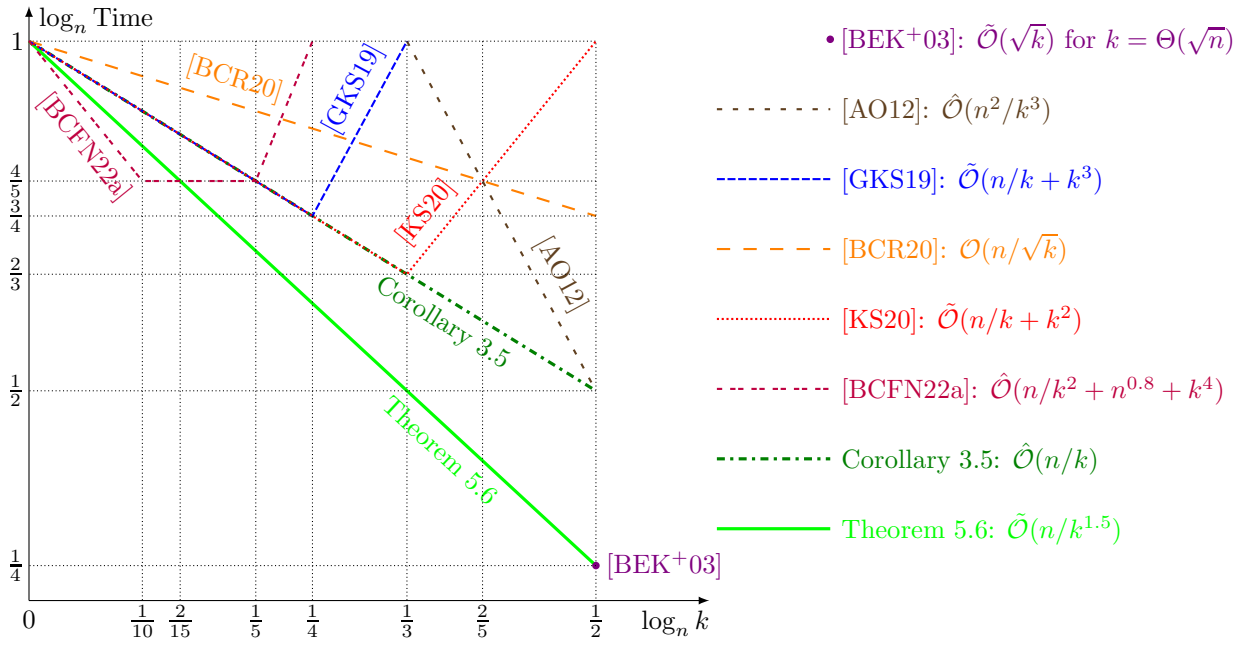

Figure 1: The running times of algorithms for (k, k^2)-Gap Edit Distance.

Altogether, we obtain optimal non-adaptive query complexity for all $c > 1$, and furthermore optimal time complexity for a large regime (all $c \geq 1.5$). In particular, we achieve optimal query and time complexity for the quadratic gap edit distance problem ($c = 2$), which was the focus of all recent work [GKS19, KS20, BCR20]. When $c < 1.5$, we match the time bound of [BEK+03] and further remove their restriction that $k^c = \Omega(n)$.

Table 1 lists all the known algorithmic bounds, including our, previous, and independent results. It is instructive to compare them against $\Omega(n/k^{c-0.5})$, our (tight) lower bound for non-adaptive algorithms (Theorem 6.2). Figure 1 plots these bounds for a quadratic gap ($c = 2$). One can see that our main result, Theorem 5.6, improves over all previous results for the entire range of k, although independent work [BCFN22a] provides a further improvement for small k by using adaptive sampling and thus bypassing our lower bound.

Open Questions Our results completely resolve the non-adaptive query complexity of Gap Edit Distance. The lower bound of [BEK+03] applies to adaptive queries as well, and we match this lower bound using a non-adaptive algorithm. Hence, adaptivity cannot help at the extreme regime of $k^c = \Omega(n)$. The question remains though whether adaptivity is useful to improve the complexity further in the intermediate regime, where the $\hat{\mathcal{O}}(k^4)$ term in the running time of [BCFN22a] makes that solution slower than ours.

Another open problem is to improve the time complexity (ideally to match the query-complexity lower bound) for $1 < c < 1.5$ or to strengthen the lower bound for time complexity, showing a separation between time and query complexity (as in [AdlVKK03] for the max-cut problem, for example).

Related Work Sublinear-time algorithms were studied for several related string problems, including the Ulam metric [AN10, NSS17], longest increasing subsequence (LIS) [SS17, MS21],

and shift finding [AHIK13]. There are also sublinear-space streaming algorithms for edit distance [GJKK07, GG10, SS13, EJ15, CGK16, BZ16, CFH+21]. Several algorithms for edit distance leverage preprocessing (of one or both strings independently) to perform further computations in sublinear-time [AKO10, GRS20, BCR20, BCFN22b].

Non-Adaptive Sampling Our algorithms only require a non-adaptive sampling. While these might bring inferior performance (running time or query complexity) compared to algorithms using adaptive sampling, as indeed obtained independently of our work in [BCFN22a] for small values of k, numerous applications can gain from — or even require — non-adaptive sampling. Consider for example a distributed setting where the input XY is partitioned into $p \geq 2$ substrings, held by distinct players that communicate in the blackboard model (equivalent to a broadcast channel). One particular case of interest is two players, one holding X and the other holding Y. Every sampling algorithm $\mathcal{A}$ has an obvious distributed implementation whose communication complexity is precisely the query complexity of $\mathcal{A}$, but clearly a non-adaptive $\mathcal{A}$ requires only one round of communication (assuming shared randomness). For another example, consider $t \geq 3$ input strings $X^{(1)}, \ldots, X^{(t)}$ and a goal of estimating the edit distance between every pair of strings. When implementing a non-adaptive sampling algorithm $\mathcal{A}$, it suffices to sample each string $X^{(i)}$ only once, and use the sample across all the $t-1$ executions involving the string $X^{(i)}$, thereby running $O(t^2)$ executions of $\mathcal{A}$ using only $O(t)$ sets of samples, reducing communication by factor t compared to using adaptive sampling.

2 Preliminaries

Fact 2.1. *Let $X, Y \in \Sigma^n$. For every $i, j \in [0\,..\,n]$ with $i \leq j$, we have $\mathsf{ED}(X[i\,..\,j), Y[i\,..\,j)) \leq \mathsf{ED}(X, Y)$.*

Fact 2.2. *For all strings $X_1, X_2, Y_1, Y_2 \in \Sigma^*$, we have $\mathsf{ED}(X_1X_2, Y_1Y_2) \leq \mathsf{ED}(X_1, Y_1) + \mathsf{ED}(X_2, Y_2)$.*

Problem 2.3 ((β, α)-Gap Edit Distance)**.** *Given strings $X, Y \in \Sigma^n$ and integers $\alpha \geq \beta \geq 0$, return YES if $\mathsf{ED}(X, Y) \leq \beta$, NO if $\mathsf{ED}(X, Y) > \alpha$, and an arbitrary answer otherwise.*

Theorem 2.4 (Landau and Vishkin [LV88])**.** *There exists a deterministic algorithm that solves any instance of the (β, α)-Gap Edit Distance problem (with arbitrary $\alpha \geq \beta \geq 0$) in $\mathcal{O}(n + \beta^2)$ time.*

Fact 2.5 (see e.g. [HIM12, KOR00])**.** *There exists a randomized algorithm that solves any instance of the $(0, \alpha)$-Gap Edit Distance problem in $\mathcal{O}(\frac{n}{1+\alpha})$ time with success probability at least $\frac{2}{3}$.*

Theorem 2.6 (Andoni and Nosatzki [AN20])**.** *There exist decreasing functions $f_{AN}, g_{AN} : \mathbb{R}_+ \to \mathbb{R}_{\geq 1}$ and a randomized algorithm $\mathcal{A}$ that, given $X, Y \in \Sigma^n$ and $\varepsilon \in \mathbb{R}_+$, runs in $\mathcal{O}(g_{AN}(\varepsilon) n^{1+\varepsilon})$ time and returns a value $\mathcal{A}(X, Y, \varepsilon)$ satisfying*

$$\mathbb{P}[\mathsf{ED}(X, Y) \leq \mathcal{A}(X, Y, \varepsilon) \leq f_{AN}(\varepsilon)\mathsf{ED}(X, Y)] \geq \tfrac{2}{3}.$$

Below, f_{AN} and g_{AN} denote the functions of Theorem 2.6.

Corollary 2.7. *There exists a randomized algorithm that, given $\varepsilon, \delta \in \mathbb{R}_+$ and an instance of (β, α)-Gap Edit Distance satisfying $\alpha \geq \lfloor f_{AN}(\varepsilon)\beta \rfloor$, solves the instance in time $\mathcal{O}(g_{AN}(\varepsilon) n^{1+\varepsilon} \log \frac{1}{\delta})$ with error probability at most δ.*

Proof. Consider running the algorithm of Theorem 2.6. If $\mathsf{ED}(X,Y) \le \beta$, then the answer is at most $f_{\text{AN}}(\varepsilon)\beta < \alpha+1$ with probability at least $\frac{2}{3}$. If $\mathsf{ED}(X,Y) > \alpha$, then the answer is at least $\alpha+1$ with probability at least $\frac{2}{3}$. Hence, comparing the answer against $\alpha+1$ solves the gap problem in time $\mathcal{O}(g_{\text{AN}}(\varepsilon)n^{1+\varepsilon})$ with success probability at least $\frac{2}{3}$. The success probability can be amplified to at least $1-\delta$ by running the algorithm $\Theta(\log\frac{1}{\delta})$ times with independent randomness and returning the dominant answer. □

3 Simple Algorithm

The main result of this section is a randomized algorithm for the (β,α)-Gap Edit Distance problem that, under mild technical conditions, makes $\hat{\mathcal{O}}(\frac{\beta}{\alpha}\cdot n)$ non-adaptive queries to the two input strings. The precise time bound depends on the functions f_{AN} and g_{AN} from Theorem 2.6 (see Corollary 3.4 for the formal statement; here, we assumed fixed $\varepsilon,\delta > 0$). Our algorithm is essentially a reduction (presented in Section 3.3) to the same gap problem but with smaller gap parameters, building upon an earlier reduction of Andoni and Onak [AO12].[4] In a nutshell, these reductions partition the two input strings into blocks and call an oracle that solves gap problems on a few randomly chosen block pairs. The key difference from [AO12] is that their reduction uses one block length, while ours essentially uses all feasible block lengths.

We start below with an overview of both reductions (Section 3.1), followed by a quick proof of their reduction (Section 3.2), which makes it easier to read our reduction (Section 3.3) and also to compare the two. To obtain our final result, we only need to implement the oracle, and we simply plug in the state-of-the-art almost-linear-time algorithm of [AN20] into our reduction (Section 3.4).

3.1 Overview

To simplify this overview, we shall assume an algorithm that approximates the edit distance within factor $f = \hat{\mathcal{O}}(1)$ in time $\hat{\mathcal{O}}(n)$, and we shall refer to it as an oracle that solves (β,α)-Gap Edit Distance in almost-linear time whenever $\alpha \ge f\beta$. Such algorithms were devised in [AO12, AN20], and their precise bounds are not important for this overview.

