


An Improved Sketching Algorithm for Edit Distance

Ce Jin ✉ 🏠 

MIT, Cambridge, MA, USA

Jelani Nelson ✉ 🏠 

University of California at Berkeley, CA, USA

Kewen Wu ✉ 🏠 

University of California at Berkeley, CA, USA

Abstract

We provide improved upper bounds for the simultaneous sketching complexity of edit distance. Consider two parties, Alice with input $x \in \Sigma^n$ and Bob with input $y \in \Sigma^n$, that share public randomness and are given a promise that the edit distance $\text{ed}(x, y)$ between their two strings is at most some given value k . Alice must send a message sx and Bob must send sy to a third party Charlie, who does not know the inputs but shares the same public randomness and also knows k . Charlie must output $\text{ed}(x, y)$ precisely as well as a sequence of $\text{ed}(x, y)$ edits required to transform x into y . The goal is to minimize the lengths $|sx|, |sy|$ of the messages sent.

The protocol of Belazzougui and Zhang (FOCS 2016), building upon the random walk method of Chakraborty, Goldenberg, and Koucký (STOC 2016), achieves a maximum message length of $\tilde{O}(k^8)$ bits, where $\tilde{O}(\cdot)$ hides $\text{poly}(\log n)$ factors. In this work we build upon Belazzougui and Zhang's protocol and provide an improved analysis demonstrating that a slight modification of their construction achieves a bound of $\tilde{O}(k^3)$.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases edit distance, sketching

Digital Object Identifier 10.4230/LIPIcs.STACS.2021.45

Related Version *Full Version*: <https://arxiv.org/abs/2010.13170>

Funding *Ce Jin*: Supported by Akamai Presidential Fellowship.

Jelani Nelson: Supported by NSF award CCF-1951384, ONR grant N00014-18-1-2562, ONR DORECG award N00014-17-1-2127, an Alfred P. Sloan Research Fellowship, and a Google Faculty Research Award.

Acknowledgements We thank Qin Zhang for answering several questions about [7]. C. J. thanks Virginia Vassilevska Williams for several helpful discussions. We thank anonymous reviewers for their helpful comments.

1 Introduction

The edit distance $\text{ed}(x, y)$ between two strings is defined to be the minimum number of character insertions, deletions, or substitutions required to transform x into y . It is one of the most well-studied distance measures on strings, with applications in information retrieval, natural language processing, and bioinformatics. If x, y are each at most length n , the textbook Wagner-Fischer algorithm computes $\text{ed}(x, y)$ exactly in $O(n^2)$ time, with the only improvement since being by a $\log n$ factor due to Masek and Paterson [23]. It has since been shown that an $O(n^{2-\epsilon})$ time algorithm does not exist for any constant $\epsilon > 0$ unless the *Strong Exponential Time Hypothesis* fails [5]. Since the work of [23], several subsequent works have considered setups beyond offline exact algorithms for edit distance, such as faster approximation algorithms [4, 2, 12, 11, 21, 3], metric embeddings [26, 14, 19, 22], smoothed complexity [1, 9], quantum algorithms [8], sublinear time algorithms



© Ce Jin, Jelani Nelson, and Kewen Wu;

licensed under Creative Commons License CC-BY 4.0

38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021).

Editors: Markus Bläser and Benjamin Monmege; Article No. 45; pp. 45:1–45:16

Leibniz International Proceedings in Informatics



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



for gap versions [6, 17, 10, 20], and communication complexity and sketching/streaming [13, 7, 15, 18, 16]. In this work we focus on communication complexity, and specifically *simultaneous communication complexity*.

In the communication model, Alice has input string x and Bob has y . They, or a third party, would like to compute $\text{ed}(x, y)$ as well as a minimum length sequence of edits for transforming x into y . We consider the setting of shared public randomness amongst all parties. The one-way setting in which Alice sends a single message to Bob, who must then output $\text{ed}(x, y)$, is known as the *document exchange problem* and has a long history. In the promise version of the problem for which we are promised $\text{ed}(x, y) \leq k$, Orlitisky [25] gave a deterministic protocol in which Alice only sends $O(k \log(n/k))$ bits in the case of binary strings, which is optimal, with the downside that Bob’s running time to process her message is exponential. Haeupler recently used public randomness to improve Bob’s running time to polynomial with the same asymptotic message length, and it is now known that a polynomial-time recovery algorithm is achievable deterministically if one increases the message length to $O(k \log^2(n/k))$ [15, 18]. Belazzougui and Zhang [7] studied the harder *simultaneous communication* model in which Alice and Bob each send messages to a third party Charlie, who knows neither string but shares knowledge of the public randomness, and Charlie must output $\text{ed}(x, y)$ as well as the edits required to transform x into y . In this model they gave a protocol in which each player sends $O(k^8 \log^5 n) = \tilde{O}(k^8)$ bits.¹

► **Definition 1** (Problem $\mathcal{Q}_{n,k,\delta}$). *Alice and Bob and a referee share public randomness. Alice (resp., Bob) gets a length- n input string x (resp., y) over alphabet Σ , and then sends a “sketch” $sx \in \{0, 1\}^*$ (resp., sy) to the referee. We say the size of the sketch is maximum length of strings sx and sy . After receiving the sketches sx and sy ,*

- *if $\text{ed}(x, y) \leq k$, the referee needs to compute $\text{ed}(x, y)$ as well as an optimal edit sequence from x to y , with success probability at least $1 - \delta$;*
- *if $\text{ed}(x, y) > k$, the referee needs to report “error”, with success probability at least $1 - \delta$.*

Main contribution

We build upon and improve techniques developed in [7] to show that a very slight modification of their protocol needs a sketch size of only $\tilde{O}(k^3)$ bits to solve problem $\mathcal{Q}_{n,k,\delta}$. More precisely, the bound is $O(k^3 \log^2(n/\delta) \log n)$ bits.²

1.1 Proof Overview

We provide a high-level description of the previous results [13, 7] that we build on, and then briefly describe our new ideas.

