A faster 3-approximation for sublinear edit distances

Emily Fox Ethan Mader Kent Quanrud Borna Tavasoli Jihan Wang

Abstract

We study the problem of approximating the edit distance between two strings of length n in strongly subquadratic time. [CDG+20] gave the first constant-factor approximation, in $\tilde{O}(n^{12/7})$ randomized time. This was the first constant factor approximation to run in strongly subquadratic time. [And20; GRS20] improved the approximation factor to $(3+\varepsilon)$, with the algorithm in [GRS20] running in $\tilde{O}(n^{8/5+\varepsilon}\operatorname{poly}(1/\varepsilon))$ randomized time. Building on previous work [CDG+20; And20; GRS20], we present a $(3+\varepsilon)$-approximation algorithm running in $\tilde{O}\Big(n^{6/5}\mathrm{OPT}^{2/5}\Big)$ time, where OPT is the optimum edit distance.

1 Introduction

The edit distance (often called Levenshtein distance [Lev65]) between two strings X and Y is the minimum number of character insertions, deletions, and substitutions required to transform X into Y. It can be computed in $O(n^2)$ time using a dynamic programming algorithm [WF74], and that time has been reduced to $O(n^2/\log n)$ [MP80]. When the edit distance OPT is small, these times can be improved to $O(n\mathrm{OPT})$ [Ukk85; Mye86] or even $\tilde{O}\Big(n+\mathrm{OPT}^2\Big)$ [Mye86]. Substantial improvements are unlikely, as Backurs and Indyk [BI15] showed no strongly subquadratic time algorithm exists under the strong exponential time hypothesis (SETH).

The $\tilde{O}\Big(n+\mathrm{OPT}^2\Big)$ time algorithm [Mye86] immediately yields an $O(n)$-time $\sqrt{n}$-approximation algorithm as well. The authors of [BJK+04] obtained an $O\Big(n^{3/7}\Big)$-approximation running in $\tilde{O}(n)$ time[1] (see also [CH02],) and Andoni, Krauthgamer, and Onak [AKO10] obtained a $\log^{O(1/\varepsilon)}(n)$-approximation running in time $n^{1+\varepsilon}$. The authors of [BEG+21] obtained a $(3+\varepsilon)$-approximation running in $\tilde{O}\Big(n^{2-4/21}\Big)$ time in a quantum model of computation, and a $(1+\varepsilon)$-approximation in logarithmically many rounds in the MapReduce model.

A recent breakthrough by the authors of [CDG+20] resulted in the first constant-factor approximation in truly-subquadratic time. Two independent follow-up works [And20; GRS20] reduce the approximation factor to essentially 3. For inputs where $\mathrm{OPT}=\Omega(n)$, Andoni [And20] obtained a $(3+\varepsilon)$-approximation that runs in $\tilde{O}_\varepsilon\Big(n^{8/5}\Big)$ time. Goldenberg, Rubinstein, and Saha [GRS20] obtained a $(3+\varepsilon)$-approximation running in $n^{8/5+o_\varepsilon(1)}$ time for all values of OPT. We discuss the techniques of [CDG+20; And20; GRS20] in Section 2 below. Other subquadratic approximation algorithms include an $n^{1+\varepsilon}$-time $2^{2^{O(1/\varepsilon)}}$-approximation [AN20] and an $n^2/2^{\log^{\Omega(1)}(n)}$-time $(1+\varepsilon)$-approximation [MR26].

[1] We use $O_\varepsilon(\cdots)$ to hide ε factors, $\tilde{O}(\cdots)$ to hide $\log(n)$ factors, and $\tilde{O}_\varepsilon(\cdots)$ to hide $(\varepsilon,\log(n))$ factors in asymptotic bounds.

Our contribution We present a faster algorithm to obtain a $(3+\varepsilon)$-approximation of the edit distance between two strings where $\mathrm{OPT} = o(n)$. Our presentation and analysis is self-contained, and they unify ideas from [And20; GRS20]. Specifically, we show the following theorem.

Theorem 1 (Main Theorem)**.** *A $(3+\varepsilon)$-approximation to the edit distance between two strings of total length n can be computed with high probability in $\tilde{O}\Big(n^{6/5}\mathrm{OPT}^{2/5}/\varepsilon^{19/5}\Big)$ randomized time.*

2 High-level overview

Let X and Y be two strings of total length $n = |X| + |Y|$. The quadratic running time of the standard dynamic programming algorithm for edit distance reflects an all-to-all comparison between every character of X and every character of Y. Recent subquadratic-time approximation algorithms avoid making all n^2 comparisons by leveraging underlying geometric structure [CDG+20; BEG+21; And20; GRS20]. Our algorithm, described below, combines ideas from these preceding works.

By standard techniques, we assume we know OPT up to a constant factor. The starting point, following [BEG+21; CDG+20], shrinks the $|X| \times |Y|$ dynamic programming table by "chunking" the inputs to form a coarser table. For a parameter $d \in [n]$ to be determined, we partition X into $|X|/d$ consecutive substrings $\mathcal{X}$ of length d. (Here, a "substring" entails both its alphabetic content and its coordinates in the original string.) Any edit sequence converting X to Y implicitly edits each of these substrings in $\mathcal{X}$ to a disjoint substring of Y, so that the concatenation of the piecemeal edits recovers Y. In this sense, we can say that each $x \in \mathcal{X}$ is "matched" to some (unknown) substring of Y, and the sum of edit distances from each substring of X to its matched substring of Y gives the total edit distance from X to Y. Of course, the matched substrings of Y need not partition Y symmetrically into substrings of length d. To approximate these target matches, we construct a larger family $\mathcal{Y}$ of substrings of Y. We want this family to be as small as possible but still "cover" all potential matched substrings up to a small loss in approximation. The substrings in $\mathcal{Y}$ vary by length and are overlapping. Giving up $(1+\varepsilon)$-factors in the approximation ratio along the way, it suffices to consider a set of $O(\log(n)/\varepsilon^2)$ lengths close to d. The substrings are overlapping and generated at fixed strides of length at least $\varepsilon^2\mathrm{OPT}d/8n$. We also include empty substrings (located regularly at the same stride) in $\mathcal{Y}$. Ultimately, $\mathcal{X}$ has size $|X|/d$ and $\mathcal{Y}$ has size $O\Big(\frac{n^2}{\varepsilon^4\mathrm{OPT}d}\Big)$.

With a $(1+\varepsilon)$-factor loss in approximation quality, we now have a partition $\mathcal{X}$ of X, a family of substrings $\mathcal{Y}$ covering Y, and an unknown mapping $\mu : \mathcal{X} \to \mathcal{Y}$ such that the concatenation of $\mu(x)$ over $x \in \mathcal{X}$ essentially recovers Y. μ is *monotone*. That is, if x precedes x' in X, then $\mu(x)$ precedes $\mu(x')$ in Y. [And20; GRS20] showed that μ can also be assumed to be (essentially) *smooth*: If x is located close to x' in X, then $\mu(x)$ is located close to $\mu(x')$. ([And20] calls this property *Lipschitz*, while [GRS20] calls it *skewness*.) Finally, μ is *banded*, in the sense that the coordinates of x in X must be within OPT of the coordinates of μ in Y. [GRS20] applied bandedness to restrict the coarse DP table, while we push the idea to apply throughout our algorithm.

We say that two substrings are Δ-*local* if their coordinates are within Δ of each other, and the Δ-*neighborhood* of a substring consists of all Δ-local substrings. If we knew the edit distance between every $x \in \mathcal{X}$ and every OPT-local $y \in \mathcal{Y}$, then we could compute μ by a dynamic program with $|\mathcal{X}| \times (\mathrm{OPT}/n)|\mathcal{Y}| = O(n^2/\varepsilon^4 d^2)$ states. This dynamic program can be interpreted as a lossy compression of the original $|X| \times |Y|$ dynamic program.

Computing the edit distances between all OPT-local pairs in $\mathcal{X}$ and $\mathcal{Y}$ directly leads to the overall running time still being quadratic. However, we have only given up $(1+\varepsilon)$-error to this point. The more interesting part of the algorithm, described below, essentially obtains $(3+\varepsilon)$-approximations

for these OPT-local pairs substantially faster. Before describing the algorithm, we note two helpful relaxations for the task at hand. First, we don't really need to approximate the edit distance to *every* OPT-local pair, as long as we can ensure that we cover every pair matched by μ. (Here, the locality of μ will allow us to restrict our attention to fewer pairs.) Second, we don't have to obtain a $(3+\varepsilon)$-approximation uniformly for every pair matched by μ, as long as we have a $(3+\varepsilon)$-approximation in the aggregate. (There may be some pairs that we fail to get a good approximation for, but we will be able to charge off the error elsewhere.)

We approximate the distances between $\mathcal{X}$ and $\mathcal{Y}$ in iterations parameterized by some $\tau \in [0,1]$. The first iteration starts with $\tau \approx \mathrm{OPT}/n$ and each iteration increases τ by a $(1+\varepsilon)$-factor. Call a match $(x, \mu(x))$ *τ-close* if x and $\mu(x)$ have edit distance at most τd. The goal of an iteration with a fixed τ is to obtain the edit distance for (essentially) all τ-close matches. Within this iteration, we only ever compute edit distances up to $O(\tau d)$, in $O(\tau^2 d^2)$ time.