We first sketch the algorithm (reduction) of Andoni and Onak [AO12]. It partitions the two input strings X,Y into $m := \frac{n}{b}$ blocks of length b that will be determined later, denoting their respective ith blocks by X_i and Y_i for $i \in [0\,..\,m)$. If the algorithm determines that $\mathsf{ED}(X_i,Y_i) > \beta$ for some $i \in [0\,..\,m)$, then, by Fact 2.1, also $\mathsf{ED}(X,Y) > \beta$, and the algorithm is safe to return NO. The algorithm's strategy is just to search for such a "NO witness"; for this, it samples several indices i, calls the oracle to solve $(\beta, f\beta)$-Gap Edit Distance on the corresponding pairs (X_i,Y_i), and returns NO if and only if at least one of the oracle calls returned NO. This algorithm is clearly correct whenever $\mathsf{ED}(X,Y) \le \beta$, so we only need to consider $\mathsf{ED}(X,Y) > \alpha$. In that case, by Fact 2.2, $\mathsf{ED}(X_i,Y_i) > \frac{\alpha}{m}$ holds for an average i (a crude intuition is that an average block "contains" many edit operations). For this sketch, let us consider only the two extreme scenarios. In one scenario, $\mathsf{ED}(X_i,Y_i)$ has the same value for all i; if $\frac{\alpha}{m} \ge f\beta$, then, no matter which block our algorithm samples, the oracle will return NO on it, and our algorithm will also return NO. We will thus constrain our choice of m to satisfy $\frac{\alpha}{m} \ge f\beta$. In the other extreme scenario, $\mathsf{ED}(X_i,Y_i)$ has a large value for a few indices i and a small value, say zero for simplicity, for all other indices. These large values are bounded by $\mathsf{ED}(X_i,Y_i) \le b$; hence, the first group must contain at least $\frac{\alpha}{b}$ indices i (again by Fact 2.2). To have a good chance of sampling at least one of these indices, our algorithm should sample each i with probability (at least) $\rho = \Omega(\frac{b}{\alpha})$. To optimize algorithm's query

[4] The reduction in [AO12] is presented as an application of their almost-linear-time approximation algorithm.

complexity, we set the parameters to minimize the sampling rate ρ, i.e., minimize b or, equivalently, maximize m. Due to the constraint from above, the optimal choice is thus $b = \frac{n}{m} = \frac{n \cdot f\beta}{\alpha}$. The query complexity of this algorithm is $\mathcal{O}(\rho n) = \mathcal{O}(\frac{b}{\alpha} \cdot n) = \hat{\mathcal{O}}(\frac{n^2 \cdot \beta}{\alpha^2})$, and the running time is almost-linear in the query complexity, and thus bounded similarly.

The true limitation of this approach is that it uses a single block length b. It is somewhat hidden because we compare $\mathsf{ED}(X_i, Y_i)$ only against the natural threshold β (and $f\beta$, but the factor f is almost negligible here), which leads to an optimal choice of b. One idea is to use a different block length b, or even multiple lengths. But should it be larger or smaller? And what advantage can we gain from it?

What turns out to work well is a multi-level approach, which partitions the input strings into blocks of different lengths (all powers of 2) and samples blocks from all the levels at the same rate ρ. The query complexity is $O(\rho n)$ for each level, and there are only $O(\log n)$ levels, but we can now use sampling rate $\rho = \Theta(\frac{f\beta}{\alpha})$, which is significantly lower than $\Theta(\frac{nf\beta}{\alpha^2})$ needed in [AO12]. To understand this improvement in the sampling rate, recall the two extreme scenarios mentioned above. In the first scenario, where edits are spread evenly among the length-b blocks for $b = \Omega(\frac{n \cdot f\beta}{\alpha})$, we already argued that querying any length-b block suffices to detect a "NO witness". In the second scenario, consider shorter blocks of length $\mathcal{O}(f\beta)$, and suppose that each $\mathsf{ED}(X_i, Y_i)$ is either zero or exceeds $f\beta$. The number of indices i in the latter group must be at least $\frac{\alpha}{\mathcal{O}(f\beta)}$ (again by Fact 2.2), and they are all "NO witnesses". To have a good chance of sampling at least one of them, it suffices to use rate $\rho = \Theta(\frac{f\beta}{\alpha})$.

Our proof considers all levels and, for each position $j \in [1\,..\,n]$, identifies a "suitable" level based on the distribution of errors in the proximity of j. This is achieved by decomposing $[1\,..\,n]$ into blocks of varying sizes, so that the block covering position j reveals the level responsible for j. Perhaps surprisingly, the analysis is thus adaptive even though the algorithm only makes non-adaptive queries!

3.2 The Reduction of Andoni and Onak

We recall a sublinear-time algorithm of Andoni and Onak [AO12, Section 4] that makes calls to an oracle solving the same gap problem but with a smaller gap. They implement this oracle using their main result, which is an almost-linear-time approximation algorithm. We review their proof to illustrate how our algorithm and analysis are different.

Theorem 3.1. *There exists a randomized reduction that, given a parameter $\phi \in \mathbb{Z}_+$ and an instance of (β, α)-Gap Edit Distance satisfying $\frac{1}{3}\alpha \geq \phi \geq \beta \geq 1$, solves the instance using $\mathcal{O}(\frac{n}{\alpha})$ non-adaptive calls to an oracle for (β, ϕ)-Gap Edit Distance involving substrings of total length $\mathcal{O}(\frac{\phi n^2}{\alpha^2})$. The reduction takes $\mathcal{O}(\frac{n}{\alpha})$ time, does not access the input strings, and errs with probability at most $\frac{1}{e}$.*

Proof. Let us partition the input strings X, Y into $m := \lceil \frac{n}{b} \rceil$ *blocks* of length $b := \lceil \frac{3\phi n}{\alpha} \rceil$ (the last blocks might be shorter), denoting the ith blocks by X_i and Y_i for $i \in [0\,..\,m)$. For a *sampling rate* $\rho := \frac{b^2}{\phi n}$, the algorithm performs $\lceil m\rho \rceil$ iterations. In each iteration, the algorithm chooses $i \in [0\,..\,m)$ uniformly at random and makes an oracle call to solve an instance (X_i, Y_i) of (β, ϕ)-Gap Edit Distance. The algorithm returns YES if and only if all oracle calls return YES.

The total length of substrings involved in the oracle calls is $\mathcal{O}(\rho m \cdot b) = \mathcal{O}(\rho n) = \mathcal{O}(\frac{b^2}{\phi}) = \mathcal{O}(\frac{\phi n^2}{\alpha^2})$, whereas the running time and number of oracle calls are $\mathcal{O}(\rho m) = \mathcal{O}(\frac{b}{\phi}) = \mathcal{O}(\frac{n}{\alpha})$.

To prove the correctness of this reduction (assuming the oracle makes no errors), let $B := \{i \in$

$[0\,..\,m) : \mathsf{ED}(X_i, Y_i) > \phi\}$ and observe that $\mathsf{ED}(X, Y) \le |B|b + m\phi$. Hence, if $\mathsf{ED}(X, Y) > \alpha$, then

$$|B| > \tfrac{\alpha - m\phi}{b} \ge \tfrac{\alpha b - 2n\phi}{b^2} \ge \tfrac{3\phi n - 2\phi n}{b^2} = \tfrac{1}{\rho}.$$

The probability that the algorithm returns YES is then at most

$$\left(1 - \tfrac{|B|}{m}\right)^{\lceil \rho m \rceil} \le \exp\left(-\tfrac{|B|}{m} \cdot \lceil \rho m \rceil\right) \le \exp\left(-\rho |B|\right) \le \exp(-1).$$

On the other hand, if $\mathsf{ED}(X, Y) \le \beta$, then Fact 2.1 implies that $\mathsf{ED}(X_i, Y_i) \le \beta$ for all $i \in [0\,..\,m)$. Consequently, all oracle calls return YES, and so does the entire algorithm. □

3.3 Our Reduction

Theorem 3.2. *There exists a randomized reduction that, given a parameter $\phi \in \mathbb{Z}_+$ and an instance of (β, α)-Gap Edit Distance satisfying $\frac{1}{10}\alpha \ge \phi \ge \beta \ge 1$, solves the instance using $\mathcal{O}(\frac{n}{\alpha})$ non-adaptive calls to an oracle for (β, ϕ)-Gap Edit Distance involving substrings of total length $\mathcal{O}(\frac{\phi n \log n}{\alpha})$. The reduction takes $\mathcal{O}(\frac{n}{\alpha})$ time, does not access the input strings, and errs with probability at most $\frac{1}{e}$.*

The algorithm For every *level* $p \in [0\,..\,\lceil \log n \rceil]$, partition X, Y into $m_p := \lceil \frac{n}{2^p} \rceil$ *blocks* of length 2^p (the last blocks might be shorter), given by $X_{p,i} = X[i \cdot 2^p\,..\,\min(n, (i+1)2^p))$ and $Y_{p,i} = Y[i \cdot 2^p\,..\,\min(n, (i+1)2^p))$ for $i \in [0\,..\,m_p)$.

Let $\rho := \frac{10\phi}{\alpha}$. For each level $p \in [\lceil \log \phi \rceil\,..\,\lfloor \log(\rho n) \rfloor]$, our algorithm performs $\lceil \rho m_p \rceil$ iterations. In each iteration, the algorithm chooses $i \in [0\,..\,m_p)$ uniformly at random and calls an oracle to solve an instance $(X_{p,i}, Y_{p,i})$ of the (β, ϕ)-Gap Edit Distance problem. The algorithm returns YES if and only if all oracle calls (across all levels) return YES.

Complexity Analysis The total length of substrings involved in the oracle calls is

$$\mathcal{O}\left(\sum_{p=\lceil \log \phi \rceil}^{\lfloor \log(\rho n) \rfloor} 2^p \cdot \lceil \rho m_p \rceil\right) = \mathcal{O}(\rho n \log n) = \mathcal{O}(\tfrac{\phi n \log n}{\alpha}).$$

The running time and the number of oracle calls are

$$\mathcal{O}\left(\sum_{p=\lceil \log \phi \rceil}^{\lfloor \log(\rho n) \rfloor} \lceil \rho m_p \rceil\right) = \mathcal{O}\left(\sum_{p=\lceil \log \phi \rceil}^{\lfloor \log(\rho n) \rfloor} \tfrac{\rho n}{2^p}\right) = \mathcal{O}(\tfrac{\rho n}{\phi}) = \mathcal{O}\left(\tfrac{n}{\alpha}\right).$$

Correctness The core of the analysis is the following lemma, which proves that an instance with large edit distance must contain, across all the levels, many blocks of "high cost". In the lemma, these blocks are denoted by B_p for level p, as illustrated in Figure 2.

Lemma 3.3. *Consider a threshold $\tau \in \mathbb{Z}_+$. For each level $p \in [0\,..\,\lceil \log n \rceil]$, let*

$$B_p := \{i \in [0\,..\,m_p) : \mathsf{ED}(X_{p,i}, Y_{p,i}) > \tau\}.$$

If $\mathsf{ED}(X, Y) > \tau$, then $\sum_{p=\lceil \log \tau \rceil}^{\lceil \log n \rceil} |B_p| > \frac{1}{2\tau}\mathsf{ED}(X, Y)$.

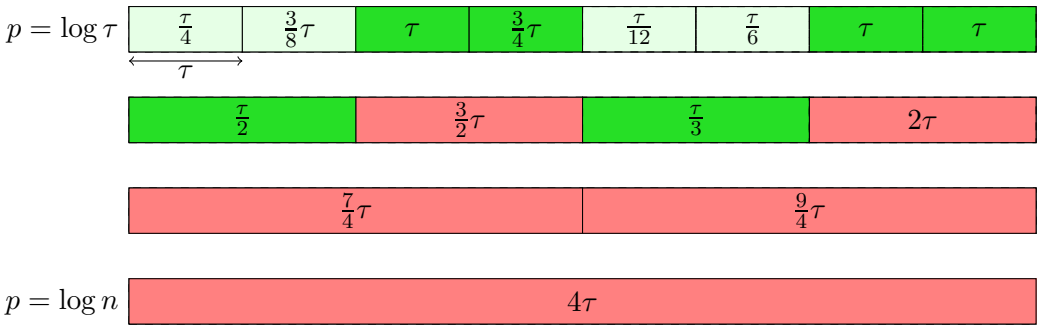

Figure 2: The multi-level partitioning of $[0\,..\,n)$ into blocks of length 2^p at each level p. Inside each rectangle we write the edit distance between the corresponding pair of blocks in X and in Y. A red color represents that the block is in B_p (high cost), and the two shades of green represent the remaining blocks. The dark green blocks color represents blocks in $\hat{B}_p \setminus B_p$ (green blocks with a red 'parent'); the crux of Lemma 3.3 is that the entire range can be decomposed into few such blocks.

Proof. We prove by induction on p that

$$\sum_{i \in B_p} \mathsf{ED}(X_{p,i}, Y_{p,i}) \le 2\tau \cdot \sum_{q=\lceil \log \tau \rceil}^{p} |B_q|.$$

The base case of $p < \lceil \log \tau \rceil$ holds trivially because then $\mathsf{ED}(X_{p,i}, Y_{p,i}) \le \tau$ for all $i \in [1\,..\,m_p)$, and thus $B_p = \emptyset$.