CGK random walk

The previous sketching result [7] uses a random walk technique developed in [13]. Given two input strings x, y of length n , we append them with infinitely many zeros and initialize two pointers $i = 1, j = 1$. In each step t , we first append $x[i]$ to Alice’s output tape (and append $y[j]$ to Bob’s output tape), and then increment i by $r_t(x[i]) \in \{0, 1\}$, and increment j by $r_t(y[j]) \in \{0, 1\}$, where $r_t: \Sigma \rightarrow \{0, 1\}$ is a random function. The process continues for

¹ We use $\tilde{O}(f)$ throughout this paper to denote $f \cdot \text{polylog}(n)$.

² We remark that both the algorithm of [7] and our improved algorithm are time-efficient, and work in the more restrictive setting where Alice and Bob have only $\text{poly}(k \log(n/\delta))$ memory and receive the input strings in a *streaming* fashion.

$3n$ steps and we consider the evolution of $i - j$, i.e., the distance between the two pointers during this random process. Observe that when $x[i] \neq y[j]$, the change of $i - j$ is a mean-zero random variable in $\{-1, 0, +1\}$ (and we call this a *progress step*); while when $x[i] = y[j]$, the difference $i - j$ will not change.

The main result of [13] shows that the number of progress steps in this random process is at least $\text{ed}(x, y)/2$, and at most $O(\text{ed}(x, y)^2)$ with constant probability. This property was used to design a sketching protocol (with public randomness for generating r_i) for estimating $\text{ed}(x, y)$ up to a quadratic factor error by applying an approximate Hamming distance sketching protocol to the two strings generated by the random walk (where a progress step corresponds to a Hamming mismatch between Alice's and Bob's output strings).

Previous algorithm

The key idea of [7] is the following. A CGK random walk naturally induces a non-intersecting matching between the input strings: we view x and y as a bipartite graph, where if (i, j) is an edge then $x[i] = y[j]$ and i, j are the pointers in some step of the walk. In particular, this matching can be viewed as an edit sequence where a character is unchanged if it is matched.

Using an exact Hamming sketch protocol (with sketch size near-linear in the number of Hamming errors), the referee can recover this matching, as well as all the unmatched characters. Although this matching may not correspond to an edit sequence of optimal length, [7] shows that if we obtain *multiple* matchings by running i.i.d. CGK random walks, then

- (a) if the *intersection* of their matched edges is contained in an optimal matching, then one can extract enough information from the matchings and unmatched characters to recover an optimal edit sequence using dynamic programming;
- (b) if we generate $\text{poly}(\text{ed}(x, y), \log n)$ many i.i.d. CGK random walks, then the precondition of Item (a) is satisfied with constant probability.

Our improvements

We obtain our result by improving the dependence on $\text{ed}(x, y)$ in Item (b) described above. In particular, we reduce the number of required random walks. Our improvements come from two parts.

To obtain the first improvement, we observe that the algorithm of [7] relies on the following two events happening. The first is that, for every edge that does appear in a (fixed) optimal matching, there should be one of the sampled CGK random walks that misses this edge. The second is that the CGK random walks should have few progress steps. In [7], they pay a union bound over the two events to make sure *all CGK random walks* are good for the decoder. This introduces a large dependence on $\text{ed}(x, y)$, mainly due to the fact that the number of Hamming errors in a CGK random walk has a heavy-tailed distribution. We manage to avoid this by arguing that these two events happen simultaneously (see Lemma 15) with decent probability, and then modifying the decoding algorithm to only consider *those good CGK random walks*.

The second improvement comes from improved analysis for Item (b), which depends on the following property of the CGK random walk [7, Lemma 16] (see Subsection 3.3 for how this property can be used): informally, if a string $X[1..L]$ has a certain kind of self-similarity (for example, it is periodic), then with some nontrivial probability, a CGK random walk on X itself starting with two pointers $i = 2, j = 1$ will not pass through the state $(i = L, j = L)$. To be more precise, if there is a non-intersecting matching between $X[1..L]$

and itself, where every matched edge (I, J) satisfies $I > J$, and the number of singletons (unmatched characters) is at most K , then the CGK random walk will miss $(i = L, j = L)$ with $\Omega(1/K^2)$ probability.³

We use a more technical analysis to improve the bound to $\Omega(1/K)$ (see Proposition 18). Now we informally describe our main idea. Starting from the state $(i = 2, j = 1)$, with at least $\Omega(1/K)$ probability it will first reach a state (i, j) with $i - j > d_0 = \Theta(K)$ before reaching $i - j = 0$ (note that $i - j$ can never become negative). Then we will show that with good probability $i - j$ will remain in the range $[d_0/2, 3d_0/2]$. To do this, we show an $O(d_0^2)$ upper bound on the expected total number of progress steps, and use the fact that the expected deviation produced by a P -step one-dimensional random walk is $O(\sqrt{P})$.

To bound the expected total number of progress steps, we divide the evolution of the state (i, j) into several phases, where in each phase the pointers move from a *stable state* (i, j) to another *stable state* (i', j') , satisfying $j' \geq i$ and $i' \geq 2j - i$. Here, a *stable state* (i, j) informally means that we have a good upper bound of $\text{ed}(X[j..i - 1], X[i..2i - j - 1])$ in terms of the number of singletons in the range $[j..2i - j - 1]$ (for example, if X is “close” to a string with period p , and $i - j$ is approximately a multiple of p , then (i, j) is a stable state). We will bound the expected number of progress steps in one phase by $O((i - j) \cdot S + S^2)$, where S denotes the number of singletons in the range $[j..i' - 1]$. We can see the sum of S over all phases is at most $2K$ since each singleton is counted at most twice. Hence, summing up over all phases would give the desired $O(K^2)$ upper bound, if we assume $i - j = \Theta(d_0)$. Although this assumption may lead to circular reasoning, we can get around this issue by a more careful argument.