Let k be a parameter to be determined. We say that a substring $x \in \mathcal{X}$ is *dense* (resp., *approximately dense*) if the τ-close OPT-local substrings in $\mathcal{Y}$ occupy at least k (resp., $k/4$) distinct start cuts. We say x is sparse if they occupy fewer than k start cuts.

Suppose we sample each start cut with probability $\Omega(\log(n)/k)$ and keep every substring beginning at a sampled cut. For each sampled substring y_j, we directly compute all τ-close x-substrings and y-substrings in its OPT-neighborhood. Call $x_i \in \mathcal{X}$ *covered* if it has $\Omega(\log(n))$ neighbors and *uncovered* otherwise. With high probability, all dense x_i are covered, all covered intervals are approximately dense, and all uncovered x_i are sparse. For every dense x_i, the distance from x_i to its closest sample y_j, plus the distance from y_j to its match $\mu(x_i)$, gives a 3-approximation to the distance from x_i to $\mu(x_i)$. This follows from the triangle inequality [BEG+21; CDG+20]. For a fixed τ, this dense phase takes $\tilde{O}(n\mathrm{OPT}/\varepsilon^2 k)$ time.

Consider the remaining set of uncovered strings. These strings have very few τ-close Y-strings in their respective neighborhoods. Here, we have two cases. If a $\mathrm{poly}(\varepsilon/\log n)$-fraction of the uncovered strings are matched by μ with distance τd, then a uniform sample of $O(\mathrm{poly}(\log(n)/\varepsilon))$ uncovered strings will identify at least one such string x_i. Such a string x_i has τ-close neighbors at fewer than k start cuts, which we compute directly. By locality, any other uncovered string x_i' located near x_i has its match near x_i. This leads to a divide-and-conquer strategy following [And20], where we divide on the sparse uncovered substrings, and use the τ-close neighbors at $\tilde{O}(k)$ start cuts from the samples of one iteration to localize the edit distance computations in the next iteration. For a fixed τ, this sparse phase takes $\tilde{O}(nk\tau d/\varepsilon^3)$ time.

Otherwise, fewer than a $\mathrm{poly}(\varepsilon/\log n)$-fraction of the uncovered strings x_i have distance τd to their match $\mu(x_i)$. In turn, these strings represent a $\mathrm{poly}(\varepsilon/\log n)$-fraction of the total cost of the uncovered strings. As observed by [GRS20], we can afford to "miss" them in this iteration, charging the error to the vast majority of uncovered strings of greater cost, which are matched in subsequent iterations with higher values of τ.

Balancing the running times at $k = \sqrt{\varepsilon \mathrm{OPT}/\tau d}$, and over all values of τ, gives a total running time of $\tilde{O}\Big(n\sqrt{\mathrm{OPT}\, d}/\varepsilon^{7/2}\Big)$. The sets of distances computed across τ are collated in a table $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \to \mathbb{Z}_{\geq 0}$. The table $\mathcal{E}$ forms the input data to a dynamic program, and building the table and solving the program takes $\tilde{O}(n^2/\varepsilon^5 d^2)$ time. Balancing terms at $d = n^{2/5}/\varepsilon^{3/5}\mathrm{OPT}^{1/5}$ gives the overall running time of $\tilde{O}\Big(n^{6/5}\mathrm{OPT}^{2/5}/\varepsilon^{19/5}\Big)$.

Summary of novel contributions. At a high level, our algorithm here is a novel amalgamation of techniques spread throughout prior work [CDG+20; GRS20; And20]. The technique of sampling

to cover the dense substrings and applying the triangle inequality in the analysis is from [BEG+21; CDG+20]. The decomposition between dense and sparse substrings, and the locality-based divide-and-conquer framework for sparse uncovered substrings, is from [And20]. ([GRS20] also leverages locality with a different design.) The technique of iterating by distance scale τ, so that one can use faster small edit-distance computations along the way, is from [GRS20]. On top of these previous techniques, we apply bandedness aggressively to limit the edit distance computations throughout all stages of the algorithm. Doing so correctly requires a fair amount of technical work including a new charging scheme for bounding the time spent working at various distance scales τ.

Remark. Andoni [And20] gives an algorithm for general values of OPT with approximation bounds that include an additive term. However, a few small adjustments turn it into a proper multiplicative approximation. First, integrating bandedness into [And20] cancels out the additive factor β for $\beta = \varepsilon\mathrm{OPT}$, leaving a $(3+\varepsilon)$-approximation for all OPT. Second, the text-searching step in [And20], where a substring x of X is compared to a sequence of overlapping substrings of Y, can be accelerated by a known reduction to planar multiple-source shortest paths (MSSP) [Kle05] (see [CKW23]). These two ideas bring [And20] to a $(3+\varepsilon)$-approximation running in $\tilde{O}_\varepsilon\Big(n^{8/5}\Big)$ randomized time.

3 Preliminaries

Let $\mathrm{ED}(x,y)$ denote the edit distance of two strings x, y. Let $X, Y \in \Sigma^*$ be two input strings of total length $n = |X| + |Y|$, let $\mathrm{OPT} = \mathrm{ED}(X,Y)$ be their true edit distance, and let $\varepsilon \in (0,1)$ be fixed. The two strings need not have the same length; we use n as an upper bound on both. We assume that $\mathrm{OPT} \geq n^{3/4}$, since otherwise the claimed running time follows from [Mye86].

The core algorithm is parameterized by a parameter Δ, which approximates OPT. The rest of the article proves the following bounds parameterized by Δ.

Lemma 1. *There exists an algorithm that takes as input two strings $X, Y \in \Sigma^*$ of total length n, $\varepsilon \in (0,1)$, and a value $\Delta \in [n]$ and returns a value λ such that $\mathrm{OPT} \leq \lambda$. If $\mathrm{OPT} \leq \Delta \leq 2\mathrm{OPT}$, then with high probability, $\lambda \leq (3+\varepsilon)\mathrm{OPT}$. The algorithm runs in $\tilde{O}\Big(n^{6/5}\Delta^{2/5}/\varepsilon^{19/5}\Big)$ randomized time.*

Theorem 1 follows from Lemma 1 with $O(\log(n))$ overhead by enumerating Δ in powers of 2 and stopping when $\lambda \leq (3+\varepsilon)\Delta$. Henceforth we assume $\mathrm{OPT} \leq \Delta \leq 2\mathrm{OPT}$.

Inspired by [GRS20], the algorithm is an iterative algorithm where each iteration operates at a particular "error scale" τ_i. For $i \in \mathbb{Z}_{\geq 0}$, let

$$\tau_i \stackrel{\text{def}}{=} \varepsilon(1+\varepsilon)^i \frac{\Delta}{n}.$$

Let $\mathcal{I} = \{i \in \mathbb{Z}_{\geq 0} : \tau_i < 1\}$. Loosely speaking, the ith iteration (starting from $i = 0$) will restrict its edit distance computations where the edit distance is at most τ_i per character. We round the geometric sequence down at the top so that the maximum scale is 1 exactly; the algorithm stops before τ_i crosses 1.

As observed in [GRS20], with nearly linear randomized time preprocessing of X and Y separately, one can compute the edit distance between two substrings u and v of (either) X and Y with high probability in time $\tilde{O}\Big(\mathrm{ED}^2(u,v)\Big)$. This is used to speed up edit distance computations within

each distance scale i. The same bound is attained by extending furthest-reaching paths along the diagonals of the edit distance table, as in [Mye86], using longest-common-extension queries on the preprocessed strings. Since values along a diagonal never decrease, a single run with threshold K from two start positions gives, in $\tilde{O}(K^2)$ time, the edit distance between every pair of prefixes of the two suffixes that is at most K. We therefore charge one computation per pair of start cuts, covering every length at once.

Like previous work [CDG+20; And20; GRS20], our algorithm works with families $\mathcal{X}$ and $\mathcal{Y}$ of substrings of X and Y respectively. For two indices $i \le j$ of the string in question, let $X_{i:j}$ (resp. $Y_{i:j}$) denote the substring of X (resp. Y) from index i (inclusive) to index j (exclusive). A substring encompasses both its alphabetic contents and its coordinates in the original string. We use "distance" to refer to the edit distance between substrings, and "locality" to refer to the distance between the underlying coordinates. Each insertion or deletion changes $i - j$ by one at unit cost, so every edit sequence of cost at most Δ stays within the band $|i - j| \le \Delta$, and its final vertex $(|X|, |Y|)$ lies in that band because $\mathrm{OPT} \ge ||X| - |Y||$.

Let $d \in [n]$ be a parameter to be determined with $d = o(n)$. Without loss of generality we assume that d divides $|X|$: padding X with fewer than d dummy characters changes OPT by at most d, which is $o(\varepsilon \mathrm{OPT})$ for our choice of d and the assumption $\mathrm{OPT} \ge n^{3/4}$. Let $\mathcal{X}$ partition X into contiguous substrings of length d; i.e., $\mathcal{X} = \{X_{1:d+1}, X_{d+1:2d+1}, \ldots\}$. $\mathcal{X}$ has $R \stackrel{\text{def}}{=} |X|/d \le n/d$ substrings.