For the inductive step, consider $p \ge \lceil \log \tau \rceil$. Observe that each $i \in [0\,..\,m_p)$ satisfies $X_{p,i} = X_{p-1,2i} \cdot X_{p-1,2i+1}$ (if $X_{p-1,2n_p+1}$ is undefined, we set it to be empty) and, similarly, $Y_{p,i} = Y_{p-1,2i} \cdot Y_{p-1,2i+1}$. Consequently, we define $\hat{B}_{p-1} = \bigcup_{i \in B_p} \{2i, 2i+1\}$, with $|\hat{B}_{p-1}| = 2|B_p|$, and derive

$$\begin{aligned}
\sum_{i \in B_p} \mathsf{ED}(X_{p,i}, Y_{p,i}) &\overset{\text{Fact 2.2}}{\le} \sum_{j \in \hat{B}_{p-1}} \mathsf{ED}(X_{p-1,j}, Y_{p-1,j}) \\
&\le \sum_{j \in \hat{B}_{p-1} \setminus B_{p-1}} \mathsf{ED}(X_{p-1,j}, Y_{p-1,j}) + \sum_{j \in B_{p-1}} \mathsf{ED}(X_{p-1,j}, Y_{p-1,j}) \\
&\overset{\text{induction}}{\le} \tau \cdot |\hat{B}_{p-1} \setminus B_{p-1}| + 2\tau \cdot \sum_{q=\lceil \log \tau \rceil}^{p-1} |B_q| \\
&\le \tau \cdot |\hat{B}_{p-1}| + 2\tau \cdot \sum_{q=\lceil \log \tau \rceil}^{p-1} |B_q| \\
&= 2\tau \cdot \sum_{q=\lceil \log \tau \rceil}^{p} |B_q|.
\end{aligned}$$

This completes the inductive proof.

The lemma follows by applying the inequality proved above and observing that $\mathsf{ED}(X, Y) > \tau$ implies $B_{\lceil \log n \rceil} = \{0\}$:

$$\mathsf{ED}(X, Y) = \mathsf{ED}(X_{\lceil \log n \rceil, 0}, Y_{\lceil \log n \rceil, 0}) \le 2\tau \sum_{p=\lceil \log \tau \rceil}^{\lceil \log n \rceil} |B_p|. \qquad \square$$

Let us proceed with the correctness analysis of our algorithm. First, suppose that $\mathsf{ED}(X,Y) > \alpha$. Using Lemma 3.3 with $\tau := \phi$ and the fact $\alpha \geq 10\phi > \tau$, we then obtain

$$\sum_{p=\lceil \log \phi \rceil}^{\lceil \log n \rceil} |B_p| \geq \tfrac{\mathsf{ED}(X,Y)}{2\tau} > \tfrac{\alpha}{2\phi} = \tfrac{5}{\rho}.$$

Using a naive bound

$$\sum_{p=\lceil \log(\rho n) \rceil}^{\lceil \log n \rceil} |B_p| \leq \sum_{p=\lceil \log(\rho n) \rceil}^{\lceil \log n \rceil} m_p \leq \sum_{p=\lceil \log(\rho n) \rceil}^{\lceil \log n \rceil} \tfrac{2n}{2^p} \leq \tfrac{4}{\rho},$$

we conclude that $\sum_{p=\lceil \log \phi \rceil}^{\lfloor \log(\rho n) \rfloor} |B_p| \geq \frac{1}{\rho}$. For each level $p \in [\lceil \log \phi \rceil \,..\, \lfloor \log(\rho n) \rfloor]$, the probability that a single oracle (at a fixed iteration) call returns YES is at most $1 - \frac{|B_p|}{m_p} \leq \exp\left(-\frac{|B_p|}{m_p}\right)$. Across all levels $p \in [\lceil \log \phi \rceil \,..\, \lfloor \log(\rho n) \rfloor]$, the probability that all calls return YES is at most

$$\exp\left(-\sum_{p=\lceil \log \phi \rceil}^{\lfloor \log(\rho n) \rfloor} \tfrac{|B_p|}{m_p} \cdot \lceil \rho m_p \rceil\right) \leq \exp\left(-\sum_{p=\lceil \log \phi \rceil}^{\lfloor \log(\rho n) \rfloor} \rho |B_p|\right) \leq \exp(-1).$$

Thus, the algorithm returns YES with probability at most $\frac{1}{e}$.

Now, suppose that $\mathsf{ED}(X,Y) \leq \beta$. Then, Fact 2.1 implies $\mathsf{ED}(X_{p,i}, Y_{p,i}) \leq \beta$ for all $p \in [0\,..\,\lceil \log n \rceil]$ and $i \in [0\,..\,m_p)$. Consequently, each oracle call returns YES, so our algorithm also returns YES. This completes the proof of Theorem 3.2.

3.4 Corollaries (by Plugging Known Algorithms)

Corollary 3.4. *There exists a non-adaptive randomized algorithm that, given parameters $\varepsilon, \delta \in \mathbb{R}_+$ and an instance of (β,α)-Gap Edit Distance satisfying $\alpha \geq \lfloor f_{AN}(\varepsilon)\beta \rfloor$, solves the instance in time $\mathcal{O}(\frac{1+\beta}{1+\alpha} f_{AN}(\varepsilon) g_{AN}(\varepsilon) \cdot n^{1+\varepsilon} \log^2 n \cdot \log \frac{1}{\delta})$, using $\mathcal{O}(\frac{1+\beta}{1+\alpha} f_{AN}(\varepsilon) \cdot n \log n \cdot \log \frac{1}{\delta})$ queries to the input strings, and with error probability at most δ, where f_{AN} and g_{AN} are the functions of Theorem 2.6.*

Proof. If $\beta = 0$, then we use the algorithm of Fact 2.5, which takes time $\mathcal{O}(\frac{n}{1+\alpha})$. If $\lfloor f_{\mathrm{AN}}(\varepsilon)\beta \rfloor \leq \alpha < 10 f_{\mathrm{AN}}(\varepsilon)\beta$, we use the algorithm of Corollary 2.7, which takes $\mathcal{O}(g_{\mathrm{AN}}(\varepsilon) n^{1+\varepsilon})$ time and $\mathcal{O}(n)$ queries. In the remaining case of $0 < 10 f_{\mathrm{AN}}(\varepsilon)\beta \leq \alpha$, we use the reduction of Theorem 3.2 with $\phi = \lfloor f_{\mathrm{AN}}(\varepsilon)\beta \rfloor$ and the oracle implemented using Corollary 2.7. The oracle is randomized, so we need to set its error probability to $\Theta(\frac{\alpha}{n})$ so that all oracle calls are correct with large constant probability. An oracle call involving a pair of strings of length m takes time $\mathcal{O}(g_{\mathrm{AN}}(\varepsilon) m^{1+\varepsilon} \log n) = \mathcal{O}(g_{\mathrm{AN}}(\varepsilon) m \cdot n^{\varepsilon} \log n)$, and the total length of all strings involved in the oracle calls is $\mathcal{O}(\frac{\beta}{\alpha} \cdot f_{\mathrm{AN}}(\varepsilon) n \log n)$; therefore, the total running time is $\mathcal{O}(\frac{\beta}{\alpha} \cdot f_{\mathrm{AN}}(\varepsilon) g_{\mathrm{AN}}(\varepsilon) \cdot n^{1+\varepsilon} \log^2 n)$. This completes the algorithm's description for $\delta > \frac{1}{e}$. For general $\delta > 0$, we amplify the success probability by taking the majority answer among $\mathcal{O}(\log \frac{1}{\delta})$ independent repetitions of the entire algorithm. ☐

Next, we observe that the running time can be expressed as $\hat{\mathcal{O}}(\frac{1+\beta}{1+\alpha} \cdot n)$ as long as $\frac{\alpha}{\beta} = \omega(1)$.

Corollary 3.5. *Let $s : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ be a function such that $\lim_{x\to\infty} \frac{s(x)}{x} = \infty$. There exists a randomized algorithm that solves any instance of (β,α)-Gap Edit Distance with $\alpha \geq s(\beta)$ in time $\hat{\mathcal{O}}(\frac{1+\beta}{1+\alpha} \cdot n)$ correctly with high probability.*

Proof. Observe that there exists a function $\varepsilon : \mathbb{Z}_{\geq 0} \to \mathbb{R}_+$ such that $\varepsilon(n) = o(1)$, $f_{\text{AN}}(\varepsilon(n)) \leq \log n$, $g_{\text{AN}}(\varepsilon(n)) \leq \log n$, and $f_{\text{AN}}(\varepsilon(n)) \leq \frac{s(x)}{x}$ for all $n \in \mathbb{Z}_{\geq 0}$ and $x > \log n$. If $\alpha \geq f_{\text{AN}}(\varepsilon(n))\beta$, we use Corollary 3.4 with $\varepsilon = \varepsilon(n)$, which takes $\tilde{\mathcal{O}}(\frac{1+\beta}{1+\alpha} f_{\text{AN}}(\varepsilon(n)) g_{\text{AN}}(\varepsilon(n)) n^{1+\varepsilon(n)}) = \hat{\mathcal{O}}(\frac{1+\beta}{1+\alpha} n)$ time. If $\alpha < f_{\text{AN}}(\varepsilon(n))\beta$ and $\beta \leq \log n$, we use Theorem 2.4, which takes $\mathcal{O}(n + \beta^2) = \mathcal{O}(n) = \mathcal{O}(\frac{1+\alpha}{1+\beta} \cdot \frac{1+\beta}{1+\alpha} \cdot n) = \mathcal{O}(f_{\text{AN}}(\varepsilon(n)) \cdot \frac{1+\beta}{1+\alpha} \cdot n) = \hat{\mathcal{O}}(\frac{1+\beta}{1+\alpha} \cdot n)$ time. In the remaining case of $\alpha < f_{\text{AN}}(\varepsilon(n))\beta$ and $\beta > \log n$, we have $\alpha < f_{\text{AN}}(\varepsilon(n))\beta \leq s(\beta)$, which contradicts our assumption $\alpha \geq s(\beta)$. □

4 Improved Query Complexity

In this section, we improve the query complexity of the algorithm described in Section 3, solving the (β, α)-Gap Edit Distance with query complexity $\tilde{\mathcal{O}}\left(\frac{n\sqrt{\beta}}{\alpha}\right)$ provided that $\alpha \gg \beta$.

4.1 Overview

Recall that Theorem 3.2 provides a randomized reduction from the (β, α)-Gap Edit Distance problem to the (β, ϕ)-Gap Edit Distance problem. We used $\mathsf{ED}(X_{i,p}, Y_{i,p}) \leq \mathsf{ED}(X, Y)$ to justify the correctness for YES instances: The input can be safely rejected as soon as we discover that $\mathsf{ED}(X_{i,p}, Y_{i,p}) > \beta$ holds for some level p and index $i \in [0\,..\,m_p)$. If we were guaranteed that $\mathsf{ED}(X_{i,p}, Y_{i,p}) \leq \psi$ holds with good probability (over random $i \in [0\,..\,m_p)$) for some $\psi < \beta$, then we could use an oracle for the (ψ, ϕ)-Gap Edit Distance problem instead of the (β, ϕ)-Gap Edit Distance problem, i.e., our reduction would produce instances of the Gap Edit Distance problem with a larger gap. Unfortunately, this is not the case in general. In particular, if X consists of distinct characters and Y is obtained by moving the last $s \leq \frac{1}{2}n$ characters of X to the front, then $\mathsf{ED}(X_{i,p}, Y_{i,p}) = \mathsf{ED}(X, Y) = 2s$ holds for all levels $p \geq \log(2s)$ and indices $i \in [0\,..\,m_p)$. Nevertheless, in this example, the optimal alignment between $X_{i,p}$ and $Y_{i,p}$ is very simple: up to the shift by s characters (which effectively removes the last s characters of $X_{i,p}$ and the first s characters of $Y_{i,p}$), the two blocks are perfectly aligned. In general, for a fixed alignment $\mathcal{A}$ of X and Y, the induced alignment of $X_{i,p}$ and $Y_{i,p}$ performs the edits that $\mathcal{A}$ would make on $X_{i,p}$ and $Y_{i,p}$, and the only effect of edits that $\mathcal{A}$ makes outside these blocks is that some leading and trailing characters of $X_{i,p}$ and $Y_{i,p}$ need to be deleted (because $\mathcal{A}$ aligns them with characters outside the considered blocks). Thus, in a YES-instance, for a random $i \in [0\,..\,m_p)$, we always see up to β edits between $X_{i,p}$ and $Y_{i,p}$, but in expectation only $\frac{\beta}{m_p}$ of these edits cannot be attributed to a *shift* between $X_{i,p}$ and $Y_{i,p}$. This motivates the following notion.