Organization

We give several needed definitions in Section 2. In Section 3 we state and analyze our sketching algorithm, which as mentioned, is mostly similar to [7] but with small modifications. Section 4 is devoted to our main technical lemma. In comparison with the proof overview, Section 3 is for the first improvement and Section 4 is for the second improvement. Then we discuss limits on our approach and further problems in Section 5. The lower bounds and missing proofs can be found in full version.

2 Preliminaries

In this section we introduce formal definitions.

2.1 Notations

Let $[n]$ denote $\{1, 2, \dots, n\}$, and let $[l..r]$ denote $\{l, l + 1, \dots, r\}$. Let \circ denote string concatenation. Let \mathbb{N} denote the set of natural numbers $\{0, 1, \dots\}$. We consider sketching protocols for strings in Σ^n in this work, where Σ denotes the alphabet. We assume $|\Sigma| \leq \text{poly}(n)$ and $0 \in \Sigma$.⁴

³ There is a subtle gap in the proof of [7, Lemma 16]. On page 18 of their full version, they bounded the number of progress steps in two cases: (1) at least one of the pointers is not in any cluster, and (2) both of the two pointers are in the same cluster. (Their terminology *cluster* refers to a contiguous sequence of matched edges with no singletons in-between.) However, they did not analyze the case where the two pointers are separated in different clusters, and it was not clear to us how to repair that gap using the techniques developed in [7].

⁴ For larger alphabet the algorithm still works but some $\log n$ terms in the bounds become $\log |\Sigma|$. For example, the sketch size will be $O(k^3 \log(n|\Sigma|/\delta) \log(n/\delta) \log n)$. Alternatively, the parties can hash Σ into a new alphabet of size $O(n^2/\gamma)$ and have no hash collisions on the characters appearing in x, y with probability at least $1 - \gamma$.

For a string $s \in \Sigma^n$ and index $1 \leq i \leq n$, $s[i]$ (or sometimes s_i) denotes the i -th character of s . For $1 \leq i \leq j \leq n$, $s[i..j]$ denotes the substring $s[i] \circ s[i+1] \circ \dots \circ s[j]$. If $i > j$ then $s[i..j]$ is the empty string.

2.2 Edit Distance

► **Definition 2** (Edit distance $\text{ed}(\cdot, \cdot)$). *The edit distance between two strings x and y , denoted by $\text{ed}(x, y)$, is the minimum number of edits (insertions, deletions, and substitutions⁵) required to transform x to y .*

► **Definition 3** (Matching induced by edit sequence $\mathcal{M}(S)$). *Given strings x, y and an edit sequence S , we can construct a bipartite graph between x and y , where every character in x that is not substituted nor deleted is connected by an edge to its counterpart in y . These edges form a non-intersecting matching, which we denote by $\mathcal{M}(S)$. Moreover, when S achieves optimal edit distance, we say $\mathcal{M}(S)$ is an optimal matching.*

Though there may be multiple optimal matchings, the following definition specifies a canonical one.

► **Definition 4** (Greedy optimal matching \mathcal{M} , [7]). *Let x, y be two strings. For each edit sequence S achieving optimal edit distance, let $\mathcal{M}(S)$ be the matching induced by S . Then the greedy optimal matching \mathcal{M} is defined to be the smallest $\mathcal{M}(S)$ in lexicographical order. Specifically, we represent $\mathcal{M}(S)$ as a sequence of (i, j) pairs then sort the sequence lexicographically, and the greedy optimal matching is such that this sorted sequence is as lexicographically small as possible.*

2.3 The CGK Random Walk

We review a useful random process called the *CGK random walk*, which was first introduced by Chakraborty, Goldenberg, and Koucký [13], and played a central role in the sketching algorithm of [7].

► **Definition 5** (CGK random walk $\lambda_r(s)$, [13]). *Given a string $s \in \Sigma^n$, an integer $m \geq 0$, and a sequence of $m \cdot |\Sigma|$ random coins interpreted as a random function $r: [m] \times \Sigma \rightarrow \{0, 1\}$, the m -step CGK random walk is a length- m string $\lambda_r(s) \in \Sigma^m$ defined by the following process:*

- Append s with infinitely many zeros.
- Initialize the pointer $p \leftarrow 1$ and the output string $s' \leftarrow \emptyset$.
- For each step $i = 1, \dots, m$:
 - Append $s[p]$ to s' .
 - Update $p \leftarrow p + r(i, s[p])$.
- Output $s' =: \lambda_r(s)$.

For a contiguous segment of the output string $\lambda_r(s)$, the pre-image of this segment refers to the corresponding substring in the original input string s (which may also include the appended trailing zeros if the walk extends beyond s).

⁵ There is another definition of edit distance, denoted by $\text{ed}'(x, y)$, where only insertions and deletions are allowed. We have $\text{ed}(x, y) \leq \text{ed}'(x, y) \leq 2 \cdot \text{ed}(x, y)$, and $\text{ed}'(x, y) = |x| + |y| - \text{LCS}(x, y)$, where LCS stands for *longest common subsequence*. The algorithm in [7], as well as our modification of it, can be easily adapted to work for this variant of edit distance as well.

Due to its usefulness in the two-party setting with public randomness, we also frequently use the term CGK random walk to refer to a pair of random walks (as defined in Definition 5) performed on two input strings x, y using the shared random string r .