We break up Y into overlapping substrings with varying lengths and strides, depending on the scale $i \in \mathcal{I}$. For $i \in \mathcal{I}$, let $\gamma_i = \lfloor \varepsilon \tau_i d/4 \rfloor$. Let $G_i = \{0, |Y|\} \cup \{j\gamma_i : j \in \mathbb{Z}_{\ge 0},\ 0 \le j\gamma_i \le |Y|\}$ be the grid with stride γ_i, or $G_i = \{0, 1, \ldots, |Y|\}$ if $\gamma_i = 0$. Let $\mathcal{Y}'_i$ be the family of all substrings of Y with endpoints in G_i and length within $\tau_i d$ of d combined with empty strings located in G_i. Formally,

$$\mathcal{Y}'_i = \{Y_{s:t+1} : s, t \in G_i,\ s < t,\ |t + 1 - s - d| \le \tau_i d\} \cup \{Y_{s:s} : s \in G_i\}.$$

In either case G_i has $O(n/\varepsilon\tau_i d)$ points, and an interval of length $2\tau_i d$ holds $O(1/\varepsilon)$ of them, so each start cut admits that many lengths. Hence $\mathcal{Y}'_i$ has $O(n/\varepsilon^2\tau_i d)$ substrings. For $i \in \mathcal{I}$, let $\mathcal{Y}_i = \mathcal{Y}'_i$ be the set of Y-substrings at level i. Let $\mathcal{Y} = \bigcup_i \mathcal{Y}'_i$.

$\mathcal{X}$ chunks X into R pieces. In the optimum (X, Y)-edit sequence, each $x \in \mathcal{X}$ is edited into a substring of Y that together cover Y. Intuitively, $\mathcal{Y}$ approximates these unknown target substrings of Y. The algorithm will try to assemble a mapping $\mu : \mathcal{X} \to \mathcal{Y}$ such that the concatenation of $\mu(x)$ over $x \in \mathcal{X}$ is very close to Y.

Because of the imprecise chunking on $\mathcal{Y}$, the concatenation won't be exact. Two matched substrings $\mu(x), \mu(x')$ can overlap, and there may be some indices that aren't covered at all. For a mapping $\mu : \mathcal{X} \to \mathcal{Y}$, the *cost* of μ is defined as

$$\mathrm{cost}(\mu) = \sum_{i=1}^{R} \mathrm{ED}(x_i, \mu(x_i)) + \mathrm{s}(\mu(x_1)) + \sum_{i=2}^{R} |\mathrm{s}(\mu(x_i)) - \mathrm{e}(\mu(x_{i-1}))| + (|Y| - \mathrm{e}(\mu(x_R))),$$

where $x_1, \ldots, x_R$ enumerates the substrings in $\mathcal{X}$ in sequential order, $\mathrm{s}(y)$ is the starting index of a substring y, and $\mathrm{e}(y)$ is the ending index (exclusive) of y.$\mathrm{cost}(\mu)$ is the cost to first edit each x_i into $\mu(x_i)$, and then edit the concatenation $\mu(x_1), \mu(x_2), \ldots$ into Y by simply deleting overlapping characters and inserting omitted letters. In particular it is easy to compute given the piecewise edit distances $\mathrm{ED}(x_i, \mu(x_i))$. We have

$$\mathrm{ED}(X, Y) \le \sum_{i=1}^{R} \mathrm{ED}(x_i, \mu(x_i)) + \mathrm{ED}(Y', Y) \le \mathrm{cost}(\mu)$$

where $Y' = \mu(x_1)\mu(x_2)\cdots$ concatenates the individual substring matches.

We compete with an unknown mapping $\mu : \mathcal{X} \to \mathcal{Y}$, derived from the optimal edit sequence, with certain geometric properties with respect to the underlying coordinates. We say μ is *monotone* if for any two $x, x' \in \mathcal{X}$ with x preceding x', $\mu(x)$ ends before $\mu(x')$ begins. For a constant $c \geq 4$, μ is *smooth* if for any two $x, x' \in \mathcal{X}$ with nonempty $\mu(x), \mu(x')$, we have $|\mathrm{s}(\mu(x)) - \mathrm{s}(\mu(x'))| \leq c|\mathrm{s}(x) - \mathrm{s}(x')|$; i.e., μ is c-Lipschitz with respect to the underlying starting indices. Lastly, for a parameter c, we say μ is *c-banded* if $|\mathrm{s}(\mu(x)) - \mathrm{s}(x)| \leq c$ for all $x \in \mathcal{X}$.

As observed by [And20; GRS20], there exists a monotone, smooth, and $O(\mathrm{OPT})$-banded μ with cost close to OPT. Our analysis also employs maps $\mu_i : \mathcal{X} \to \mathcal{Y}_i$ for each $i \in \mathcal{I}$ that are quantifiably similar to μ but map into the set of substrings $\mathcal{Y}_i$ considered by the algorithm at distance scale i.

Lemma 2. *Let $\Delta \in [\mathrm{OPT}, 2\mathrm{OPT}]$. There exist a matching $\mu : \mathcal{X} \to \mathcal{Y}$ and, for each $i \in \mathcal{I}$, a matching $\mu_i : \mathcal{X} \to \mathcal{Y}_i$, with the following properties. Letting $\mathcal{X}_\emptyset \subseteq \mathcal{X}$ be the set of substrings mapped to empty substrings:*

(a) μ is monotone with pairwise-disjoint substrings $\mu(x)$, and μ and every μ_i are smooth and $O(\Delta)$-banded on their nonempty mappings.

(b) $(3+\varepsilon)\sum_{x \notin \mathcal{X}_\emptyset} \mathrm{ED}(x, \mu(x)) + d|\mathcal{X}_\emptyset| + \Big(|Y| - \sum_x |\mu(x)|\Big) \leq (3 + O(\varepsilon))\mathrm{OPT}$.

(c) For $x \notin \mathcal{X}_\emptyset$ and $i \in \mathcal{I}$:

(i) If $\mu(x) \in \mathcal{Y}'_i$, then $(1+\varepsilon)\,\mathrm{ED}(x, \mu(x)) \leq \tau_i d \leq (1+\varepsilon)^2\,\mathrm{ED}(x, \mu(x))$.

(ii) If $\mu(x) \in \mathcal{Y}_i$, then $\mu_i(x) = \mu(x)$.

(iii) If $\mu(x) \in \mathcal{Y}'_{i'}$ for some $i' < i$, then $\mathrm{ED}(x, \mu_i(x)) \leq \tau_i d$ and both endpoints of $\mu(x)$ and $\mu_i(x)$ differ by at most $O(\varepsilon\tau_i d)$.

The analysis ultimately appeals property (b) to obtain the final approximation bound. The $d|\mathcal{X}_\emptyset|$ term represents the obvious fact that the edit distance between any $x \in \mathcal{X}$ and an empty string is always d. The $(3+\varepsilon)$-factor comes from the fact that we are only able to obtain (essentially) $(3+\varepsilon)$-multiplicative approximations to the edit distance for general pairs (x, y).

The proof of Lemma 2, which extends similar analyses [And20; GRS20], is given in Appendix A. The smoothness guarantee is slightly stronger than in previous work [And20; GRS20], where smoothness constant was $O(1/\varepsilon)$. This improves the ε-factors in the running time.

4 Single-threshold iteration

As discussed in Section 2, the algorithm proceeds in iterations where the ith iteration operates at distance scale τ_i. In the global scheme, the ith iteration addresses pairs $(x, \mu_i(x))$ with edit distance roughly $\tau_i d$.

At scale $i \in \mathcal{I}$, we only compute edit distances up to $O(\tau_i d)$ between substrings in $\mathcal{X}$ and substrings in $\mathcal{Y}_i$. This restriction speeds up each iteration along two dimensions. First, every edit distance computation takes only $O(\tau_i^2 d^2)$ time, instead of d^2. Second, $\mathcal{Y}_i$ has only $O(n/\varepsilon^2 \tau_i d)$ substrings, and can be much smaller than all of $\mathcal{Y}$. The rest of the section proves the following.

Lemma 3. *Let $i \in \mathcal{I}$. There is a randomized algorithm that computes a set of Δ-local pairs $\mathcal{Q}_i \subseteq \mathcal{X} \times \mathcal{Y}_i$ and approximate distances $d_i : \mathcal{Q}_i \to \mathbb{Z}_{\geq 0}$. The following hold with high probability for any smooth and $O(\Delta)$-banded matching $\mu_i : \mathcal{X} \to \mathcal{Y}_i$:*

(a) For all $(x, y) \in \mathcal{Q}_i$, $d_i(x, y) \geq \mathrm{ED}(x, y)$.

(b) For all $x \in \mathcal{X}$ *with* $(x, \mu_i(x)) \in \mathcal{Q}_i$, $d_i(x, \mu_i(x)) \leq \mathrm{ED}(x, \mu_i(x)) + 2\tau_i d$.

(c) $$\sum_{\substack{x \in \mathcal{X} \\ \mathrm{ED}(x,\mu_i(x)) \leq \tau_i d \\ (x,\mu_i(x)) \notin \mathcal{Q}_i}} \tau_i d \leq \frac{\varepsilon}{\log n} \sum_{\substack{x \in \mathcal{X} \\ \mathrm{ED}(x,\mu_i(x)) > \tau_i d}} \mathrm{ED}(x, \mu_i(x)).$$

The algorithm runs in $\tilde{O}\Big(n\Delta^{1/2}\tau_i^{1/2}d^{1/2}/\varepsilon^{5/2}\Big)$ *randomized time.*

For the rest of this section, fix $i \in \mathcal{I}$ and a smooth and banded matching $\mu_i : \mathcal{X} \to \mathcal{Y}_i$. Call two substrings a, b *close* if $\mathrm{ED}(a, b) \leq \tau_i d$ and *far* otherwise. When $x \in \mathcal{X}$ is close to $\mu_i(x)$, we say x is *good* and $(x, \mu_i(x))$ is a *good match*; otherwise x is *bad* and $(x, \mu_i(x))$ is a *bad match*.