Definition 4.1. For two strings $X, Y \in \Sigma^*$ and a threshold $\beta \in \mathbb{Z}_{\geq 0}$, define the *$\beta$-shifted edit distance* $\mathsf{ED}_\beta(X, Y)$ as

$$\min\left(\bigcup_{\Delta=0}^{\min(|X|,|Y|,\beta)} \left\{\mathsf{ED}(X[\Delta\,..\,|X|), Y[0\,..\,|Y|-\Delta)), \mathsf{ED}(X[0\,..\,|X|-\Delta), Y[\Delta\,..\,|Y|))\right\}\right).$$

Note that $\mathsf{ED}_\beta(X, Y) \leq \mathsf{ED}(X, Y) \leq \mathsf{ED}_\beta(X, Y) + 2\beta$ holds for every $\beta \in \mathbb{Z}_{\geq 0}$.

As argued above, the YES-instances of (β, α)-Gap Edit Distance satisfy $\mathbb{E}_i[\mathsf{ED}_\beta(X_{p,i}, Y_{p,i})] \leq \frac{\beta}{m_p}$. Given that we sample blocks with rate ρ, we expect to see $\mathcal{O}(1)$ blocks with $\mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) > \psi$ if we appropriately set $\psi = \tilde{\Theta}(\rho\beta)$. Moreover, this statement is also true with high probability. Furthermore, the argument in the proof of Theorem 3.2 can be strengthened to prove that, in a NO-instance, with high probability, we see $\Omega(1)$ blocks with $\mathsf{ED}(X_{p,i}, Y_{p,i}) > 3\phi$. Thus, instead of

using an oracle for the (β, ϕ)-Gap Edit Distance problem, we can use an oracle for the β-Shifted $(\psi, 3\phi)$-Gap Edit Distance problem defined as follows.

Problem 4.2 (β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance)**.** *Given strings $X, Y \in \Sigma^n$ and integer thresholds $\alpha \geq \beta \geq \gamma \geq 0$, return YES if $\mathsf{ED}_\beta(X, Y) \leq \gamma$, NO if $\mathsf{ED}(X, Y) > 3\alpha$, and an arbitrary answer otherwise.*

The idea to separate the shift from the "local" edits originates from [BEK+03], but they were only able to solve the (β, α)-Gap Edit Distance problem for $\alpha = \Omega(n)$. Combining their insight into our reduction of Theorem 3.2, we can handle a much wider range of parameters.

Similarly to [BEK+03], our algorithm is recursive in nature, with β decreased in each level (until it reaches 0). There is a key difference, though: They reduce the Shifted Gap Edit Distance problem to a more general problem, which becomes even more complicated in subsequent recursion levels. Our new insight is that, surprisingly, the Shifted Gap Edit Distance problem can be reduced back to (multiple instances of) Gap Edit Distance. This yields an algorithm with a much cleaner structure, and furthermore improves the query complexity because all these instances operate on relatively few different input strings, which can be easily exploited due to the non-adaptive nature of our approach. In fact, this query complexity is optimal (up to $\log^{\mathcal{O}(1)}(n)$ terms) for non-adaptive algorithms, as indicated by our lower bound, which generalizes the one in [BEK+03].

In this section, we present a solution that achieves this optimal query complexity but does not significantly improve the running time compared to Section 3. The latter issue is addressed in Section 5, where we exploit dependencies between the Shifted Gap Edit Distance instances produced throughout the recursive calls. Specifically, we provide a more efficient implementation for the batched version of the β-Shifted $(0, 3\alpha)$-Gap Edit Distance problem arising at the lowest level of our recursion. Moreover, we carefully adjust the parameters at the three lowest levels of recursion so that they produce batches with desirable properties. Up to logarithmic factors, our time bound matches that of [BEK+03], but the latter is valid only for $\alpha = \Omega(n)$.

4.2 From Gap Edit Distance to Shifted Gap Edit Distance

Below, we reduce the (β, α)-Gap Edit Distance problem to the β-Shifted $(\psi, 3\phi)$-Gap Edit Distance problem, where $\phi \geq \beta$ can be adjusted and ψ is set to $\tilde{\mathcal{O}}(\frac{\beta\phi}{\alpha})$. Our immediate application in Section 4.4 uses $\phi = \beta$, but subsequent speedups in Section 5 sometimes require $\phi \gg \beta$.

Lemma 4.3. *There exists a randomized reduction that, given a parameter $\phi \in \mathbb{Z}_+$ and an instance of (β, α)-Gap Edit Distance satisfying $\phi \geq \beta \geq \psi := \lfloor \frac{112\beta\phi\lceil \log n \rceil}{\alpha} \rfloor$, solves the instance using $\mathcal{O}(\frac{n}{\alpha})$ non-adaptive calls to an oracle for β-Shifted $(\psi, 3\phi)$-Gap Edit Distance involving substrings of total length $\mathcal{O}(\frac{\phi n \log n}{\alpha})$. The reduction costs $\mathcal{O}(\frac{n}{\alpha})$ time, does not access the input strings, and errs with probability at most $\frac{1}{e}$.*

Proof. Let $\rho = \frac{84\phi}{\alpha}$ and $\tau = 3\phi$. For each level $p \in [\lceil \log \tau \rceil \mathinner{..} \lfloor \log(\rho n) \rfloor]$, our algorithm performs $\lceil \rho m_p \rceil$ iterations. In each iteration, the algorithm chooses $i \in [0 \mathinner{..} m_p)$ uniformly at random and solves an instance $(X_{p,i}, Y_{p,i})$ of the β-Shifted $(\psi, 3\phi)$-Gap Edit Distance problem. Finally, the algorithm returns YES if the number $\hat{b}$ of oracle calls with NO answers satisfies $\hat{b} \leq 5$; if $\hat{b} \geq 6$, the algorithm returns NO.

Let us first analyze the complexity of the algorithm. The total length of all strings involved in the oracle calls is

$$\mathcal{O}\left(\sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} 2^p \cdot \lceil \rho m_p \rceil\right) = \mathcal{O}(\rho n \log n) = \mathcal{O}(\tfrac{\phi n \log n}{\alpha}).$$

The running time and the number of oracle calls are

$$\mathcal{O}\left(\sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} \lceil \rho m_p \rceil\right) = \mathcal{O}\left(\sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} \frac{\rho n}{2^p}\right) = \mathcal{O}\left(\tfrac{\rho n}{\tau}\right) = \mathcal{O}\left(\tfrac{n}{\alpha}\right).$$

Let us now proceed with the algorithm correctness. If $\mathsf{ED}(X, Y) > \alpha$, then we use Lemma 3.3. Due to $\alpha \geq 112\phi\lceil \log n \rceil > 3\phi = \tau$, we have $\sum_{p=\lceil \log \tau \rceil}^{\lceil \log n \rceil} |B_p| > \frac{\alpha}{2\tau} = \frac{14}{\rho}$. At the same time, $\sum_{p=\lceil \log(\rho n) \rceil}^{\lceil \log n \rceil} |B_p| \leq \sum_{p=\lceil \log n \rceil}^{\infty} \frac{2n}{2^p} \leq \frac{4}{\rho}$, so $\sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} |B_p| \geq \frac{10}{\rho}$. For a fixed iteration at level $p \in [\lceil \log \tau \rceil \,..\, \lceil \log n \rceil]$, the probability that the oracle call returns NO is at least $\frac{|B_p|}{m_p}$. Across all iterations and all levels $p \in [\lceil \log \tau \rceil \,..\, \lfloor \log(\rho n) \rfloor]$, the expected number of NO answers is therefore

$$\mathbb{E}\left[\hat{b}\right] \geq \sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} \frac{|B_p| \lceil \rho m_p \rceil}{m_p} \geq \rho \cdot \sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} |B_p| \geq 10.$$

By the Chernoff bound, we thus have

$$\mathbb{P}\left[\hat{b} \leq 5\right] = \mathbb{P}\left[\hat{b} \leq (1 - \tfrac{1}{2}) \cdot 10\right] \leq \exp\left(-\frac{(\frac{1}{2})^2 \cdot 10}{2}\right) = \exp(-\tfrac{5}{4}) < \exp(-1).$$

Finally, consider the case of $\mathsf{ED}(X, Y) \leq \beta$. For every level $p \in [0 \,..\, \lceil \log n \rceil]$, we define a set $G_p = \{i \in [1 \,..\, m_p) : \mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) > \psi\}$ corresponding to oracle calls that may return NO.

Claim 4.4. *We have* $\sum_{p=1}^{\lceil \log n \rceil} |G_p| \leq \frac{2\beta\lceil \log n \rceil}{\psi+1} \leq \frac{3}{2\rho}$.

Proof. For each level $p \in [1 \,..\, \lceil \log n \rceil]$, let us consider a partition $Y = \bigodot_{i \in [0 \,..\, m_p)} Y'_{p,i}$ such that $\mathsf{ED}(X, Y) = \sum_{i \in [0 \,..\, m_p)} \mathsf{ED}(X_{p,i}, Y'_{p,i})$. We claim that $\mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) \leq 2\mathsf{ED}(X_{p,i}, Y'_{p,i})$. If $|Y'_{p,i}| \leq \beta$, then $\mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) \leq \max(0, |X_{p,i}| - \beta) \leq \mathsf{ED}(X_{p,i}, Y'_{p,i})$ and the claim holds trivially. Thus, we assume $|Y'_{p,i}| > \beta$ and consider two cases.

First, suppose that $Y'_{p,i}$ starts at position $i \cdot 2^p + \Delta$ for $\Delta \geq 0$. We then have $\Delta \leq \mathsf{ED}(X[0 \,..\, i \cdot 2^p), Y[0 \,..\, i \cdot 2^p + \Delta)) = \sum_{j=0}^{i-1} \mathsf{ED}(X_{p,j}, Y'_{p,j}) \leq \mathsf{ED}(X, Y) \leq \beta$ and thus:

$$\begin{aligned}
\mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) &\leq \mathsf{ED}(X_{p,i}[0 \,..\, |X_{p,i}| - \Delta), Y_{p,i}[\Delta \,..\, |Y_{p,i}|)) \\
&\leq \mathsf{ED}(X_{p,i}[0 \,..\, |X_{p,i}| - \Delta), Y'_{p,i}[0 \,..\, |Y'_{p,i}| - \Delta)) + \big||Y_{p,i}| - |Y'_{p,i}|\big| \\
&\leq \mathsf{ED}(X_{p,i}, Y'_{p,i}) + \big||X_{p,i}| - |Y'_{p,i}|\big| \\
&\leq 2\mathsf{ED}(X_{p,i}, Y_{p,i}).
\end{aligned}$$

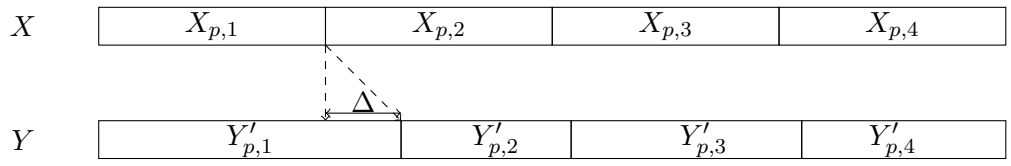

Figure 3: The partitions $X = \bigodot_{i \in [0 \,..\, m_p)} X_{p,i}$ and $Y = \bigodot_{i \in [0 \,..\, m_p)} Y'_{p,i}$.