Consider a CGK random walk λ on two input strings x, y . We use p_i (resp., q_i) to denote the pointer on string x (resp., y) at the beginning of step i . We refer to the pair (p_i, q_i) as the state of λ at the i -th step, and we write $(p, q) \in \lambda$ if λ passes through the state (p, q) , i.e., there exists some i for which $p_i = p$ and $q_i = q$. We say the i -th step of λ is a progress step if the i -th characters of the output strings $\lambda_r(x)$ and $\lambda_r(y)$ differ, or equivalently, $x[p_i] \neq y[q_i]$.⁶ We say λ walks through x, y , if in the end the two pointers satisfy $p_m \geq |x|$ and $q_m \geq |y|$.

The following theorem establishes the connection between CGK random walks and edit distance. Informally, when $\text{ed}(x, y)$ is small, with good probability the number of progress steps in λ is also small (or equivalently, the Hamming distance between the output strings $\lambda_r(x), \lambda_r(y)$ is small). We provide a simpler proof for its Item (3) in full version.

- **Theorem 6** ([13, Theorem 4.1]). *Let λ be an m -step CGK random walk on x, y . Then*
- (1) *if $m \geq 3 \cdot \max\{|x|, |y|\}$, then λ walks through x, y with probability at least $1 - e^{-\Omega(m)}$;*
 - (2) *given $\lambda_r(x)$ and r , we can reconstruct the pre-image of $\lambda_r(x)$;*
 - (3) $\Pr\left[\#\text{progress steps in } \lambda \geq (T \cdot \text{ed}(x, y))^2\right] \leq O(1/T)$.

2.4 Random Walks

We frequently relate the CGK random walk to the following one-dimensional random walk.

► **Definition 7** (One-dimensional unbiased and self-looped random walk). *A stochastic process $X = (X_t)_{t \in \mathbb{N}}$ on integers is a one-dimensional unbiased and self-looped random walk if its transition satisfies*

$$X_i = \begin{cases} X_{i-1} - 1 & \text{w.p., } 1/4, \\ X_{i-1} & \text{w.p., } 1/2, \\ X_{i-1} + 1 & \text{w.p., } 1/4. \end{cases}$$

► **Remark 8.** Let λ be a CGK random walk on two strings and (p, q) be its state. Define $\Delta = p - q$. Then Δ can be viewed as a one-dimensional unbiased and self-looped random walk, which makes a transition when and only when λ makes a progress step.

By Remark 8 and the martingale property, we have the following lemma, the proof of which can be found in full version.

► **Lemma 9.** *Consider an ∞ -step CGK random walk λ on x, y , where p, q are the pointers on x, y respectively. Let u be an index and let $U, V \geq u - 1$ be any integers. Then the following holds.*

- (1) *Let T_0 be the first time that $p_{T_0} \geq u$. Then we have $\mathbb{E}[|p_{T_0} - q_{T_0}|] \leq 4 \cdot \text{ed}(x[1..U], y[1..V])$.*
- (2) *Let T_1 be the first time that $(p_{T_1} \geq u) \wedge (q_{T_1} \geq u)$. Then we have $\mathbb{E}[|p_{T_1} - q_{T_1}|] \leq 4 \cdot \text{ed}(x[1..U], y[1..V])$.*

⁶ Our definition of “progress step” is different from that of [7], which additionally requires at least one of the two pointers moves forward in that step.

3 Sketches for Edit Distance

For the rest of the paper, we use the following notational conventions:

- n is the length of the input strings; $m := 3n$ is the number of steps in a CGK random walk.
- x, y are the input strings of length n , which is appended with infinitely many zeros; we are promised $\text{ed}(x, y) \leq k$.⁷
- when we use (\cdot, \cdot) to denote a CGK state or an edge between x, y , the first coordinate is a pointer on x and the second is on y .
- \mathcal{M} is the greedy optimal matching of x, y .

Our goal is to prove the following theorem.

► **Theorem 10.** *There is a sketching algorithm for $\mathcal{Q}_{n,k,\delta}$ of sketch size $O(k^3 \log^2(n/\delta) \log n)$ bits. Moreover, the algorithm has the following properties.*

- *The encoding algorithm used by Alice (resp., Bob) only assumes one-pass streaming access to the input string x (resp., y). The time complexity per character is $\text{poly}(k \log(n/\delta))$, and the space complexity is $O(k^3 \log^2(n/\delta) \log n)$ bits.⁸*
- *The decoding algorithm used by the referee has time complexity $\text{poly}(k \log(n/\delta))$.*

In Subsection 3.1, we review the general framework of [7]’s sketching protocol, and highlight our key improvement in Lemma 15. We will prove this key lemma in Subsection 3.2 and Subsection 3.3.

3.1 General Framework

We adopt the definition of *effective alignments* from [7]. Intuitively, an effective alignment between two strings x, y contains the information of an edit sequence from x to y , but does not contain the information of unchanged characters.

► **Definition 11** (Effective alignment \mathcal{A} , [7]). *For two strings $x, y \in \Sigma^n$, an effective alignment \mathcal{A} between x and y is a triplet (G, g_x, g_y) , where*

- *$G = (V_x, V_y, E)$ is a bipartite matching where nodes $V_x = [n], V_y = [n]$ correspond to indices of characters in x and y respectively, and every matched edge $(i, j) \in E$ satisfies $x[i] = y[j]$. Moreover, the matched edges are non-intersecting, i.e., for every pair of distinct edges $(i, j), (i', j') \in E$, we have $i < i'$ iff $j < j'$.*
- *g_x (resp., g_y) is a partial function defined on the set of unmatched nodes $U_x \subseteq V_x$ (resp., $U_y \subseteq V_y$). For each $i \in U_x$ (resp., $j \in U_y$), define $g_x(i) = x[i]$ (resp., $g_y(j) = y[j]$).*

► **Definition 12** (Effective alignments consistent with a CGK random walk, [7]). *Let λ be a CGK random walk on x, y , where p, q are the pointers on x and y respectively. If λ walks through x, y , then we say an effective alignment $\mathcal{A} = (G, g_x, g_y)$ is consistent with λ if for every matched edge $(p, q) \in G$, we have $(p, q) \in \lambda$.*

As mentioned in Subsection 1.1, Alice and Bob use public randomness to instantiate $\tau = O(k \log(n/\delta))$ independent CGK random walks $\lambda_1, \dots, \lambda_\tau$ on x, y . Then, for each CGK random walk λ_i , Alice constructs a sketch sx_i based on her part of the random walk $\lambda_i(x)$,

⁷ We also analyze the behaviour of our algorithms when $\text{ed}(x, y) > k$ in full version.