The high-level goal is to obtain a $(3 + \varepsilon)$-approximation to the edit distance for every good x. The algorithm actually obtains $(3+\varepsilon)$-approximation for a randomized set $\mathcal{Q}_i \subseteq \mathcal{X} \times \mathcal{Y}_i$ (property (b) above). While some good matches are left out, with high probability, they only constitute a small fraction of the total cost of μ_i (property (c)).

The high-level strategy, following previous work [And20; GRS20], works on two phases based on a sparse/dense decomposition of $\mathcal{X}$. For a parameter k to be determined, $x \in \mathcal{X}$ is *dense* if the Δ-local $y \in \mathcal{Y}_i$ close to x occupy at least k start cuts, *approximately dense* if they occupy at least $k/4$ start cuts, and *sparse* if it is not dense.

The first phase, called the "dense phase", addresses the good and dense x, and takes advantage of the fact that it is easy to sample substrings close to x. A sampled $y \in \mathcal{Y}_i$ serves as a waypoint (so to speak) to the actual match $\mu_i(x)$. The second phase, called the "sparse phase", leverages the fact that each sparse substring x is only close to a small number of local substrings. By smoothness, the small set of all close $y \in \mathcal{Y}_i$ to some sparse and good x constrains the set of possible matches for any good x' local to x. This leads to a divide-and-conquer scheme where the edit computations are restricted by localization.

Phase 1: Density classification and covering dense substrings by the triangle inequality. Let $C \subseteq \mathcal{Y}_i$ consist of every $y \in \mathcal{Y}_i$ whose start cut is sampled, sampling each start cut independently with probability $\log(n)/k$. We compute $\mathrm{ED}(x, c)$ up to $\tau_i d$ and $\mathrm{ED}(c, y)$ up to $2\tau_i d$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}_i$, and $c \in C$ restricted to Δ-local pairs x, c and x, y.

Let $\mathcal{X}' \subseteq \mathcal{X}$ be the set of $x \in X$ that are close to Δ-local $c \in C$ at $\log(n)/2$ or more start cuts, and $\mathcal{X}'' = \mathcal{X} \setminus \mathcal{X}'$. By standard concentration bounds, every $x \in \mathcal{X}'$ is approximately dense, and every $x \in \mathcal{X}''$ is sparse.

Let $x \in \mathcal{X}'$, and let $c \in C$ minimize $\mathrm{ED}(x, c)$ among Δ-local $c \in C$. Then $\mathrm{ED}(x, c) \leq \tau_i d$. By the triangle inequality,

$$\mathrm{ED}(c, \mu_i(x)) \leq \mathrm{ED}(c, x) + \mathrm{ED}(x, \mu_i(x)) \leq \tau_i d + \mathrm{ED}(x, \mu_i(x)).$$

In turn, $\mathrm{ED}(x, c) + \mathrm{ED}(c, \mu_i(x)) \leq \mathrm{ED}(x, \mu_i(x)) + 2\tau_i d$. We set $d'_i(x, y) = \mathrm{ED}(x, c) + \mathrm{ED}(c, y)$ for all Δ-local $y \in \mathcal{Y}_i$.

Each $x \in \mathcal{X}$ and each start cut of G_i is paired with $\tilde{O}(\Delta/\varepsilon\tau_i dk)$ sampled start cuts in expectation, and each pair takes one $\tilde{O}(\tau_i^2 d^2)$-time computation. Since $|\mathcal{X}|, |G_i| = O(n/\varepsilon\tau_i d)$, we spend a total of $\tilde{O}(n\Delta/\varepsilon^2 k)$ expected time computing edit distances involving C. We store d'_i implicitly, by the minimizing c for each $x \in \mathcal{X}'$ together with the computed distances $\mathrm{ED}(c, y)$, so each entry is

evaluated in constant time when it is read; these reads are charged to building $\mathcal{E}$ in Lemma 6. In summary, we have:

Lemma 4. *There is a randomized algorithm that computes a set $\mathcal{X}' \subseteq \mathcal{X}$ and distances $d_i' : \mathcal{Q}_i' \to \mathbb{Z}_{\geq 0}$ for a set $\mathcal{Q}_i' \subseteq \mathcal{X}' \times \mathcal{Y}_i$ such that, with high probability:*

(a) For all $(x, y) \in \mathcal{Q}_i'$, $d_i'(x, y) \geq \mathrm{ED}(x, y)$.

(b) $\mathcal{X}'$ contains all dense $x \in \mathcal{X}$. $\mathcal{Q}_i'$ contains all $O(\Delta)$-local pairs (x, y) with $x \in \mathcal{X}'$.

(c) For all $x \in \mathcal{X}'$, $(x, \mu_i(x)) \in \mathcal{Q}_i'$ and

$$d_i'(x, \mu_i(x)) \leq \mathrm{ED}(x, \mu_i(x)) + 2\tau_i d.$$

The algorithm runs in $\tilde{O}(\Delta n/\varepsilon^2 k)$ randomized time.

Phase 2: Divide-and-conquer over sparse substrings via locality. After phase 1, with high probability, we have well-approximated the distance of every good match $(x, \mu_i(x))$ with $x \in \mathcal{X}'$. There remains a set $\mathcal{X}'' \subseteq \mathcal{X}$ of sparse substrings of X. Some of these substrings are good and we want to approximate them. The bad sparse substrings add noise to the current iteration.

This phase of the iteration is a divide-and-conquer scheme based on the smoothness of μ and sparsity of the remaining strings, inspired by [And20; GRS20]. To build intuition, let $x \in \mathcal{X}''$ be a good sparse substring. Since x is sparse, the y-substrings close and Δ-local to x occupy fewer than k start cuts. We can identify all these targets by directly computing $\mathrm{ED}(x, y)$ up to $\tau_i d$ against every Δ-local y.

While it is expensive to compare x to every Δ-local y, the computation reveals significant information about the matches of other substrings x' close to x. The Δ-local substrings that can be matched to x occupy fewer than k start cuts because x is sparse. Now consider an $x' \in \mathcal{X}$ adjacent to x. By smoothness, $\mu(x')$ must be close to one of these candidates for x. The smoothness facilitates a divide-and-conquer scheme, where we divide $\mathcal{X}$ based on their coordinates in X, and invoke smoothness to limit the number of edit distance computations in smaller subproblems.

One technical issue is the bad substrings, which are indistinguishable from good substrings and do not provide any helpful locality data. To address this, each subproblem samples a sizable number of sparse substrings and identifies the close potential candidates for each sampled substring. If a single sparse substring is good, that gives all the information we need to localize child subproblems. If all sampled substrings are bad, then the localization invariants for all subproblems are broken. With high probability, however, almost all of the sparse substrings must be bad. Bad substrings are matched with greater edit distance than good substrings, so the good substrings represent a negligible fraction of the total edit distance cost. As observed by [GRS20], in the grand scheme of things, we can afford to "miss" a few of the good substrings and fall back on suboptimal matches in later iterations.

Lemma 5. *Let $\mathcal{X}'' \subseteq \mathcal{X}$ be a set of sparse substrings. There is a randomized algorithm that computes distances $d_i'' : \mathcal{Q}_i'' \to \mathbb{Z}_{\geq 0}$ for a set $\mathcal{Q}_i'' \subseteq \mathcal{X}'' \times \mathcal{Y}_i$ such that with high probability:*

(a) If $(x, y) \in \mathcal{Q}_i''$, then $d_i''(x, y) = \mathrm{ED}(x, y)$.

(b)
$$\sum_{\substack{good\ x \in \mathcal{X}'' \\ (x, \mu_i(x)) \notin \mathcal{Q}_i''}} \tau_i d \leq \frac{\varepsilon}{\log(n)} \sum_{bad\ x \in \mathcal{X}''} \mathrm{ED}(x, \mu_i(x)).$$

The algorithm runs in $\tilde{O}(nk\tau_i d/\varepsilon^3)$ randomized time.

Proof. Each subproblem in the divide-and-conquer scheme is defined by a dyadic interval $I \subseteq [|X|]$ and a subset of Y substrings $\mathcal{Y}_{i,I} \subseteq \mathcal{Y}_i$. Let $\mathcal{X}''_I$ be the subset of sparse substrings whose starting coordinates are contained in I. We operate under the assumption that all good $x \in \mathcal{X}''_I$ have $\mu_i(x) \in \mathcal{Y}_{i,I}$. We will always have $|\mathcal{Y}_{i,I}| \leq \tilde{O}\Big(\frac{k|I|}{\varepsilon^3\tau_i d}\Big)$. In the root subproblem, $I = [|X|]$, $\mathcal{Y}_{i,I} = \mathcal{Y}_i$, and $\mathcal{X}''_I = \mathcal{X}''$.