Similarly, if $Y'_{p,i}$ starts at position $i \cdot 2^p - \Delta$ for some $\Delta \geq 0$, then $\Delta \leq \sum_{j=0}^{i-1} \mathsf{ED}(X_{p,j}, Y'_{p,j}) \leq \mathsf{ED}(X, Y) \leq \beta$ and

$$\begin{aligned}\mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) &\leq \mathsf{ED}(X_{p,i}[\Delta\,..\,|X_{p,i}|), Y_{p,i}[0\,..\,|Y_{p,i}| - \Delta)) \\ &\leq \mathsf{ED}(X_{p,i}[\Delta\,..\,|X_{p,i}|), Y'_{p,i}[\Delta\,..\,|Y'_{p,i}|)) + \big||Y_{p,i}| - |Y'_{p,i}|\big| \\ &\leq \mathsf{ED}(X_{p,i}, Y'_{p,i}) + \big||X_{p,i}| - |Y'_{p,i}|\big| \\ &\leq 2\mathsf{ED}(X_{p,i}, Y'_{p,i})\end{aligned}$$

Thus, $\sum_{p=1}^{\lceil \log n \rceil} \sum_{i=0}^{m_p - 1} \mathsf{ED}_\beta(X_{p,i}, Y_{p,i}) \leq 2\beta\lceil \log n \rceil$ and at most $\frac{2\beta\lceil \log n \rceil}{\psi+1}$ terms exceed ψ. □

For a fixed iteration at level $p \in [\lceil \log \tau \rceil\,..\,\lfloor \log(\rho n) \rfloor]$, the probability that the oracle call returns NO is at most $\frac{|G_p|}{m_p}$. Across all iterations and levels $p \in [\lceil \log \tau \rceil\,..\,\lfloor \log(\rho n) \rfloor]$, the expected number of NO answers is therefore

$$\mathbb{E}\left[\hat{b}\right] \leq \sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} \frac{|G_p|\lceil \rho m_p \rceil}{m_p} \leq 2\rho \sum_{p=\lceil \log \tau \rceil}^{\lfloor \log(\rho n) \rfloor} |G_p| \leq 3.$$

By the Chernoff bound, we thus have

$$\mathbb{P}\left[\hat{b} \geq 6\right] = \mathbb{P}\left[\hat{b} \geq (1+1) \cdot 3\right] \leq \exp(-\tfrac{1^2 \cdot 3}{2+1}) = \tfrac{1}{e}.$$ □

4.3 From Shifted Gap Edit Distance to Gap Edit Distance

Lemma 4.5. *There exists a deterministic reduction that, given instance of* β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance *satisfying* $\alpha \geq 3\gamma$*, solves the instance using* $\mathcal{O}\big(\frac{1+\beta}{1+\gamma}\big)$ *non-adaptive calls to an oracle for* $(3\gamma, \alpha)$-Gap Edit Distance *involving* $\mathcal{O}\big(\frac{\sqrt{1+\beta}}{\sqrt{1+\gamma}}\big)$ distinct *substrings. each of length at most* n*. The reduction takes* $\mathcal{O}(\frac{1+\beta}{1+\gamma})$ *time and does not access the input strings.*

Proof. Let $\xi \in [\gamma\,..\,\beta]$ be a parameter set later and $n' = n - \beta$. We use the oracle to solve several instances of the $(3\gamma, \alpha)$-Gap Edit Distance problem and return YES if and only if at least one oracle call returned YES. These instances are $(X[x\,..\,x+n'), Y[y\,..\,y+n'))$ for all $x \in [0\,..\,\beta]$ such that $x \equiv 0 \pmod{1+\xi}$ or $x \equiv \beta \pmod{1+\xi}$, all $y \in [0\,..\,\xi]$ such that $y \equiv 0 \pmod{1+\gamma}$, and all $y \in [\beta - \xi\,..\,\beta]$ such that $y \equiv \beta \pmod{1+\gamma}$. The number of oracle calls is at most $4\big\lceil \frac{1+\beta}{1+\xi} \big\rceil \big\lceil \frac{1+\xi}{1+\gamma} \big\rceil \leq 16 \cdot \frac{1+\beta}{1+\gamma}$, but the number of distinct strings involved in these calls is at most $2\big\lceil \frac{1+\beta}{1+\xi} \big\rceil + 2\big\lceil \frac{1+\xi}{1+\gamma} \big\rceil$, which is $\mathcal{O}\big(\frac{\sqrt{1+\beta}}{\sqrt{1+\gamma}}\big)$ if we set $1 + \xi = \lfloor \sqrt{(1+\beta)(1+\gamma)} \rfloor$.

Suppose that $\mathsf{ED}_\beta(X, Y) \leq \gamma$. First, consider the case when $\mathsf{ED}_\beta(X, Y) = \mathsf{ED}(X[\Delta\,..\,n), Y[0\,..\,n - \Delta))$ for some $\Delta \in [0\,..\,\beta)$. Let us choose the smallest $x \in [\Delta\,..\,\beta]$ with $x \equiv \beta \pmod{1+\xi}$ and the largest $y \in [0\,..\,x - \Delta]$ with $y \equiv 0 \pmod{1+\gamma}$. Observe that

$$\begin{aligned}\mathsf{ED}(X[x\,..\,x+n'), Y[y\,..\,y+n')) &\leq 2\gamma + \mathsf{ED}(X[x\,..\,x+n'), Y[x-\Delta\,..\,x-\Delta+n')) \\ &\leq 2\gamma + \mathsf{ED}(X[\Delta\,..\,n), Y[0\,..\,n-\Delta)) \leq 3\gamma.\end{aligned}$$

Hence, the oracle call for (x, y) must return YES.

Similarly, let us consider the case when $\mathsf{ED}_\beta(X, Y) = \mathsf{ED}(X[0\,..\,n - \Delta), Y[\Delta\,..\,n))$ for some $\Delta \in [0\,..\,\beta)$. Let us choose the largest $x \in [0\,..\,\beta - \Delta]$ with $x \equiv 0 \pmod{1+\xi}$ and the smallest $y \in [x + \Delta\,..\,\beta]$ with $y \equiv \beta \pmod{1+\gamma}$. Observe that

$$\begin{aligned}\mathsf{ED}(X[x\,..\,x+n'), Y[y\,..\,y+n')) &\leq 2\gamma + \mathsf{ED}(X[x\,..\,x+n'), Y[x+\Delta\,..\,x+\Delta+n')) \\ &\leq 2\gamma + \mathsf{ED}(X[0\,..\,n-\Delta), Y[\Delta\,..\,n)) \leq 3\gamma\end{aligned}$$

Hence, the oracle call for (x, y) must return YES.

Next, suppose that some oracle call for (x, y) returned YES. This implies $\mathsf{ED}(X[x\,..\,x+n'), Y[y\,..\,y+n')) \le \alpha$ for some $x, y \in [0\,..\,\beta]$. At the same time, we have

$$\mathsf{ED}(X[0\,..\,x), Y[0\,..\,y)) \le \max(x, y) \le \beta, \qquad \text{and}$$
$$\mathsf{ED}(X[x+n'\,..\,n), Y[y+n'\,..\,n)) \le \max(\beta - x, \beta - y) \le \beta.$$

Hence, $\mathsf{ED}(X, Y) \le \alpha + 2\beta \le 3\alpha$ holds as claimed. □

4.4 Baseline Implementation

Proposition 4.6. *There exists a non-adaptive algorithm that, given $h \in \mathbb{Z}_{\ge 0}$, $\delta \in \mathbb{R}_+$, and an instance of the (β, α)-Gap Edit Distance problem, satisfying $\beta < (336\lceil \log n \rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}}$, solves the instance in $\mathcal{O}\big(\frac{1+\beta}{1+\alpha} \cdot n \log^{2h} n \cdot \log \frac{1}{\delta} \cdot 2^{\mathcal{O}(h)}\big)$ time, using $\mathcal{O}\big(\frac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^{2h} n \cdot \log \frac{1}{\delta} \cdot 2^{\mathcal{O}(h)}\big)$ queries, and with error probability at most δ.*

Proposition 4.7. *There exists a non-adaptive algorithm that, given $h \in \mathbb{Z}_{\ge 0}$, $\delta \in \mathbb{R}_+$, and an instance of the β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance problem satisfying $\gamma < \frac{1}{3}(336\lceil \log n \rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}}$, solves the instance in $\mathcal{O}\big(\frac{1+\beta}{1+\alpha} \cdot n \log^{2h} n \cdot \log \frac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\big)$ time, using $\mathcal{O}\big(\frac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^{2h} n \cdot \log \frac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\big)$ queries, and with error probability at most δ.*

Proof of Propositions 4.6 and 4.7. As for (β, α)-Gap Edit Distance, let us assume that $\delta > \frac{1}{e}$; in general, we amplify the success probability by repeating the algorithm $\mathcal{O}(\log \frac{1}{\delta})$ times. If $\beta = 0$ (and, in particular, $h = 0$), we simply use Fact 2.5. Otherwise, we apply Lemma 4.3 with $\phi = \beta$ using our β-Shifted $(\psi, 3\phi)$-Gap Edit Distance algorithm (with parameters $h - 1$ and $\Theta(\frac{1}{n})$) as the oracle. This is valid because

$$\psi \le \frac{112\beta^2 \lceil \log n \rceil}{\alpha} < \frac{112\beta^2 \lceil \log n \rceil}{\beta^{\frac{h+1}{h}} \cdot (336\lceil \log n \rceil)^{\frac{h+1}{2}}} = \frac{1}{3} \cdot (336\lceil \log n \rceil)^{\frac{1-h}{2}} \cdot \phi^{\frac{h-1}{h}} \le \frac{1}{3} \cdot \phi < \beta.$$

The running time is

$$\mathcal{O}\left(\tfrac{\phi \log n}{\alpha} \cdot \tfrac{\beta}{\phi} \cdot n \log^{2h-2} n \cdot \log n \cdot 2^{\mathcal{O}(h-1)}\right) = \mathcal{O}\left(\tfrac{\beta}{\alpha} \cdot n \log^{2h} n \cdot 2^{\mathcal{O}(h)}\right),$$

whereas the query complexity is

$$\mathcal{O}\left(\tfrac{\phi \log n}{\alpha} \cdot \tfrac{\sqrt{\beta}}{\phi} \cdot n \log^{2h-2} n \cdot \log n \cdot 2^{\mathcal{O}(h-1)}\right) = \mathcal{O}\left(\tfrac{\sqrt{\beta}}{\alpha} \cdot n \log^{2h} n \cdot 2^{\mathcal{O}(h)}\right).$$

As for the β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance problem, we apply Lemma 4.5 using our $(3\gamma, \alpha)$-Gap Edit Distance algorithm (with parameters h and $\Theta(\frac{\delta}{n})$) as the oracle. This is valid since $3\gamma < (336\lceil \log n \rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}} \le \alpha$. The running time is

$$\mathcal{O}\left(\tfrac{1+\beta}{1+\gamma} \cdot \tfrac{1+3\gamma}{1+\alpha} \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right) = \mathcal{O}\left(\tfrac{1+\beta}{1+\alpha} \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right),$$

whereas the query complexity is

$$\mathcal{O}\left(\tfrac{\sqrt{1+\beta}}{\sqrt{1+\gamma}} \cdot \tfrac{\sqrt{1+3\gamma}}{1+\alpha} \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right) = \mathcal{O}\left(\tfrac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right).$$

□

5 Faster Implementation

In this section, we improve the running time while preserving the query complexity behind Proposition 4.6. The main trick is to consider a *batched* version on the (β,α)-Gap Edit Distance and β-Shifted $(\gamma,3\alpha)$-Gap Edit Distance problems: Instances $(X_1,Y_1),\ldots,(X_q,Y_q)$ form a *batch* if $X_1=\cdots=X_q$.

5.1 Shifted Gap Edit Distance for $h=0$

Lemma 5.1. *There exists a non-adaptive algorithm that, given a parameter $\delta\in\mathbb{R}_+$, and a batch of q instances of β-Shifted $(0,3\alpha)$-Gap Edit Distance, solves the instances in $\mathcal{O}\big(\frac{\sqrt{q(q+\beta)}}{1+\alpha}\cdot n\log\frac{n}{\delta}\big)$ time with each answer correct with probability at least $1-\delta$. Moreover, at most $\mathcal{O}\big(\frac{1+\beta}{1+\alpha}\cdot n\log\frac{n}{\delta}\big)$ characters of the common string X are accessed.*

Proof. We simulate the algorithm in the proof of Lemma 4.5, setting $1+\xi=\big\lceil\frac{\sqrt{q+\beta}}{\sqrt{q}}\big\rceil$. For each instance, this yields $\mathcal{O}(1+\beta)$ oracle calls asking to solve the $(0,\alpha)$-Gap Edit Distance problem for (X',Y') with $|X'|=|Y'|=n'\le n$. The set of pairs (X',Y') involved in these calls can be expressed as $\mathcal{X}\times\mathcal{Y}$, where $|\mathcal{X}|=\mathcal{O}\big(\frac{1+\beta}{1+\xi}\big)=\mathcal{O}\big(\min\big(1+\beta,\sqrt{q(q+\beta)}\big)\big)$ and $|\mathcal{Y}|=\mathcal{O}(1+\xi)=\mathcal{O}\big(\frac{\sqrt{q+\beta}}{\sqrt{q}}\big)$. Moreover, since our algorithm is non-adaptive, the set $\mathcal{X}$ is the same for all q instances.