⁸ The algorithm may use a large number of shared random bits, which can be reduced using Nisan’s generator [24]. The main cost, as we can see from the proof, comes from the CGK random walk. We refer readers to [13] for more details on reducing randomness for the CGK random walk.

and Bob similarly constructs sy_i based on his part of the random walk $\lambda_i(y)$. The referee receives sx_i, sy_i , and tries to extract an effective alignment \mathcal{A}_i from the sketches. Each sx_i (and sy_i) has length $O(k^2 \log(n/\delta) \log n)$. The properties of this protocol are summarized as follows.

► **Construction 13** (Sketch for each random walk, adapting [7]). Let $C \geq 1$ be some large constant and $\eta \in (0, 1)$. There exists an efficient sketching algorithm such that the following holds. Let λ be an m -step CGK random walk on x (and y). Then,

- the sketch size and encoding space are $O(k^2 \log(n/\eta) \log n)$ bits;
- the encoding time per character and decoding time are both $\text{poly}(k \log(n/\eta))$;
- for fixed λ, x, y the following hold with success probability at least $1 - \eta$:
 - the decoder either (a) reports “error”, or (b) outputs an effective alignment \mathcal{A} consistent with λ ;
 - when λ walks through x, y and contains at most $C \cdot k^2$ progress steps, (b) occurs.

The formal proof of Construction 13 can be found in full version.

The final sketches are simply $sx = sx_1 \circ \dots \circ sx_\tau$ and $sy = sy_1 \circ \dots \circ sy_\tau$. The referee tries to obtain an effective alignment from every (sx_i, sy_i) , and then uses the following lemma to compute $\text{ed}(x, y)$ and recover an optimal edit sequence.

► **Lemma 14** ([7, Lemma 14 and Lemma 19]). *There exists a deterministic algorithm taking (sx, sy) as input such that the following holds.*

- *The running time of the algorithm is $\text{poly}(|sx| + |sy|) = \text{poly}(k \log(n/\delta))$.*
- *Let $\mathcal{A}_{i_1}, \dots, \mathcal{A}_{i_w}$ be the effective alignments⁹ decoded from $(sx_1, sy_1), \dots, (sx_\tau, sy_\tau)$. If $w \geq 1$ and each \mathcal{A}_{i_j} is consistent with λ_{i_j} , then the algorithm outputs a valid edit sequence. If, additionally, \mathcal{M} goes through all edges that are common to $\mathcal{A}_{i_1}, \dots, \mathcal{A}_{i_w}$, then the edit sequence is optimal.*

Now we state our key lemma.

► **Lemma 15** (Key Lemma). *There exist some large constants $C_1, C_2 \geq 1$ such that the following holds. Let λ be an ∞ -step CGK random walk on x, y . Then for any fixed $(u, v) \notin \mathcal{M}$, $x[u] = y[v]$, we have*

$$\Pr \left[(u, v) \notin \lambda \wedge \# \text{progress steps in } \lambda \leq C_1 \cdot k^2 \right] \geq \frac{1}{C_2 \cdot k}.$$

Here we reiterate that Lemma 15 summarizes our improvement over the previous work of [7] in two aspects (as mentioned in Subsection 1.1): (1) The previous work only gave a lower bound on $\Pr[(u, v) \notin \lambda]$, while we bound the probability of two events happening simultaneously; (2) The previous work only gave a bound of $\Omega(1/k^2)$, while we give an $\Omega(1/k)$ bound. The proof of this Lemma 15 is divided into two parts in Subsection 3.2 and Subsection 3.3, in which a technical proposition that leads to the improvement in Item (2) will be proved in Section 4.

Assuming Lemma 15, we can prove Theorem 10.

⁹ Although we can check if \mathcal{A}_{i_j} is an effective alignment, we cannot verify (without knowing λ_{i_j}) if \mathcal{A}_{i_j} is an effective alignment *consistent with* λ_{i_j} . This subtle difference comes from that in Construction 13 we do not give any guarantee outside the $1 - \eta$ success probability, where the decoder might provide some effective alignment that is *not* consistent with λ_{i_j} .

Proof of Theorem 10. Let C_3 be a large constant.

For the encoding part, we instantiate $\tau = C_2 k \cdot C_3 \log(n/\delta) = O(k \cdot \log(n/\delta))$ independent m -step CGK random walks $\lambda_i, i \in [\tau]$; and construct each sx_i, sy_i using Construction 13 with parameter $C = C_1, \eta = \delta/(2\tau)$.

For the decoding part, we run the decoding procedure in Construction 13 to obtain $\mathcal{A}_{i_1}, \dots, \mathcal{A}_{i_w}$ for Lemma 14. If $w = 0$ or the edit sequence from Lemma 14 has more than k edits, we report “error”; otherwise we output the edit sequence together with the corresponding edit distance.

Bounds on the parameters. By constructing each sx_i (and sy_i) in parallel, the final sketch size and encoding space are $\tau \cdot O(k^2 \log(n/\eta) \log n) = O(k^3 \log^2(n/\delta) \log n)$ (we omit the space for storing auxiliary information (e.g., pointers) in the calculation, since these are minor terms). The encoding time per character is $\tau \cdot \text{poly}(k \log(n/\eta)) = \text{poly}(k \log(n/\delta))$. The decoding time follows immediately from Lemma 14.