In the base case, when I has size $O(d)$, we just directly compute $\mathrm{ED}(x, y)$ for all $x \in \mathcal{X}''_I$ and $y \in \mathcal{Y}_{i,I}$.

In the general setting, let I_1, I_2 be the dyadic intervals splitting I in half. For $j \in \{1, 2\}$, let $S_j \subseteq \mathcal{X}''_{I_j}$ sample $O\Big(\log^2(n)/\varepsilon\Big)$ sparse substrings uniformly at random. We compute $\mathrm{ED}(x, y)$ up to $\tau_i d$ between every sampled $x \in S_j$ and every Δ-local $y \in \mathcal{Y}_{i,I}$. To assemble $\mathcal{Y}_{i,I_j}$, first let $T_j \subseteq \mathcal{Y}_{i,I}$ be the set of all $y \in \mathcal{Y}_{i,I}$ that were found to be close to some Δ-local $x \in S_j$. We have T_j occupies at most $k|S_j| = \tilde{O}(k/\varepsilon)$ start cuts because each $x \in S_j$ is sparse. Set $\mathcal{Y}_{i,I_j}$ to be the set of all $y \in \mathcal{Y}_{i,I}$ that are $|I_j|$-local to some $y' \in T_j$. $\mathcal{Y}_{i,I_j}$ has at most $O(k|S_j||I_j|/\varepsilon^2\tau_i d) \leq \tilde{O}(k|I_j|/\varepsilon^3\tau_i d)$ substrings. We recurse on each I_j and $\mathcal{Y}_{i,I_j}$.

In the analysis, we have two cases. If S_j contains a good sparse substring x', then since μ_i is smooth and $\mathcal{Y}_{i,I_j}$ contains all $O(|I_j|)$-local substrings to $\mu_i(x')$, $\mathcal{Y}_{i,I_j}$ contains $\mu_i(x)$ for all good $x \in \mathcal{X}''_{I_j}$. If S_j does not contain any good sparse substrings, then with high probability, at most $O(\varepsilon/\log(n))$-fraction of $\mathcal{X}''_{I_j}$ is good, so at least a $(1 - \varepsilon/\log(n))$-fraction is bad. These bad sparse substrings have $\mathrm{ED}(x, \mu_i(x)) > \tau_i d$. Hence the good substrings' total upper bound is at most an $\varepsilon/\log(n)$-fraction of $\sum_{\text{bad } x \in \mathcal{X}''_{I_j}} \mathrm{ED}(x, \mu_i(x))$.

Ultimately, the branches of the recursion tree either successfully sample good substrings and localize down to the leaf subproblems, or fail at some subtree where the edit distance upper bounds of the good substrings represent an $\varepsilon/\log(n)$-fraction of all the edit distances of substrings in this subtree.

For the running time, there are $O(\log(n))$ levels of the recursion. The ℓth level has 2^ℓ subintervals I of size $|I| = O(n/2^\ell)$. We perform an $O(\tau_i^2 d^2)$-time edit distance computation for each $x \in S_I$ and start cut of $\mathcal{Y}_{i,I}$. The $\tilde{O}(k/\varepsilon)$ start cuts of the parent's T_j give $\mathcal{Y}_{i,I}$ at most $\tilde{O}(k|I|/\varepsilon^2\tau_i d)$ start cuts, so there are $\tilde{O}\Big(\frac{k|I|}{\varepsilon^3\tau_i d}\Big) = \tilde{O}\Big(\frac{kn}{\varepsilon^3 2^\ell \tau_i d}\Big)$ such pairs, since $|S_I| = O\Big(\log^2(n)/\varepsilon\Big)$. Across all sub-intervals and all levels, we perform $\tilde{O}\Big(\frac{kn}{\varepsilon^3\tau_i d}\Big)$ edit distance computations for $\tilde{O}(nk\tau_i d/\varepsilon^3)$ time total. □

Completing the proof of Lemma 3. By Lemma 4, we can compute a randomized partition of $\mathcal{X}$ into $\mathcal{X}'$, $\mathcal{X}''$ with randomized distances $d'_i : \mathcal{Q}'_i \to \mathbb{Z}_{\geq 0}$ for a randomized set $\mathcal{Q}'_i \subseteq \mathcal{X}' \times \mathcal{Y}_i$ in $\tilde{O}\Big(\frac{\Delta n}{\varepsilon^2 k}\Big)$ randomized time. By Lemma 5, we can compute randomized distances $d''_i : \mathcal{Q}''_i \to \mathbb{Z}_{\geq 0}$ for a randomized set $\mathcal{Q}''_i \subseteq \mathcal{X}'' \times \mathcal{Y}_i$ in $\tilde{O}(nk\tau_i d/\varepsilon^3)$ randomized time. Let $\mathcal{Q}_i = \mathcal{Q}'_i \sqcup \mathcal{Q}''_i \subseteq \mathcal{X} \times \mathcal{Y}_i$. Define $d_i : \mathcal{Q}_i \to \mathbb{Z}_{\geq 0}$ by $d_i(x, y) = d'_i(x, y)$ for $x \in \mathcal{X}'$ and $d_i(x, y) = d''_i(x, y)$ for $x \in \mathcal{X}''$.

From the properties of Lemma 4 and Lemma 5, we have $d_i(x, y) \geq \mathrm{ED}(x, y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}_i$, proving (a). For (b), if $x \in \mathcal{X}'$ and $(x, \mu_i(x)) \in \mathcal{Q}_i$, then $d_i(x, \mu_i(x)) \leq \mathrm{ED}(x, \mu_i(x)) + 2\tau_i d$. If $x \in \mathcal{X}''$ and $(x, \mu_i(x)) \in \mathcal{Q}_i$, then $d_i(x, \mu_i(x)) = \mathrm{ED}(x, \mu_i(x))$. For (c), $(x, \mu_i(x)) \in \mathcal{Q}_i$ for all

$x \in \mathcal{X}'$. Thus if $(x, \mu_i(x)) \notin \mathcal{Q}_i$, $x \in \mathcal{X}''$. Thus

$$\sum_{\substack{\text{good } x \in \mathcal{X}'' \\ (x,\mu_i(x)) \notin \mathcal{Q}_i}} \tau_i d \leq \frac{\varepsilon}{\log(n)} \sum_{\text{bad } x \in \mathcal{X}} \text{ED}(x, \mu_i(x)),$$

as desired.

We choose $k = \sqrt{\frac{\varepsilon\Delta}{\tau_i d}}$ to balance the two bounds. This gives us $\tilde{O}\Big(n\Delta^{1/2}\tau_i^{1/2}d^{1/2}/\varepsilon^{5/2}\Big)$ randomized time total, as claimed.

5 The full algorithm and analysis

This algorithm brings together the data collected across the distance scales and completes the algorithm. The key lemma, Lemma 6 below, combines the partial sets of distances $d_i : \mathcal{Q}_i \to \mathbb{Z}_{\geq 0}$ from Lemma 3 across $i \in \mathcal{I}$ and builds a unified table of distance estimates $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \to \mathbb{Z}_{\geq 0}$. Each entry in $\mathcal{E}$ is an overestimate, but in the aggregate it gives an (essentially) 3-approximation over all the pairs $(x, \mu(x))$ in the unknown target matching μ given by Lemma 2. $\mathcal{E}$ provides all the data needed to solve a dynamic programming problem that actually obtains the approximate edit sequence, as in prior work.

Lemma 6. *Let $\mu : \mathcal{X} \to \mathcal{Y}$ and $\mu_i : \mathcal{X} \to \mathcal{Y}_i$ for $i \in \mathcal{I}$ satisfy Lemma 2. For $i \in \mathcal{I}$, let $\mathcal{Q}_i \subseteq \mathcal{X} \times \mathcal{Y}_i$ and $d_i : \mathcal{Q}_i \to \mathbb{Z}_{\geq 0}$ be a randomized set and table satisfying properties of Lemma 3 with respect to μ_i. In $\tilde{O}\Big(\frac{n^2}{\varepsilon^5 d^2}\Big)$ time, one can construct a table $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \to \mathbb{Z}_{\geq 0}$ with the following properties.*

(a) $\mathcal{E}(x, y) \geq \text{ED}(x, y)$ *for all* $x \in \mathcal{X}$ *and* Δ*-local* $y \in \mathcal{Y}$.

(b) $\displaystyle\sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \mathcal{E}(x, \mu(x)) \leq (3 + O(\varepsilon)) \sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \text{ED}(x, \mu(x)) + O(\varepsilon \text{OPT}).$

Proof. The high-level idea is to use the per-scale distance estimates $d_i : \mathcal{Q}_i \to \mathbb{Z}_{\geq 0}$ to fill out $\mathcal{E}(x, y)$. These distances are good approximations, but they are not exhaustive, with some matched pairs $(x, \mu_i(x))$ omitted from $\mathcal{Q}_i$ at their appropriate scales i. The missed pairs $(x, \mu_i(x))$ at a scale i represent a small fraction of the total edit distance, and they are recovered (in the aggregate) at larger scales i via small perturbations of $\mu(x)$ that we call "shifts".