Recall that the reduction of Lemma 4.5 returns YES if and only if at least one of the oracle calls returns YES. To simulate implementing the calls using the algorithm of Fact 2.5, we construct a random sample $S\subseteq[0\,..\,n')$ of size $\Theta\big(\frac{n'\log\frac{n}{\delta}}{1+\alpha}\big)$. We build a set $\mathcal{X}_S:=\{X'[S]:X'\in\mathcal{X}\}$ and, for each $Y'\in\mathcal{Y}$, we check whether $Y'[S]\in\mathcal{X}_S$. If so, then we return YES. If processing all $Y'\in\mathcal{Y}$ is completed without a YES answer, then we return NO (for the given instance).

If $\mathsf{ED}(X',Y')>\alpha$ holds for all $(X',Y')\in\mathcal{X}\times\mathcal{Y}$, then, by the union bound, the probability that $Y'[S]\in\mathcal{X}_S$ holds for some $Y'\in\mathcal{Y}$ is at most δ. Thus, the algorithm returns YES with probability at most δ. On the other hand, if $X'=Y'$ holds for some $(X',Y')\in\mathcal{X}\times\mathcal{Y}$, then $X'[S]=Y'[S]$, and we do return YES due to $Y'[S]\in\mathcal{X}_S$.

If $\mathcal{X}_S$ is implemented as a ternary trie [BS79], then its construction cost is

$$\mathcal{O}\left(|\mathcal{X}|\left(\log|\mathcal{X}|+\tfrac{n\log\frac{n}{\delta}}{1+\alpha}\right)\right)=\mathcal{O}\left(|\mathcal{X}|\cdot\tfrac{n\log\frac{n}{\delta}}{1+\alpha}\right)=\mathcal{O}\left(\frac{\min\left(1+\beta,\sqrt{q(q+\beta)}\right)}{1+\alpha}\cdot n\log\tfrac{n}{\delta}\right).$$

The time complexity of the second step is

$$\mathcal{O}\left(|\mathcal{Y}|\left(\log|\mathcal{X}|+\tfrac{n\log\frac{n}{\delta}}{1+\alpha}\right)\right)=\mathcal{O}\left(\tfrac{\sqrt{q+\beta}}{(1+\alpha)\sqrt{q}}\cdot n\log\tfrac{n}{\delta}\right)$$

per instance and $\mathcal{O}\big(\frac{\sqrt{q(q+\beta)}}{1+\alpha}\cdot n\log\frac{n}{\delta}\big)$ in total. ∎

5.2 Gap Edit Distance for $h=1$

Lemma 5.2. *There exists a non-adaptive algorithm that, given a parameter $\delta\in\mathbb{R}_+$ and a batch of q instances of (β,α)-Gap Edit Distance satisfying $\beta^2\le\frac{\alpha}{336\lceil\log n\rceil}$, solves the instances in $\mathcal{O}\big(\frac{\sqrt{q(q+\beta)}}{1+\alpha}\cdot n\log^2 n\cdot\log\frac{1}{\delta}\big)$ time with each answer correct with probability at least $1-\delta$. Moreover, at most $\mathcal{O}\big(\frac{1+\beta}{1+\alpha}\cdot n\log^2 n\cdot\log\frac{1}{\delta}\big)$ characters of the common string X are accessed.*

Proof. Let us assume that $\delta > \frac{1}{e}$; in general, we amplify the success probability by repeating the algorithm $\mathcal{O}(\log \frac{1}{\delta})$ times. If $\beta = 0$, then we simply use Fact 2.5. Otherwise, we apply Lemma 4.3 with $\phi = \beta$ and the algorithm of Lemma 5.1 (with parameter $\Theta(\frac{1}{n})$) as the oracle. This is valid because

$$\psi = \left\lfloor \frac{112\beta^2 \lceil \log n \rceil}{\alpha} \right\rfloor \leq \left\lfloor \frac{112\alpha \lceil \log n \rceil}{336\alpha \lceil \log n \rceil} \right\rfloor = \lfloor \tfrac{1}{3} \rfloor = 0.$$

Since the algorithm of Lemma 4.3 is non-adaptive, the queries remain batched. The total running time is

$$\mathcal{O}\left(\tfrac{\phi \log n}{\alpha} \cdot \tfrac{\sqrt{q(q+\beta)}}{1+\phi} \cdot n \log n \right) = \mathcal{O}\left(\tfrac{\sqrt{q(q+\beta)}}{1+\alpha} \cdot n \log^2 n \right),$$

whereas the number of accessed characters of the common string X does not exceed

$$\mathcal{O}\left(\tfrac{\phi \log n}{\alpha} \cdot \tfrac{1+\beta}{1+\phi} \cdot n \log n \right) = \mathcal{O}\left(\tfrac{1+\beta}{1+\alpha} \cdot n \log^2 n \right). \qquad \square$$

5.3 Shifted Gap Edit Distance for $h = 1$

Lemma 5.3. *There exists a non-adaptive algorithm that, given a parameter $\delta \in \mathbb{R}_+$ and a batch of q instances of β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance satisfying $\gamma^2 \leq \frac{\alpha}{3024\lceil \log n \rceil}$, solves the instances in*

$$\mathcal{O}\left(\left(\tfrac{\sqrt{q(q+\beta)}}{1+\alpha} + \tfrac{q(1+\beta)}{1+\alpha} \cdot \sqrt{\tfrac{\log n}{1+\alpha}} \right) n \log^2 n \cdot \log \tfrac{n}{\delta} \right)$$

time, using $\mathcal{O}\big(\frac{\sqrt{q(q+\beta)}}{1+\alpha} \cdot n \log^2 n \cdot \log \frac{n}{\delta}\big)$ queries, and with each answer correct with probability at least $1 - \delta$. Moreover, at most $\mathcal{O}\big(\frac{1+\beta}{1+\alpha} \cdot n \log^2 n \cdot \log \frac{n}{\delta}\big)$ characters of the common string X are accessed.

Proof. If $\gamma = 0$, then we simply use Lemma 5.1. Consequently, we henceforth assume $\alpha \geq \beta \geq \gamma > 0$.

In the remaining case, we proceed as in the proof of Lemma 4.5 except that we artificially increase γ to

$$\bar{\gamma} := \min\left(\beta, \left\lfloor \sqrt{\tfrac{\alpha}{3024\lceil \log n \rceil}} \right\rfloor\right),$$

set $\xi = \max\left(\bar{\gamma}, \min\left(\beta, \left\lfloor \frac{\bar{\gamma}\sqrt{\beta}}{\sqrt{q}} \right\rfloor\right)\right)$, and use the algorithm of Lemma 5.2 (with parameter $\Theta(\frac{\delta}{n})$) as the oracle; this is valid due to $(3\bar{\gamma})^2 \leq \frac{\alpha}{336\lceil \log n \rceil}$. The input instances are solved using $\mathcal{O}\big(\frac{\beta}{\xi}\big)$ batches of $\mathcal{O}(\frac{q\xi}{\bar{\gamma}})$ oracle calls, and these batches only differ in the strings X' (common to each batch).

If $q \leq \frac{\bar{\gamma}^2}{\beta}$, then $\xi = \beta$, so the input instances are solved using $\mathcal{O}(1)$ batches of $\mathcal{O}(\frac{q\beta}{\bar{\gamma}}) = \mathcal{O}(\bar{\gamma})$ oracle calls, and hence the total running time and the query complexity are

$$\mathcal{O}\left(\tfrac{\sqrt{\frac{q\beta}{\bar{\gamma}} \cdot \bar{\gamma}}}{\alpha} \cdot n \log^2 n \cdot \log \tfrac{n}{\delta} \right) = \mathcal{O}\left(\tfrac{\sqrt{q\beta}}{\alpha} \cdot n \log^2 n \cdot \log \tfrac{n}{\delta} \right).$$

If $q > \frac{\bar{\gamma}^2}{\beta}$, then the input instances are solved using $\mathcal{O}\big(\frac{\beta}{\xi}\big)$ batches of $\Theta(\frac{q\xi}{\bar{\gamma}}) = \Omega(\frac{\xi\bar{\gamma}}{\beta}) = \Omega(\bar{\gamma})$ oracle calls. Hence, the total running time is

$$\begin{aligned} \mathcal{O}\left(\tfrac{\beta}{\xi} \cdot \tfrac{q\xi}{\bar{\gamma}} \cdot \tfrac{1}{\alpha} \cdot n \log^2 n \cdot \log \tfrac{n}{\delta} \right) &= \mathcal{O}\left(\tfrac{q\beta}{\alpha\bar{\gamma}} \cdot n \log^2 n \cdot \log \tfrac{n}{\delta} \right) \\ &= \mathcal{O}\left(\left(\tfrac{q}{\alpha} + \tfrac{q\beta}{\alpha} \cdot \sqrt{\tfrac{\log n}{\alpha}} \right) \cdot n \log^2 n \cdot \log \tfrac{n}{\delta} \right). \end{aligned}$$

Moreover, the batches of oracle calls differ only by the strings X' (common to each batch). Hence, the query complexity can be bounded as follows:

$$\mathcal{O}\left(\left(\tfrac{\beta}{\xi}\cdot\tfrac{\bar{\gamma}}{\alpha}+\tfrac{q\xi}{\bar{\gamma}}\cdot\tfrac{1}{\alpha}\right)\cdot n\log^2 n\cdot\log\tfrac{n}{\delta}\right)=\mathcal{O}\left(\left(\tfrac{\beta\gamma}{\alpha\beta}+\tfrac{\beta\bar{\gamma}\sqrt{q}}{\alpha\bar{\gamma}\sqrt{\beta}}+\tfrac{q\bar{\gamma}\sqrt{\beta}}{\alpha\bar{\gamma}\sqrt{q}}+\tfrac{q\bar{\gamma}}{\alpha\bar{\gamma}}\right)\cdot n\log^2 n\cdot\log\tfrac{n}{\delta}\right)$$
$$=\mathcal{O}\left(\left(\tfrac{\bar{\gamma}}{\alpha}+\tfrac{\sqrt{q\beta}}{\alpha}+\tfrac{q}{\alpha}\right)\cdot n\log^2 n\cdot\log\tfrac{n}{\delta}\right)=\mathcal{O}\left(\tfrac{\sqrt{q(q+\beta)}}{\alpha}\cdot n\log^2 n\cdot\log\tfrac{n}{\delta}\right).$$

In any case, the number of accessed characters of the common string X does not exceed

$$\mathcal{O}\left(\tfrac{\beta}{\xi}\cdot\tfrac{\bar{\gamma}}{\alpha}\cdot n\log^2 n\cdot\log\tfrac{n}{\delta}\right)=\mathcal{O}\left(\tfrac{\beta}{\alpha}\cdot n\log^2 n\cdot\log\tfrac{n}{\delta}\right).$$ □

5.4 Gap Edit Distance for $h = 2$

Lemma 5.4. *There exists a non-adaptive algorithm that, given a parameter $\delta\in\mathbb{R}_+$ and a batch of q instances of β-Shifted $(\gamma,3\alpha)$-Gap Edit Distance satisfying $\beta\le\frac{\alpha^{2/3}}{336\lceil\log n\rceil}$, solves the instances in*

$$\mathcal{O}\left(\left(\tfrac{\sqrt{q(q+\beta)}}{1+\alpha}+\tfrac{q(1+\beta)^2}{(1+\alpha)^2}\cdot\log^2 n\right)\cdot n\log^4 n\cdot\log\tfrac{1}{\delta}\right)$$

time, using $\mathcal{O}\big(\tfrac{\sqrt{q(q+\beta)}}{1+\alpha}\cdot n\log^4 n\cdot\log\tfrac{1}{\delta}\big)$ queries, and with each answer correct with probability at least $1-\delta$. Moreover, at most $\mathcal{O}\big(\tfrac{1+\beta}{1+\alpha}\cdot n\log^4 n\cdot\log\tfrac{1}{\delta}\big)$ characters of the common string X are accessed.