Analysis of the algorithm when $\text{ed}(x, y) \leq k$. Let $S = \{(u, v) \in [n]^2 \mid (u, v) \notin \mathcal{M}, x[u] = y[v]\}$ and define events

- \mathcal{E}_i : λ_i walks through x, y .
- $\mathcal{E}'_i(u, v)$ for $(u, v) \in S$: $(u, v) \notin \lambda_i \wedge \#\text{progress steps in } \lambda_i \leq C_1 \cdot k^2$.

Then

$$\begin{aligned} & \Pr[\forall (u, v) \in S, \exists i \in [\tau], \mathcal{E}_i \wedge \mathcal{E}'_i(u, v)] \\ & \geq 1 - \sum_{(u, v) \in S} (1 - \Pr[\neg \mathcal{E}_1] - \Pr[\neg \mathcal{E}'_1(u, v)])^\tau \\ & \geq 1 - n^2 \cdot \left(1 - e^{\Omega(n)} - \frac{1}{C_2 \cdot k}\right)^\tau \quad (\text{due to Theorem 6 and Lemma 15}) \\ & \geq 1 - \frac{\delta}{2}. \end{aligned} \tag{1}$$

Let $\lambda_{i_1}, \dots, \lambda_{i_w}$ be the random walks walking through x, y and containing at most $C_1 \cdot k^2$ progress steps. Since $\eta = \delta/(2\tau)$ in Construction 13 and by union bound, the decoder, with probability at least $1 - \delta/2$, for each (sx_i, sy_i) either reports “error”, or outputs an effective alignment \mathcal{A}_i consistent with λ_i . Conditioning on this, Construction 13 must at least obtain effective alignments $\mathcal{A}_{i_1}, \dots, \mathcal{A}_{i_w}$ that are consistent with the corresponding random walks. Combined with (1), with probability at least $1 - \delta$, for any $(u, v) \in S$ there exists some λ_{i_j} missing it. Then the edit sequence from Lemma 14 is optimal. ◀

3.2 Proof of Lemma 15: Case $|u - v| > 100 \cdot k$

Proof of Lemma 15: Case $|u - v| > 100 \cdot k$. Assume without loss of generality $u > v$. We stop λ when it meets u . Then by Item (1) in Lemma 9, at this time the state (p, q) satisfies $\mathbb{E}[|p - q|] \leq 4 \cdot k$. Hence by Markov’s inequality,

$$\Pr[(u, v) \notin \lambda] \geq \Pr[p - q \leq 100 \cdot k] = 1 - \Pr[p - q > 100 \cdot k] \geq 1 - \frac{4 \cdot k}{100 \cdot k} = 0.96. \tag{2}$$

On the other hand, by setting C_1 large enough we know from Theorem 6

$$\Pr[\#\text{progress steps in } \lambda \leq C_1 \cdot k^2] \geq 0.99.$$

Hence, by setting C_2 large enough, we have

$$\Pr\left[(u, v) \notin \lambda \wedge \#\text{progress steps in } \lambda \leq C_1 \cdot k^2\right] \geq 0.96 + 0.99 - 1 \geq \frac{1}{C_2 \cdot k}. \quad \blacktriangleleft$$

3.3 Proof of Lemma 15: Case $|u - v| \leq 100 \cdot k$

First we need the following definition.

► **Definition 16** (Stable zone \mathcal{Z} , [7]). *The stable zone \mathcal{Z} of (u, v) consists of substrings $x[u'..u], y[v'..v]$ of equal length $L = u - u' + 1 = v - v' + 1$, where $L \leq \min\{u, v\}$ is the maximum possible length satisfying $x[u'..u] = y[v'..v]$. In particular, $u - v = u' - v'$; and $(u', v') \neq (1, 1)$ as $(u, v) \notin \mathcal{M}$.*

Moreover, we say a state (p, q) enters \mathcal{Z} if $p \geq u'$ and $q \geq v'$.

We will find Claim 17 useful. For completeness we include its proof in full version.

▷ **Claim 17** ([7, Claim 21]). Consider an ∞ -step CGK random walk λ on x, y , where p, q are the pointers on x, y respectively. Let T be the first time that λ enters \mathcal{Z} , i.e., $(p_T \geq u') \wedge (q_T \geq v')$. Then $\Pr[p_T - q_T \neq u - v] = \Pr[p_T - q_T \neq u' - v'] \geq 2/3$.

We will also rely on the following technical result, the proof of which is in Section 4.

► **Proposition 18.** *There exists a universal constant $C_4 \geq 1$ such that the following holds. Assume X, Y are two identical length- L strings over alphabet Σ . Assume there exists a size- M matching $(i_1, j_1), \dots, (i_M, j_M) \in [L]^2$ such that*

- $i_t > j_t$ and $X[i_t] = Y[j_t]$ hold for all $t \in [M]$;
- $i_1 < i_2 < \dots < i_M$ and $j_1 < j_2 < \dots < j_M$.

Let $\rho = C_4 \cdot (L - M)$ and (\hat{I}, \hat{J}) be any state satisfying $\hat{I} - \hat{J} \geq \rho$. Then a CGK random walk on X, Y starting from (\hat{I}, \hat{J}) will miss (L, L) with probability at least 0.5.

By symmetry, we derive the following corollary.

► **Corollary 19.** *Let $C_4 \geq 1$ be the same constant in Proposition 18. Assume X, Y are two identical length- L strings over alphabet Σ . Assume there exists a size- d matching $(i_1, j_1), \dots, (i_M, j_M) \in [L]^2$ such that*

- $i_t > j_t$ holds for all $t \in [M]$, or $i_t < j_t$ holds for all $t \in [M]$;
- $X[i_t] = Y[j_t]$ holds for all $t \in [M]$;
- $i_1 < i_2 < \dots < i_M$ and $j_1 < j_2 < \dots < j_M$.