Formally, for each $y \in \mathcal{Y}$ we define the shifts $S(y)$ to be the set of $y' \in \mathcal{Y}_i$ for all $i \in \mathcal{I}$ such that both endpoints of y and y' differ by $O(\varepsilon\tau_i d)$. Then $|S(y)| = O(|\mathcal{I}|) = O(\log(n)/\varepsilon)$. Observe that for all $x \in \mathcal{X}$ and $i \in \mathcal{I}$, $\mu_i(x) \in S(\mu(x))$.

For all $x \in \mathcal{X}$ and Δ-local $y \in \mathcal{Y}$, we initialize $\mathcal{E}(x, y) = \max\{|x|, |y|\}$. For all $i \in \mathcal{I}$ and $(x, y) \in \mathcal{Q}_i$, we set $\mathcal{E}(x, y) = d_i(x, y)$. Then, for all Δ-local pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and all $y' \in S(y)$, we set

$$\mathcal{E}(x, y) = \min\{\mathcal{E}(x, y), d_{i'}(x, y') + |\text{s}(y) - \text{s}(y')| + |\text{e}(y) - \text{e}(y')|\}.$$

Here, the difference in endpoints in the RHS overestimates $\text{ED}(y, y')$, and by the triangle inequality, the whole quantity still overestimates $\text{ED}(x, y)$.

Clearly $\mathcal{E}$ overestimates the actual edit distances, per property (a). It remains to prove property (b).

For each iteration $i \in \mathcal{I}$, let

$$\mathcal{X}_i' = \{x \in \mathcal{X} : \operatorname{ED}(x, \mu_i(x)) \le \tau_i d,\ (x, \mu_i(x)) \notin \mathcal{Q}_i\}$$

be the set of "good" substrings "missed" at scale i. For $i \in \mathcal{I}$, noting that $\mu_i(x) = \mu(x)$ whenever $\operatorname{ED}(x, \mu_i(x)) > \tau_i d$, property (c) of Lemma 3 implies that

$$\sum_{x \in \mathcal{X}_i'} \tau_i d \le \frac{\varepsilon}{\log n} \sum_{x \in \mathcal{X}} \operatorname{ED}(x, \mu(x)) \le \frac{\varepsilon}{\log n} \operatorname{OPT}.$$

We claim that

$$\mathcal{E}(x, \mu(x)) \le (3 + O(\varepsilon)) \operatorname{ED}(x, \mu(x)) + O\left(\frac{\varepsilon \Delta d}{n}\right) + O\left(\varepsilon \sum_{i:\, x \in \mathcal{X}_i'} \tau_i d\right) \tag{1}$$

for all $x \in \mathcal{X} \setminus \mathcal{X}_\emptyset$, where we recall that $\mathcal{X}_\emptyset$ is the subset of strings that are mapped to empty strings. If Equation (1) holds, then summing over all $x \in \mathcal{X} \setminus \mathcal{X}_\emptyset$, we have

$$\begin{aligned}
\sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \mathcal{E}(x, \mu(x)) &\le (3 + O(\varepsilon)) \sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \operatorname{ED}(x, \mu(x)) + O\left(\frac{\varepsilon \Delta d |\mathcal{X}|}{n}\right) + O\left(\varepsilon \sum_{x \in \mathcal{X}} \sum_{i:\, x \in \mathcal{X}_i'} \tau_i d\right) \\
&= (3 + O(\varepsilon)) \sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \operatorname{ED}(x, \mu(x)) + O(\varepsilon \operatorname{OPT}) + O\left(\varepsilon \sum_{i \in \mathcal{I}} \sum_{x \in \mathcal{X}_i'} \tau_i d\right) \\
&\le (3 + O(\varepsilon)) \sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \operatorname{ED}(x, \mu(x)) + O(\varepsilon \operatorname{OPT}) + O\left(\frac{\varepsilon^2 |\mathcal{I}|}{\log n} \operatorname{OPT}\right) \\
&\le (3 + O(\varepsilon)) \sum_{x \in \mathcal{X} \setminus \mathcal{X}_\emptyset} \operatorname{ED}(x, \mu(x)) + O(\varepsilon \operatorname{OPT}),
\end{aligned}$$

as desired.

To prove Equation (1), let $x \in \mathcal{X} \setminus \mathcal{X}_\emptyset$, and let i be the unique index such that $\mu(x) \in \mathcal{Y}_i'$. Then $(1 + \varepsilon) \operatorname{ED}(x, \mu(x)) \le \tau_i d \le (1 + \varepsilon)^2 \operatorname{ED}(x, \mu(x)) + \varepsilon \Delta d / n$ and $\mu(x) = \mu_i(x)$ by subproperties (c)(i) and (c)(ii) of Lemma 2. For $j \ge i$, we have $(x, \mu_j(x)) \notin \mathcal{Q}_j$ iff $x \in \mathcal{X}_j'$.

We have two cases depending on whether $(x, \mu_j(x)) \in \mathcal{Q}_j$ for some $j \ge i$. Suppose this is the case, and let j be the first such index. We have

$$\begin{aligned}
\mathcal{E}(x, \mu(x)) &\le d_j(x, \mu_j(x)) + 2\varepsilon \tau_j d \\
&\le \operatorname{ED}(x, \mu_j(x)) + (2 + O(\varepsilon)) \tau_j d \\
&\le \operatorname{ED}(x, \mu(x)) + (2 + O(\varepsilon)) \tau_j d
\end{aligned}$$

by property (b) of Lemma 3 and subproperty (c)(iii) of Lemma 2. When $j > i$,

$$\sum_{\ell=i}^{j-1} \tau_\ell = \tau_0 \sum_{\ell=i}^{j-1} (1 + \varepsilon)^\ell = \tau_0 \frac{(1 + \varepsilon)^j - (1 + \varepsilon)^i}{\varepsilon} = \frac{\tau_j - \tau_i}{\varepsilon}.$$

Rearranging,

$$\tau_j \le \tau_i + \varepsilon \sum_{\ell=i}^{j-1} \tau_\ell \le \tau_i + \varepsilon \sum_{\ell:\, x \in \mathcal{X}_\ell'} \tau_\ell. \tag{2}$$

Equation (2) also holds for $j = i$. Plugging back in, we have

$$\begin{aligned}\mathcal{E}(x,\mu(x)) &\le \mathrm{ED}(x,\mu(x)) + (2+O(\varepsilon))\tau_i d + O\Bigg(\varepsilon \sum_{\ell:\, x\in\mathcal{X}'_\ell} \tau_\ell d\Bigg)\\ &\le (3+O(\varepsilon))\,\mathrm{ED}(x,\mu(x)) + O\Big(\frac{\varepsilon\Delta d}{n}\Big) + O\Bigg(\varepsilon \sum_{\ell:\, x\in\mathcal{X}'_\ell} \tau_\ell d\Bigg),\end{aligned}$$

as desired.

In the remaining case, $x \in \mathcal{X}'_j$ for all $j \ge i$. Let $j = \max \mathcal{I}$ be the maximum index. We have $(1+\varepsilon)\tau_j \ge 1$. By Equation (2),

$$\mathcal{E}(x,\mu(x)) \le 2d \le 2(1+\varepsilon)\tau_j d \le 2(1+\varepsilon)\tau_i d + 2\varepsilon(1+\varepsilon)\sum_{\ell:\, x\in\mathcal{X}'_\ell} \tau_\ell d.$$

Thus

$$\begin{aligned}\mathcal{E}(x,\mu(x)) &\le 2(1+\varepsilon)\tau_i d + 2\varepsilon(1+\varepsilon)\sum_{\ell:\, x\in\mathcal{X}'_\ell} \tau_\ell d\\ &\le 2(1+\varepsilon)^3\,\mathrm{ED}(x,\mu(x)) + O\Big(\frac{\varepsilon\Delta d}{n}\Big) + O\Bigg(\varepsilon \sum_{\ell:\, x\in\mathcal{X}'_\ell} \tau_\ell d\Bigg),\end{aligned}$$

as desired. This establishes Equation (1) for all $x \in \mathcal{X} \setminus \mathcal{X}_\emptyset$ and completes the proof of property (b).

For the running time, the number of pairs of $x \in \mathcal{X}$ and Δ-local $y \in \mathcal{Y}$ is

$$\frac{n}{d}\sum_{i\in\mathcal{I}} O\Big(\frac{\Delta}{\varepsilon^2\tau_i d}\Big) = O\Bigg(\frac{n\Delta}{\varepsilon^2 d^2}\sum_{i\in\mathcal{I}}\frac{1}{\tau_i}\Bigg) = O\Big(\frac{n^2}{\varepsilon^4 d^2}\Big),$$

since a window of length Δ holds $O(\Delta/\varepsilon\tau_i d)$ points of G_i, each start admits $O(1/\varepsilon)$ lengths, and the $1/\tau_i$ decrease geometrically from $1/\tau_0 = n/\varepsilon\Delta$ with ratio $(1+\varepsilon)^{-1}$, so they sum to $O(n/\varepsilon^2\Delta)$. For every pair, there are $O(\log(n)/\varepsilon)$ shifted $y' \in S(y)$, and each takes constant time to update $\mathcal{E}$. Altogether, we spend $\tilde{O}(n^2/\varepsilon^5 d^2)$ time to build $\mathcal{E}$. □