Proof. Let us assume that $\delta>\frac{1}{e}$; in general, we amplify the success probability by repeating the algorithm $\mathcal{O}(\log\frac{1}{\delta})$ times. If $\beta^2\le\frac{\alpha}{336\lceil\log n\rceil}$, then we simply use Lemma 5.2. Otherwise, we apply Lemma 4.3 with $\phi=\left\lfloor\frac{\alpha^2}{\beta^2(336\lceil\log n\rceil)^3}\right\rfloor$ and the algorithm of Lemma 5.3 (with parameter $\Theta(\frac{1}{n})$) as the oracle. This is valid due to the following inequalities:

$$\psi\le\tfrac{112\beta\phi\lceil\log n\rceil}{\alpha}\le\tfrac{112\beta\alpha^2\lceil\log n\rceil}{\alpha\beta^2(336\lceil\log n\rceil)^3}=\tfrac{\alpha}{3\beta(336\lceil\log n\rceil)^2}<\tfrac{336\beta^2\lceil\log n\rceil}{3\beta(336\lceil\log n\rceil)^2}=\tfrac{\beta}{1008\lceil\log n\rceil}<\beta,$$
$$\psi^2\le\tfrac{(112\beta\phi\lceil\log n\rceil)^2}{\alpha^2}\le\tfrac{\phi\cdot(112\beta\lceil\log n\rceil)^2\cdot\alpha^2}{\alpha^2\cdot\beta^2(336\lceil\log n\rceil)^3}=\tfrac{\phi}{3024\lceil\log n\rceil},$$
$$\phi=\left\lfloor\tfrac{\alpha^2}{\beta^2(336\lceil\log n\rceil)^3}\right\rfloor\ge\left\lfloor\tfrac{(336\beta\lceil\log n\rceil)^3}{(336\beta\lceil\log n\rceil)^3}\right\rfloor=\beta.$$

Since the algorithm of Lemma 4.3 is non-adaptive, the queries remain batched.

The total running time is

$$\mathcal{O}\left(\tfrac{\phi\log n}{\alpha}\cdot\left(\tfrac{\sqrt{q(q+\beta)}}{\phi}+\tfrac{q\beta}{\phi}\cdot\sqrt{\tfrac{\log n}{\phi}}\right)\cdot n\log^3 n\right)=\mathcal{O}\left(\left(\tfrac{\sqrt{q(q+\beta)}}{\alpha}+\tfrac{q\beta\sqrt{\log n}}{\alpha\sqrt{\phi}}\right)\cdot n\log^4 n\right)$$
$$=\mathcal{O}\left(\left(\tfrac{\sqrt{q(q+\beta)}}{\alpha}+\tfrac{q\beta^2}{\alpha^2}\cdot\log^2 n\right)\cdot n\log^4 n\right),$$

whereas the query complexity is

$$\mathcal{O}\left(\tfrac{\phi\log n}{\alpha}\cdot\tfrac{\sqrt{q(q+\beta)}}{\phi}\cdot n\log^3 n\right)=\mathcal{O}\left(\tfrac{\sqrt{q(q+\beta)}}{\alpha}\cdot n\log^4 n\right).$$

Moreover, the number of accessed characters of the common string X does not exceed

$$\mathcal{O}\left(\tfrac{\phi\log n}{\alpha}\cdot\tfrac{\beta}{\phi}\cdot n\log^3 n\right)=\mathcal{O}\left(\tfrac{\beta}{\alpha}\cdot n\log^4 n\right).$$ □

5.5 Shifted Gap Edit Distance for $h = 2$

Lemma 5.5. *There exists a non-adaptive algorithm that, given a parameter $\delta \in \mathbb{R}_+$ and an instance of β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance satisfying $\gamma \leq \frac{\alpha^{2/3}}{1008\lceil \log n \rceil}$, solves the instance in*

$$\mathcal{O}\left(\left(\tfrac{\sqrt{1+\beta}}{1+\alpha} + \tfrac{(1+\beta)(1+\gamma)}{(1+\alpha)^2} \cdot \log^2 n\right) \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right)$$

time, using $\mathcal{O}\big(\frac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^4 n \cdot \log \frac{n}{\delta}\big)$ queries, and with error probability at most δ.

Proof. If $\gamma^2 \leq \frac{\alpha}{3024\lceil \log n \rceil}$, then we simply use Lemma 5.3. In this case, due to $\alpha \geq \beta$, the running time is

$$\mathcal{O}\left(\left(\tfrac{\sqrt{1+\beta}}{1+\alpha} + \tfrac{1+\beta}{1+\alpha} \cdot \sqrt{\tfrac{\log n}{1+\alpha}}\right) \cdot n \log^2 n \cdot \log \tfrac{n}{\delta}\right) = \mathcal{O}\left(\tfrac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^{2.5} n \cdot \log \tfrac{n}{\delta}\right).$$

Otherwise, we proceed as in the proof of Lemma 4.5 except that we set $\xi = \min(\beta, \lfloor \gamma\sqrt{\beta} \rfloor)$ and use Lemma 5.4 (with parameter $\Theta(\frac{\delta}{n})$) as the oracle (this is valid due to $3\gamma \leq \frac{\alpha^{2/3}}{336\lceil \log n \rceil}$). The input instances are solved using $\mathcal{O}\big(\frac{\beta}{\xi}\big)$ batches of $\mathcal{O}(\frac{\xi}{\gamma})$ oracle calls, and these batches only differ in the strings X' (common to each batch).

If $\beta \leq \gamma^2$, then $\xi = \beta$, so the input instances are solved using $\mathcal{O}(1)$ batches of $\mathcal{O}(\frac{\beta}{\gamma}) = \mathcal{O}(\gamma)$ queries, and the hence the running time is

$$\mathcal{O}\left(\left(\tfrac{\sqrt{\frac{\beta}{\gamma}\cdot\gamma}}{\alpha} + \tfrac{\frac{\beta}{\gamma}\cdot\gamma^2}{\alpha^2} \cdot \log^2 n\right) \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right) = \mathcal{O}\left(\left(\tfrac{\sqrt{\beta}}{\alpha} + \tfrac{\beta\gamma}{\alpha^2} \cdot \log^2 n\right) \cdot n \log^2 n \cdot \log \tfrac{n}{\delta}\right),$$

whereas the query complexity is

$$\mathcal{O}\left(\tfrac{\sqrt{\frac{\beta}{\gamma}\cdot\gamma}}{\alpha} \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right) = \mathcal{O}\left(\tfrac{\sqrt{\beta}}{\alpha} \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right).$$

Otherwise, $\xi = \gamma\sqrt{\beta}$, so the input instances are solved using $\mathcal{O}(\frac{\sqrt{\beta}}{\gamma})$ batches of $\Theta(\sqrt{\beta}) = \Omega(\gamma)$ queries, and hence the running time is

$$\begin{aligned}\mathcal{O}\left(\tfrac{\sqrt{\beta}}{\gamma} \cdot \left(\tfrac{\sqrt{\beta}}{\alpha} + \tfrac{\sqrt{\beta}\cdot\gamma^2}{\alpha^2} \cdot \log^2 n\right) \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right) &= \mathcal{O}\left(\left(\tfrac{\beta\gamma}{\alpha\gamma^2} + \tfrac{\beta\gamma}{\alpha^2} \cdot \log^2 n\right) \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right) \\ &= \mathcal{O}\left(\left(\tfrac{\beta\gamma}{\alpha\frac{\alpha}{\log n}} + \tfrac{\beta\gamma}{\alpha^2} \cdot \log^2 n\right) \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right) = \mathcal{O}\left(\tfrac{\beta\gamma}{\alpha^2} \cdot n \log^6 n \cdot \log \tfrac{n}{\delta}\right),\end{aligned}$$

where the lower bound on γ is due to $\gamma^2 > \frac{\alpha}{3024\lceil \log n \rceil}$. Moreover, the batches of oracle calls differ only by the strings X' (common to each batch). Hence, the query complexity can be bounded as follows:

$$\mathcal{O}\left(\left(\tfrac{\sqrt{\beta}}{\gamma} \cdot \tfrac{\gamma}{\alpha} + \tfrac{\sqrt{\beta}}{\alpha}\right) \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right) = \mathcal{O}\left(\tfrac{\sqrt{\beta}}{\alpha} \cdot n \log^4 n \cdot \log \tfrac{n}{\delta}\right). \qquad \square$$

5.6 Gap Edit Distance and Shifted Gap Edit Distance for $h \geq 3$

Theorem 5.6. *There exists a non-adaptive algorithm that, given $h \in \mathbb{Z}_{\geq 0}$, $\delta \in \mathbb{R}_+$, and an instance of (β, α)-Gap Edit Distance satisfying $\beta < (336\lceil \log n \rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}}$, solves the instance in*

$$\mathcal{O}\left(\left(\tfrac{\sqrt{1+\beta}}{1+\alpha} + \tfrac{(1+\beta)^2}{(1+\alpha)^2} \cdot \log^h n\right) \cdot n \log^{2h} n \cdot \log \tfrac{1}{\delta} \cdot 2^{\mathcal{O}(h)}\right)$$

time, using $\mathcal{O}\big(\frac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^{2h} n \cdot \log \frac{1}{\delta} \cdot 2^{\mathcal{O}(h)}\big)$ queries, and with error probability at most δ.

Theorem 5.7. *There exists a non-adaptive algorithm that, given $h \in \mathbb{Z}_{\geq 2}$, $\delta \in \mathbb{R}_+$, and an instance of β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance satisfying $\gamma < \frac{1}{3}(336\lceil \log n \rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}}$, solves the instance in*

$$\mathcal{O}\left(\left(\tfrac{\sqrt{1+\beta}}{1+\alpha} + \tfrac{(1+\beta)(1+\gamma)}{(1+\alpha)^2} \cdot \log^h n\right) \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right)$$

time, using $\mathcal{O}\left(\frac{\sqrt{1+\beta}}{1+\alpha} \cdot n \log^{2h} n \cdot \log \frac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right)$ queries, and with error probability at most δ.

Proof of Theorems 5.6 and 5.7. As for (β, α)-Gap Edit Distance, let us assume that $\delta > \frac{1}{e}$; in general, we amplify the success probability by repeating the algorithm $\mathcal{O}(\log \frac{1}{\delta})$ times. We use Fact 2.5 and Lemmas 5.2 and 5.4 when applicable. In particular, this covers $\beta \leq \frac{\alpha^{2/3}}{336\lceil \log n \rceil}$ and $h \leq 2$. Otherwise, we apply Lemma 4.3 with $\phi = \beta$ using our algorithm for β-Shifted $(\psi, 3\phi)$-Gap Edit Distance (with parameters $h-1$ and $\Theta(\frac{1}{n})$) as the oracle. This is valid because

$$\psi \leq \frac{112\beta^2\lceil \log n \rceil}{\alpha} < \frac{112\beta^2\lceil \log n \rceil}{\beta^{\frac{h+1}{h}} \cdot (336\lceil \log n \rceil)^{\frac{h+1}{2}}} = \frac{1}{3} \cdot (336\lceil \log n \rceil)^{\frac{1-h}{2}} \cdot \phi^{\frac{h-1}{h}}.$$

The running time is

$$\mathcal{O}\left(\tfrac{\phi \log n}{\alpha} \cdot \left(\tfrac{\sqrt{\beta}}{\phi} + \tfrac{\psi\beta}{\phi^2} \cdot \log^{h-1} n\right) \cdot n \log^{2h-1} n \cdot 2^{\mathcal{O}(h-1)}\right) = \mathcal{O}\left(\left(\tfrac{\sqrt{\beta}}{\alpha} + \tfrac{\beta^2}{\alpha^2} \cdot \log^h n\right) \cdot n \log^{2h} n \cdot 2^{\mathcal{O}(h)}\right),$$

whereas the query complexity is

$$\mathcal{O}\left(\tfrac{\phi \log n}{\alpha} \cdot \tfrac{\sqrt{\beta}}{\phi} \cdot n \log^{2h-1} n \cdot 2^{\mathcal{O}(h-1)}\right) = \mathcal{O}\left(\tfrac{\sqrt{\beta}}{\alpha} \cdot n \log^{2h} n \cdot 2^{\mathcal{O}(h-1)}\right).$$

As for β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance, we use Lemmas 5.1, 5.3, and 5.5 when applicable. In particular, this covers $\gamma \leq \frac{\alpha^{2/3}}{1008\lceil \log n \rceil}$ and $h \leq 2$. Otherwise, we apply Lemma 4.5 using our algorithm for $(3\gamma, \alpha)$-Gap Edit Distance (with parameters h and $\Theta(\frac{\delta}{n})$) as the oracle. This is valid because $3\gamma \leq (336\lceil \log n \rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}} \leq \alpha$.