Let $\rho = C_4 \cdot (L - M)$ and (\hat{I}, \hat{J}) be any state satisfying $|\hat{I} - \hat{J}| \geq \rho$. Then a CGK random walk on X, Y starting from (\hat{I}, \hat{J}) will miss (L, L) with probability at least 0.5.

Proof of Lemma 15: Case $|u - v| \leq 100 \cdot k$. Let $C_5 \geq 1$ be a large constant. We will apply Proposition 18 with parameter $M \geq L - 103 \cdot k$; and let $\rho = C_4 \cdot 103k$ be the corresponding bound in it.

We expect λ to have the following three phases:

- \mathcal{E}_1 : λ enters \mathcal{Z} in a state (p_1, q_1) within $C_5 \cdot k^2$ progress steps, where $0 < |(p_1 - q_1) - (u - v)| \leq 200 \cdot k$.
- \mathcal{E}_2 : Starting from (p_1, q_1) and within $2 \cdot \rho^2$ progress steps, λ reaches a state (p_2, q_2) where either $(p_2, q_2) > (n, n)$ or $|(p_2 - q_2) - (u - v)| \geq \rho$. Also, during the walk from (p_1, q_1) to (p_2, q_2) , λ never reaches some state (p, q) satisfying $(p - q) - (u - v) = 0$.
- \mathcal{E}_3 : $(u, v) \notin \lambda$ and $\#\text{progress steps in } \lambda \leq 2 \cdot \rho^2 + C_5 \cdot (k^2 + (\rho + 301 \cdot k)^2)$.

In fact we have the following claim, the proof of which can be found in full version.

▷ **Claim 20.** $\Pr[\mathcal{E}_1] \geq 0.5$, $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1/(2 \cdot \rho)$, and $\Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq 1/4$.

Assuming Claim 20, we have the following desired bound

$$\Pr\left[(u, v) \notin \lambda \wedge \#\text{progress steps in } \lambda \leq C_1 \cdot k^2\right] \geq \Pr[\mathcal{E}_3] \geq \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq \frac{1}{C_2 \cdot k}$$

by setting $C_1 = 2 \cdot (103 \cdot C_4)^2 + C_5 \cdot (1 + (301 + 103 \cdot C_4)^2)$ and $C_2 = 16$. ◀

4 CGK Random Walks on Self-similar Strings

This section is devoted for Proposition 18. It characterizes CGK random walks on strings of certain self-similarity, which may be interesting on its own.

► **Proposition** (Proposition 18 restated). *There exists a universal constant $C_4 \geq 1$ such that the following holds. Assume X, Y are two identical length- L strings over alphabet Σ . Assume there exists a size- M matching $(i_1, j_1), \dots, (i_M, j_M) \in [L]^2$ such that*

- $i_t > j_t$ and $X[i_t] = Y[j_t]$ hold for all $t \in [M]$;
- $i_1 < i_2 < \dots < i_M$ and $j_1 < j_2 < \dots < j_M$.

Let $\rho = C_4 \cdot (L - M)$ and (\hat{I}, \hat{J}) be any state satisfying $\hat{I} - \hat{J} \geq \rho$. Then a CGK random walk on X, Y starting from (\hat{I}, \hat{J}) will miss (L, L) with probability at least 0.5.

We will first provide necessary definitions and state basic properties in Subsection 4.1. Then present the proof in Subsection 4.2. All missing proofs can be found in full version.

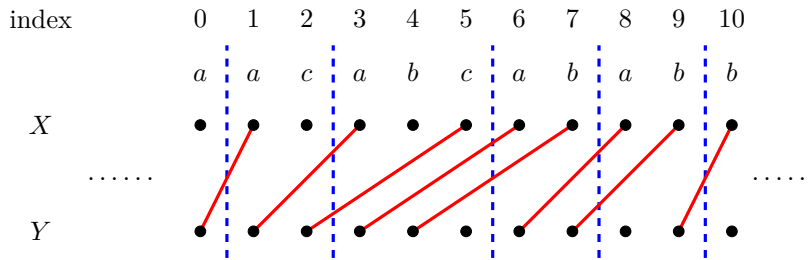
4.1 Stable States

We fix the matching in Proposition 18, so when we say (i, j) is a matched edge it means (i, j) is an edge in the matching. We extend X, Y to $X[-\infty.. \infty], Y[-\infty.. \infty]$ by adding dummy characters $X[i] = Y[i] = X[L]$ for all $i > L$, and $X[i] = Y[i] = X[1]$ for all $i < 1$. We also add matched edges $(i, i - 1)$ for all $i > L$ as well as $i \leq 1$. Note that all the edges are still non-intersecting. Though the added characters may not be consistent with the original input strings x, y , it does not change the probability of the walk missing (L, L) . Since $X = Y$ and the initial state satisfies $\hat{I} \geq \hat{J} + \rho \geq \hat{J}$, any future state (I, J) must still satisfy $I \geq J$.

We introduce the notion of *stable segment*.

► **Definition 21** (Stable segment). *We say $[l..r]$ is a stable segment, if for every matched edge (I, J) (where we must have $I > J$), exactly one of the following two conditions hold:*

- $J < l$ and $I \leq r$.
- $J \geq l$ and $I > r$.



► **Figure 1** A stable partition for $X[1..L] = Y[1..L] = acabcabab$ ($L = 9$).

For example in Figure 1, every segment separated by blue dashed lines is a stable segment.