5.1 Completing the proof of Lemma 1

Let $\Delta \in [\mathrm{OPT}, 2\mathrm{OPT}]$. For the approximation, let $\mathcal{E} : \mathcal{X}\times\mathcal{Y} \to \mathbb{Z}_{\ge 0}$ satisfy Lemma 6. Our DP output is at least OPT since $\mathcal{E}$ only gives upper bounds. It is upper-bounded by the cost of μ using our estimates $\mathcal{E}$ instead of the edit distance. By Lemma 6 and Lemma 2, this is at most

$$\begin{aligned}&\sum_{x\in\mathcal{X}\setminus\mathcal{X}_\emptyset} \mathcal{E}(x,\mu(x)) + d|\mathcal{X}_\emptyset| + (|Y| - \sum_{x\in\mathcal{X}}|\mu(x)|)\\ &\le (3+O(\varepsilon)) \sum_{x\in\mathcal{X}\setminus\mathcal{X}_\emptyset} \mathrm{ED}(x,\mu(x)) + O(\varepsilon\mathrm{OPT}) + d|\mathcal{X}_\emptyset| + (|Y| - \sum_{x\in\mathcal{X}}|\mu(x)|)\\ &\le (3+O(\varepsilon))\mathrm{OPT}.\end{aligned}$$

For the running time, by Lemma 3, we spend $\tilde{O}\Big(n\Delta^{1/2}\tau_i^{1/2}d^{1/2}/\varepsilon^{5/2}\Big)$ randomized time to obtain $\mathcal{Q}_i$, d_i for each $i \in \mathcal{I}$. The $\tau_i^{1/2}$ grow geometrically with ratio $(1+\varepsilon)^{1/2}$ and are less than 1, so

they sum to $O(1/\varepsilon)$, and across all $i \in \mathcal{I}$ we spend $\tilde{O}\Big(n\Delta^{1/2}d^{1/2}/\varepsilon^{7/2}\Big)$ randomized time. Building $\mathcal{E}$ and running the chunked DP takes $\tilde{O}\Big(\frac{n^2}{\varepsilon^5 d^2}\Big)$ time. We choose $d = \frac{n^{2/5}}{\varepsilon^{3/5}\Delta^{1/5}}$ to minimize these two running times. The total running time for fixed Δ is $\tilde{O}\Big(n^{6/5}\Delta^{2/5}/\varepsilon^{19/5}\Big)$, completing the proof of Lemma 1.

References

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

[AN20] Alexandr Andoni and Negev Shekel Nosatzki. "Edit distance in near-linear time: It's a constant factor". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 990–1001.

[And20] Alexandr Andoni. *Simple Constant-Factor Approximation to Edit Distance*. Tech. rep. Manuscript. Columbia University, Aug. 2020.

[BEG+21] Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, MohammadTaghi HajiAghayi, and Saeed Seddighin. "Approximating edit distance in truly subquadratic time: Quantum and mapreduce". In: *Journal of the ACM (JACM)* 68.3 (2021), pp. 1–41. Preliminary version in SODA, 2018.

[BI15] Arturs Backurs and Piotr Indyk. "Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)". In: *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 2015, pp. 51–58.

[BJK+04] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, and R. Kumar. "Approximating edit distance efficiently". In: *45th Annual IEEE Symposium on Foundations of Computer Science*. 2004, pp. 550–559.

[CDG+20] Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucký, and Michael Saks. "Approximating Edit Distance Within Constant Factor in Truly Sub-quadratic Time". In: *Journal of the ACM* 67.6 (Oct. 2020), pp. 1–22. URL: http://dx.doi.org/10.1145/3422823. Preliminary version in FOCS, 2018.

[CH02] Richard Cole and Ramesh Hariharan. "Approximate string matching: A simpler faster algorithm". In: *SIAM Journal on Computing* 31.6 (2002), pp. 1761–1782.

[CKW23] Alejandro Cassis, Tomasz Kociumaka, and Philip Wellnitz. "Optimal Algorithms for Bounded Weighted Edit Distance". In: *64th IEEE Annual Symposium on Foundations of Computer Science , FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*. IEEE, 2023, pp. 2177–2187. URL: https://doi.org/10.1109/FOCS57990.2023.00135.

[GRS20] Elazar Goldenberg, Aviad Rubinstein, and Barna Saha. "Does Preprocessing Help in Fast Sequence Comparisons?" In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 2020, pp. 657–670. URL: https://doi.org/10.1145/3357713.3384300.

[Kle05] Philip N Klein. "Multiple-source shortest paths in planar graphs". In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. 2005, pp. 146–155.

[Lev65] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Doklady Akademii Nauk SSSR* 163.4 (1965), pp. 845–848.

[MP80] William J Masek and Michael S Paterson. "A faster algorithm computing string edit distances". In: *Journal of Computer and System Sciences* 20.1 (1980), pp. 18–31.

[MR26] Xiao Mao and Aviad Rubinstein. "Approximation Schemes for Edit Distance and LCS in Quasi-Strongly Subquadratic Time". In: *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*. 2026, pp. 744–754.

[Mye86] Eugene W. Myers. "An $O(ND)$ difference algorithm and its variations". In: *Algorithmica* 1.2 (1986), pp. 251–266.

[Ukk85] Esko Ukkonen. "Algorithms for approximate string matching". In: *Information and control* 64.1-3 (1985), pp. 100–118.

[WF74] Robert A Wagner and Michael J Fischer. "The string-to-string correction problem". In: *Journal of the ACM (JACM)* 21.1 (1974), pp. 168–173.

A Omitted proof from Section 3

Lemma 2. *Let $\Delta \in [\mathrm{OPT}, 2\mathrm{OPT}]$. There exist a matching $\mu : \mathcal{X} \to \mathcal{Y}$ and, for each $i \in \mathcal{I}$, a matching $\mu_i : \mathcal{X} \to \mathcal{Y}_i$, with the following properties. Letting $\mathcal{X}_\emptyset \subseteq \mathcal{X}$ be the set of substrings mapped to empty substrings:*

(a) μ is monotone with pairwise-disjoint substrings $\mu(x)$, and μ and every μ_i are smooth and $O(\Delta)$-banded on their nonempty mappings.

(b) $(3+\varepsilon)\sum_{x \notin \mathcal{X}_\emptyset} \mathrm{ED}(x, \mu(x)) + d|\mathcal{X}_\emptyset| + \Big(|Y| - \sum_x |\mu(x)|\Big) \le (3 + O(\varepsilon))\mathrm{OPT}$.

(c) For $x \notin \mathcal{X}_\emptyset$ and $i \in \mathcal{I}$:

(i) If $\mu(x) \in \mathcal{Y}'_i$, then $(1+\varepsilon)\,\mathrm{ED}(x, \mu(x)) \le \tau_i d \le (1+\varepsilon)^2\,\mathrm{ED}(x, \mu(x))$.

(ii) If $\mu(x) \in \mathcal{Y}_i$, then $\mu_i(x) = \mu(x)$.

(iii) If $\mu(x) \in \mathcal{Y}'_{i'}$ for some $i' < i$, then $\mathrm{ED}(x, \mu_i(x)) \le \tau_i d$ and both endpoints of $\mu(x)$ and $\mu_i(x)$ differ by at most $O(\varepsilon\tau_i d)$.

Proof. Fix an optimal alignment π between X and Y. π is monotone wlog. For $x \in \mathcal{X}$, $\pi(x)$ is the substring of Y paired with x. The substrings $\{\pi(x)\}_{x\in\mathcal{X}}$ are pairwise disjoint and their concatenation is exactly Y. Furthermore, $\sum_{x\in\mathcal{X}} \mathrm{ED}(x, \pi(x)) = \mathrm{ED}(X, Y) = \mathrm{OPT}$.

The proof proceeds in two parts. First, we snap π to the grids G_i to create an intermediate mapping μ'. While μ' is monotone, banded, and has good overall cost, it is not smooth. To this end, we remap all "non-smooth" pairs of strings that violate smoothness to empty substrings, along with any substring in between. The resulting matching is smooth, and any additional distance induced by mapping non-smooth pairs to empty substrings is small compared to the distance already implied by any non-smooth pair.

Recall that $\tau_i = \varepsilon(1+\varepsilon)^i\Delta/n$, $\gamma_i = \lfloor \varepsilon\tau_i d/4 \rfloor$, $G_i = (\gamma_i\mathbb{Z}_{\ge 0} \cap [0, |Y|]) \cup \{|Y|\}$ (or every cut if $\gamma_i = 0$), and $\mathcal{Y}'_i$ collects all G_i-delimited intervals of length in the range $[d - \tau_i d, d + \tau_i d]$ along with the empty substrings at points of G_i.

First, let π' be the snapping of π to G_0, obtained by rounding the endpoints of each $\pi(x)$ to their nearest grid points in G_0. The $\pi'(x)$ are disjoint, (weakly) increasing, and their concatenation is Y; snapping adds at most $2\gamma_0$ per substring, so $\mathrm{OPT} \le \sum_x \mathrm{ED}(x, \pi'(x)) \le (1+\varepsilon)\mathrm{OPT}$.

Monotone, banded μ'. Fix $x \in \mathcal{X}$. If $(1+\varepsilon)\,\mathrm{ED}(x, \pi'(x)) > d$, we set $\mu'(x)$ to the empty string at the first point of G_0 at or after $\mathrm{s}(\pi'(x))$. Hereafter these strings are referred to as *dropped singletons*.