The running time is

$$\mathcal{O}\left(\tfrac{\beta}{\gamma} \cdot \left(\tfrac{\sqrt{\gamma}}{\alpha} + \tfrac{\gamma^2}{\alpha^2} \cdot \log^h n\right) \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right) = \mathcal{O}\left(\tfrac{\beta}{\gamma} \cdot \tfrac{\gamma^2}{\alpha^2} \cdot n \log^{3h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right),$$

whereas the query complexity is

$$\mathcal{O}\left(\tfrac{\sqrt{\beta}}{\sqrt{\gamma}} \cdot \tfrac{\sqrt{\gamma}}{\alpha} \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right) = \mathcal{O}\left(\tfrac{\sqrt{\beta}}{\alpha} \cdot n \log^{2h} n \cdot \log \tfrac{n}{\delta} \cdot 2^{\mathcal{O}(h)}\right). \qquad \square$$

6 Matching Lower Bound for Non-Adaptive Query Complexity

In this section we strengthen the following lower bound of [BEK+03] for the $(\beta, \frac{n}{6})$-Gap Edit Distance problem.

Proposition 6.1 ([BEK+03])**.** *For all integers $n, \alpha, \beta \in \mathbb{Z}_+$ such that $\frac{n}{6} = \alpha \geq \beta$, every algorithm solving all instances of the (β, α)-Gap Edit Distance problem has worst-case query complexity $\Omega(\sqrt{\beta})$ or error probability exceeding $\frac{1}{3}$.*

Theorem 6.2. *For all integers $n, \alpha, \beta \in \mathbb{Z}_+$ such that $\frac{n}{6} \geq \alpha \geq \beta$, every non-adaptive algorithm solving all instances the (β, α)-Gap Edit Distance problem has expected query complexity $\Omega(\frac{n\sqrt{\beta}}{\alpha})$ or error probability exceeding $\frac{1}{3}$.*

Proof. Suppose that, for some fixed $n, \alpha, \beta \in \mathbb{Z}_+$ with $\frac{n}{6} \geq \alpha \geq \beta$, there exists a non-adaptive algorithm A that uses q queries in expectation and errs with probability at most $\frac{1}{3}$. We shall derive an algorithm A' for $n = 6\alpha$ that uses $\frac{486q\alpha}{n}$ queries in the worst case; if $q = o(\frac{n\sqrt{\beta}}{\alpha})$, this would contradict Proposition 6.1.

Let us first define an algorithm A^3 that runs A three times and returns the dominant answer; it has error probability to $\frac{1+3\cdot 2}{27} = \frac{7}{27}$ and expected query complexity to $3q$. For each $i \in [0\,..\,\lfloor\frac{n}{6\alpha}\rfloor)$, let q_i be the expected number of queries that A^3 makes to $X[6\alpha i\,..\,6\alpha(i+1))$ and $Y[6\alpha i\,..\,6\alpha(i+1))$; since A^3 is non-adaptive, these values do not depend on X or Y. By linearity of expectation, we have $\sum_i q_i \leq 3q$, so there exists $i \in [0\,..\,\lfloor\frac{n}{6\alpha}\rfloor)$ such that $q_i \leq \frac{3q}{\lfloor\frac{n}{6\alpha}\rfloor} \leq \frac{36q\alpha}{n}$.

The algorithm A', given strings $X', Y' \in \Sigma^{6\alpha}$, constructs strings $X = \mathtt{a}^{6\alpha i} \cdot X' \cdot \mathtt{a}^{n-6\alpha(i+1)}$ and $Y = \mathtt{a}^{6\alpha i} \cdot Y' \cdot \mathtt{a}^{n-6\alpha(i+1)}$, where $\mathtt{a} \in \Sigma$ is an arbitrary character. Formally, this means that an oracle providing random access to (X', Y') is transformed into an oracle providing random access to (X, Y). Then, A' runs $A^3(X, Y)$, but it terminates the execution (returning an arbitrary answer) on an attempt to make more than $\frac{27}{2}q_i \leq \frac{486q\alpha}{n}$ queries to (X', Y').

This cap of the number of queries trivially bounds the query complexity of A'. As for the correctness, observe that Fact 2.1 implies $\mathsf{ED}(X, Y) = \mathsf{ED}(X', Y')$. Moreover, there is a one-to-one correspondence between the queries of $A'(X', Y')$ and the queries that $A^3(X, Y)$ makes to $X[6\alpha i\,..\,6\alpha(i+1))$ and $Y[6\alpha i\,..\,6\alpha(i+1))$. Hence, it suffices to analyze the error probability of the capped version of A^3. Without the query limit, $A^3(X, Y)$ would in expectation make q_i queries to (X', Y') and err with probability at most $\frac{7}{27}$. By Markov's inequality, the probability of making more than $\frac{27}{2}q_i$ queries to (X', Y') does not exceed $\frac{2}{27}$. Thus, the execution of A^3 is terminated with probability at most $\frac{2}{27}$. Overall, this increases the error probability from $\frac{7}{27}$ to $\frac{7+2}{27} = \frac{1}{3}$. □

References

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *56th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015*, pages 59–78. IEEE, 2015. doi:10.1109/FOCS.2015.14.

[AdlVKK03] Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of MAX-CSPs. *Journal of Computer and System Sciences*, 67(2):212–243, 2003. doi:10.1016/S0022-0000(03)00008-4.

[AHIK13] Alexandr Andoni, Haitham Hassanieh, Piotr Indyk, and Dina Katabi. Shift finding in sublinear time. In *24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 457–465. SIAM, 2013. doi:10.1137/1.9781611973105.33.

[AHWW16] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *48th Annual ACM Symposium on Theory of Computing, STOC 2016*, pages 375–388. ACM, 2016. doi:10.1145/2897518.2897653.

[AKO10] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *51st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 377–386. IEEE, 2010. doi:10.1109/FOCS.2010.43.

[AN10] Alexandr Andoni and Huy L. Nguyen. Near-optimal sublinear time algorithms for ulam distance. In Moses Charikar, editor, *21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 76–86. SIAM, 2010. doi:10.1137/1.9781611973075.8.

[AN20] Alexandr Andoni and Negev Shekel Nosatzki. Edit distance in near-linear time: it's a constant factor. In *61st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2020*, pages 990–1001. IEEE, 2020. doi:10.1109/FOCS46700.2020.00096.

[AO12] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM Journal on Computing*, 41(6):1635–1648, 2012. doi:10.1137/090767182.

[BCFN22a] Karl Bringmann, Alejandro Cassis, Nick Fischer, and Vasileios Nakos. Almost-optimal sublinear-time edit distance in the low distance regime. In *54th Annual ACM Symposium on Theory of Computing, STOC 2022*, pages 1102–1115. ACM, 2022. doi:10.1145/3519935.3519990.

[BCFN22b] Karl Bringmann, Alejandro Cassis, Nick Fischer, and Vasileios Nakos. Improved sublinear-time edit distance for preprocessed strings. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPIcs*, pages 32:1–32:20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.32.

[BCR20] Joshua Brakensiek, Moses Charikar, and Aviad Rubinstein. A simple sublinear algorithm for gap edit distance, 2020. arXiv:2007.14368.

[BEG+21] Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, MohammadTaghi Hajiaghayi, and Saeed Seddighin. Approximating edit distance in truly subquadratic time: Quantum and mapreduce. *Journal of the ACM*, 68(3):19:1–19:41, 2021. doi:10.1145/3456807.

[BEK+03] Tugkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *35th Annual ACM Symposium on Theory of Computing, STOC 2003*, pages 316–324. ACM, 2003. doi:10.1145/780542.780590.

[BES06] Tugkan Batu, Funda Ergün, and Süleyman Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 792–801. ACM, 2006. doi:10.1145/1109557.1109644.

[BI18] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM Journal on Computing*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

[BJKK04] Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004*, pages 550–559. IEEE, 2004. doi:10.1109/FOCS.2004.14.

[BK15] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *56th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015*, pages 79–97. IEEE, 2015. doi:10.1109/focs.2015.15.

[BS79] Jon Louis Bentley and James B. Saxe. Algorithms on vector sets. *SIGACT News*, 11(2):36–39, 1979. doi:10.1145/1008620.1008624.

[BZ16] Djamal Belazzougui and Qin Zhang. Edit distance: Sketching, streaming, and document exchange. In *57th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2016*, pages 51–60. IEEE, 2016. doi:10.1109/FOCS.2016.15.

[CDG+20] Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucký, and Michael E. Saks. Approximating edit distance within constant factor in truly sub-quadratic time. *Journal of the ACM*, 67(6):36:1–36:22, 2020. doi:10.1145/3422823.

[CFH+21] Kuan Cheng, Alireza Farhadi, MohammadTaghi Hajiaghayi, Zhengzhong Jin, Xin Li, Aviad Rubinstein, Saeed Seddighin, and Yu Zheng. Streaming and small space approximation algorithms for edit distance and longest common subsequence. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 54:1–54:20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.54.

[CGK16] Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *48th Annual ACM Symposium on Theory of Computing, STOC 2016*, pages 712–725. ACM, 2016. doi:10.1145/2897518.2897577.

[EJ15] Funda Ergün and Hossein Jowhari. On the monotonicity of a data stream. *Combinatorica*, 35(6):641–653, 2015. doi:10.1007/s00493-014-3035-1.

[GG10] Anna Gál and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM Journal on Computing*, 39(8):3463–3479, 2010. doi:10.1137/090770801.

[GJKK07] Parikshit Gopalan, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, pages 318–327. SIAM, 2007. URL: https://dl.acm.org/doi/10.5555/1283383.1283417.

[GKS19] Elazar Goldenberg, Robert Krauthgamer, and Barna Saha. Sublinear algorithms for gap edit distance. In *60th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2019*, pages 1101–1120. IEEE, 2019. doi:10.1109/FOCS.2019.00070.

[GRS20] Elazar Goldenberg, Aviad Rubinstein, and Barna Saha. Does preprocessing help in fast sequence comparisons? In *52nd Annual ACM Symposium on Theory of Computing, STOC 2020*, pages 657–670. ACM, 2020. doi:10.1145/3357713.3384300.

[HIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012. doi:10.4086/toc.2012.v008a014.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Comput*, 30(2):457–474, 2000. doi:10.1137/S0097539798347177.

[KS20] Tomasz Kociumaka and Barna Saha. Sublinear-time algorithms for computing & embedding gap edit distance. In *61st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2020*, pages 1168–1179. IEEE, 2020. doi:10.1109/focs46700.2020.00112.

[LV88] Gad M. Landau and Uzi Vishkin. Fast string matching with k differences. *Journal of Computer and System Sciences*, 37(1):63–78, 1988. doi:10.1016/0022-0000(88)90045-1.

[MS21] Michael Mitzenmacher and Saeed Seddighin. Improved sublinear time algorithm for longest increasing subsequence. In *32nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1934–1947. SIAM, 2021. URL: https://doi.org/10.1137/1.9781611976465.115.

[NSS17] Timothy Naumovitz, Michael E. Saks, and C. Seshadhri. Accurate and nearly optimal sublinear approximations to Ulam distance. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2012–2031. SIAM, 2017. doi:10.1137/1.9781611974782.131.

[SS13] Michael E. Saks and C. Seshadhri. Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance. In *24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1698–1709. SIAM, 2013. doi:10.1137/1.9781611973105.122.

[SS17] Michael Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. *SIAM Journal on Computing*, 46(2):774–823, 2017. doi:10.1137/130942152.