Otherwise, $(1+\varepsilon)\,\mathrm{ED}(x, \pi'(x)) \le d$. Let $i'(x) = \min\{i \in \mathcal{I} : (1+\varepsilon)\,\mathrm{ED}(x, \pi'(x)) \le \tau_i d\}$, set $i = i'(x)$, and let $\mu'(x)$ be $\pi'(x)$ with its startpoint snapped up and its endpoint snapped down to G_i. Each snap moves its endpoint inward by at most γ_i, so together they shrink the length by at most $2\gamma_i \le \varepsilon\tau_i d/2$. By symmetric difference, $||\pi'(x)| - d| \le \mathrm{ED}(x, \pi'(x)) \le \tau_i d/(1+\varepsilon)$. Thus, $|\pi'(x)| \ge \varepsilon d/(1+\varepsilon) > \varepsilon d/2 \ge 2\gamma_i$ (using $\tau_i \le 1$). Similarly, $|\pi'(x)| \le d(1+\tau_i/(1+\varepsilon))$. Therefore, the snaps cannot cross and $\mu'(x)$ is a nonempty substring of $\pi'(x)$ with length within $\tau_i d$ of d. Hence $\mu'(x) \in \mathcal{Y}'_i$, and

$$\mathrm{ED}(x, \mu'(x)) \le \mathrm{ED}(x, \pi'(x)) + 2\gamma_i \le \frac{\tau_i d}{1+\varepsilon} + \frac{\varepsilon\tau_i d}{2} \le \tau_i d,$$

which holds for all $\varepsilon \le 1$.

Each nonempty $\mu'(x) \subseteq \pi'(x)$, so μ' is monotone with disjoint mappings. Since,

$$|\mathrm{s}(x) - \mathrm{s}(\pi'(x))| \le \sum_{x'' \preceq x} \mathrm{ED}(x'', \pi'(x'')) \le (1+\varepsilon)\mathrm{OPT} \le O(\Delta),$$

and snapping shifts starts by $O(\gamma_i) = O(\varepsilon d) = O(\Delta)$ (using $d = O(\Delta/\varepsilon)$), μ' is $O(\Delta)$-banded.

Smoothening into μ. Call a pair of X-substrings x, x' *non-smooth* if their mapped distance in Y grows too fast relative to X, specifically:

$$|\mathrm{s}(\mu'(x')) - \mathrm{s}(\mu'(x))| > 4|\mathrm{s}(x') - \mathrm{s}(x)|.$$

For a non-dropped X-substring x, we say it is *pruned* if it lies between any non-smooth pair (inclusive), and *kept* otherwise. We refine our matching into μ by remapping every pruned substring to the empty string at $\mathrm{s}(\mu'(x))$. Let $\mathcal{X}_\emptyset$ denote all $x \in \mathcal{X}$ mapped to empty substrings.

Since any two kept nonempty X-substrings x, x' do not form a non-smooth pair, their mapped distance is naturally bounded by $4|\mathrm{s}(x') - \mathrm{s}(x)|$. Thus, μ is smooth (with $c = 4$), monotone and $O(\Delta)$-banded.

Properties (a) & (b). We charge the three substring types against $\sum_x \mathrm{ED}(x, \pi'(x)) = (1+\varepsilon)\mathrm{OPT}$. Let $\mathrm{ED}_x \stackrel{\text{def}}{=} \mathrm{ED}(x, \pi'(x))$.

The contribution of a kept substring is its edit distance plus any gap it leaves in Y. Since inward snapping to $G_{i'(x)}$ shifts each endpoint by at most $\gamma_{i'(x)}$, the edit distance increases by at most $2\gamma_{i'(x)}$, and the mapped length shrinks by at most $2\gamma_{i'(x)}$. Thus, it contributes $\mathrm{ED}(x, \mu(x)) + (|\pi'(x)| - |\mu(x)|) \le \mathrm{ED}_x + 4\gamma_{i'(x)}$. Since $4\gamma_{i'(x)} \le \varepsilon\tau_{i'(x)}d$, and by the minimality of $i'(x)$ we have $\tau_{i'(x)}d < (1+\varepsilon)^2\,\mathrm{ED}_x$ (for $i'(x) > 0$) or $\tau_0 d$ (for $i'(x) = 0$), the extra cost is bounded by $O(\varepsilon)\,\mathrm{ED}_x + O(\varepsilon^2\Delta d/n)$. Summing over all kept substrings, they contribute at most:

$$(1 + O(\varepsilon)) \sum_{\text{kept } x} \mathrm{ED}_x + O(\varepsilon\mathrm{OPT}).$$

For dropped singleton substrings x, mapping to an empty string contributes d to the edit distance and leaves a gap of $|\pi'(x)|$ in Y. We can bound this cost relative to its optimal cost ED_x. By

symmetric difference, $|\pi'(x)| \le d + \mathrm{ED}_x$. Since x was dropped, we know $\mathrm{ED}_x > d/(1+\varepsilon)$, yielding $d/\mathrm{ED}_x < 1+\varepsilon$. Thus:

$$\frac{d + |\pi'(x)|}{\mathrm{ED}_x} \le \frac{2d + \mathrm{ED}_x}{\mathrm{ED}_x} = 1 + \frac{2d}{\mathrm{ED}_x} < 1 + 2(1+\varepsilon) = 3 + 2\varepsilon.$$

This cost contribution is bounded by $(3+2\varepsilon)\,\mathrm{ED}_x$.

Lastly, for a pruned block B, let x_p, x_q be the first and last substrings of block B, respectively. Let $h = \mathrm{s}(x_q) - \mathrm{s}(x_p)$ be the block's span in X, and $v = \mathrm{s}(\mu'(x_q)) - \mathrm{s}(\mu'(x_p))$ be its corresponding span in Y. Since the block is remapped to an empty string, its total contribution to the cost is $h + v$. By symmetric difference, we can lower bound the cost of the pruned block B. Let $\mathrm{OPT}_B := \sum_{x \in B} \mathrm{ED}(x, \pi'(x)) \ge v - h$.

Since B is a contiguous block of pruned substrings, we can cover B by a sequence of non-smooth pairs (which individually satisfy $v_j > 4h_j$). By passing to a minimal subcover, each coordinate in B is covered at most twice. Thus, we have $\sum_j h_j \le 2h$ and, by the monotonicity of μ', $\sum_j v_j \le 2v$. Using these relations, we can lower bound v:

$$v \ge \frac{1}{2}\sum_j v_j > \frac{1}{2}\sum_j 4h_j = 2\sum_j h_j \ge 2h.$$

We will now bound the contribution of block B by its optimal cost. Since $v > 2h$, we have $v - h > h$, yielding:

$$\frac{h+v}{\mathrm{OPT}_B} \le \frac{h+v}{v-h} = 1 + \frac{2h}{v-h} < 1 + \frac{2h}{h} = 3.$$

Therefore, the cost contribution $h + v$ is strictly less than $3\mathrm{OPT}_B$.

Summing the charges over all three substring types (kept, dropped, and pruned blocks) which land on disjoint parts of OPT, the total cost is bounded by:

$$(3 + O(\varepsilon)) \sum_x \mathrm{ED}(x, \pi'(x)) + O(\varepsilon \mathrm{OPT}) = (3 + O(\varepsilon))\mathrm{OPT}.$$

Note that $|Y| - \sum_x |\mu(x)| = \sum_x (|\pi'(x)| - |\mu(x)|)$, so gaps are counted exactly once. This proves property (b). Property (a) follows directly from the construction above.

Defining μ_i. Fix $i \in \mathcal{I}$ and $x \in \mathcal{X} \setminus \mathcal{X}_\emptyset$. If $\mu(x) \in \mathcal{Y}_i$, we simply set $\mu_i(x) = \mu(x)$. This holds exactly when $i = i'(x)$, giving us subproperties (c)(i) and (c)(ii).

If $i > i'(x)$, then $\mu(x) \notin \mathcal{Y}_i$. In this case, we define $\mu_i(x)$ by snapping both endpoints of $\mu(x)$ to the nearest points in the coarser grid G_i. This ensures $\mu_i(x) \in \mathcal{Y}_i'$. Since snapping moves each endpoint by at most $\gamma_i/2$, the total edit distance induced by snapping is at most the stride length: $\mathrm{ED}(\mu(x), \mu_i(x)) \le \gamma_i = O(\varepsilon \tau_i d)$. This proves the first half of subproperty (c)(iii). If $i < i'(x)$, we set $\mu_i(x)$ to the empty substring at the point of G_i nearest the start of $\mu(x)$; no property above is claimed at these scales.

For the final bound take $i \ge i'(x)$ with $x \notin \mathcal{X}_\emptyset$. If $i = i'(x)$ then $\mu_i(x) = \mu(x)$ and $\mathrm{ED}(x, \mu_i(x)) \le \tau_i d$. If $i > i'(x)$ then $\tau_{i'(x)} \le \tau_i/(1+\varepsilon)$. By the triangle inequality,

$$\mathrm{ED}(x, \mu_i(x)) \le \mathrm{ED}(x, \mu(x)) + \gamma_i \le \tau_i d.$$

Finally, each re-snap shifts starts by $\le \gamma_i/2 = O(\varepsilon d) = O(\varepsilon)|\mathrm{s}(x') - \mathrm{s}(x)|$. Because μ is smooth and $O(\Delta)$-banded, these small shifts ensure that every μ_i is also smooth and $O(\Delta)$-banded. □