► **Remark 22.** To gain a better intuition of the definition, consider the special case where the string $X[1..L]$ has period p and every matched edge (I, J) inside segment $[1..L]$ satisfies $I - J = p$. In this periodic case, a segment contained in $[2..L - 1]$ is stable if and only if its length is p .

Our motivation is that, when there are few unmatched characters, using our more generalized definition we can approximately preserve the nice properties of periodic strings. For example, when X has period p , the strings $X[i..i + tp - 1]$ and $X[i + tp..i + 2tp - 1]$

45:12 An Improved Sketching Algorithm for Edit Distance

must be identical. In a non-periodic case, we can similarly prove that $X[i..j-1]$ and $X[j..j+(j-i)-1]$ have small edit distance if $[i..j-1]$ can be divided into several stable segments. In the remaining part of the section, readers are encouraged to use the periodic case for a more intuitive understanding.

► **Definition 23** (Stable partition \mathcal{P} and stable states). *Consider a partition $\mathcal{P} = (\mathcal{P}_i)_i$ of the integers into segments, where $\mathcal{P}_i = [p_i..p_{i+1}-1]$ and $p_i < p_{i+1}$. We say \mathcal{P} is a stable partition if every \mathcal{P}_i is a stable segment. Then we say*

- *state (I, J) is a (\mathcal{P}, b) -stable state, if there exists some i such that $J = p_i$ and $I = p_{i+b}$;*
- *state (I, J) is a b -stable state, if there exists a stable partition \mathcal{P} such that (I, J) is a (\mathcal{P}, b) -stable state;*
- *state (I, J) is a stable state, if there exists some $b \geq 0$ such that (I, J) is a b -stable state. In particular, when $I \geq J > L$, (I, J) is always a stable state.*

Given a stable partition \mathcal{P} , we can define a predecessor function for \mathcal{P} as follows.

► **Lemma 24** (Stable predecessor for a stable partition). *Let $\mathcal{P} = (\mathcal{P}_i)_i$ be a stable partition where $\mathcal{P}_i = [p_i..p_{i+1}-1]$ and $p_i < p_{i+1}$. Then there exists a non-decreasing function $\text{pred}_{\mathcal{P}}: \mathbb{Z} \rightarrow \mathbb{Z}$ such that the following holds:*

- *For every i , $\text{pred}_{\mathcal{P}}(p_{i+1}) = p_i$.*
- *For every I , we have $\text{pred}_{\mathcal{P}}(I) \leq I - 1$, and $[\text{pred}_{\mathcal{P}}(I)..I - 1]$ is a stable segment.*

Now we extend the definition to b -stable predecessor.

► **Definition 25** (b -stable predecessors $\text{pred}_{\mathcal{P}}^{(b)}(\cdot)$). *Let \mathcal{P} be a stable partition. Define*

$$\text{pred}_{\mathcal{P}}^{(b)}(I) = \begin{cases} I & b = 0, \\ \text{pred}_{\mathcal{P}}(\text{pred}_{\mathcal{P}}^{(b-1)}(I)) & b \geq 1. \end{cases}$$

As an example, in Figure 1 $\text{pred}_{\mathcal{P}}^{(3)}(8) = 1$.

We will bound the edit distance between stable states using the number of singletons.

► **Definition 26** (Singleton). *Every unmatched $X[i]$ or $Y[j]$ is called a singleton.*

Let $\text{sing}_X[l, r]$ (resp., $\text{sing}_Y[l, r]$) denote the number of singletons in $X[l..r-1]$ (resp., $Y[l..r-1]$). Let $\text{sing}[l, r] := \text{sing}_X[l, r] + \text{sing}_Y[l, r]$.

► **Lemma 27.** *Let \mathcal{P} be a stable partition. For $I < I'$, let $J = \text{pred}_{\mathcal{P}}(I)$, $J' = \text{pred}_{\mathcal{P}}(I')$. Then*

- (a) $\text{ed}(X[I..I'-1], Y[J..J'-1]) \leq \text{sing}_X[I, I'] + \text{sing}_Y[J, J'] \leq \text{sing}[J, I'];$
- (b) $|(I' - J') - (I - J)| = |(I' - I) - (J' - J)| \leq \text{sing}_X[I, I'] + \text{sing}_Y[J, J'] \leq \text{sing}[J, I'].$

4.2 Proof of Proposition 18

Before proving Proposition 18, we need the following lemma, which shows a CGK random walk goes from a stable state to a distant stable state with low cost.

► **Lemma 28** (From stable to stable). *Consider a CGK random walk starting from a stable state (I_0, J_0) , $I_0 > J_0$. Let D be a distance bound satisfying $D \geq I_0 - J_0$.*

Consider the first time $T > 0$ that either $I_T - J_T > D$, or the following three conditions hold simultaneously: $J_T \geq I_0$, and $I_T \geq 2I_0 - J_0$, and (I_T, J_T) is a stable state. Let P be the number of progress steps before time T and let $S = \text{sing}[J_0, I_T]$. Then

$$\mathbb{E}[P - 2000 \cdot (S \cdot D + S^2)] \leq 0.$$

The process of Lemma 28 consists of a “catch-up phase” (i.e., from a stable state to a distant non-stable state) and then a “stabilization phase” (i.e., from a non-stable state to a nearby stable state). We describe the latter one as Lemma 29.

► **Lemma 29** (From non-stable to stable). *Consider a CGK random walk starting from a non-stable state $(\tilde{I}_0, \tilde{J}_0)$, $\tilde{I}_0 > \tilde{J}_0$. Let \mathcal{P} be a stable partition and let b' be such that $\tilde{L}_0 < \tilde{J}_0 < \tilde{R}_0$ where $\tilde{L}_0 = \text{pred}_{\mathcal{P}}^{(b')}$ (\tilde{I}_0) , $\tilde{R}_0 = \text{pred}_{\mathcal{P}}^{(b'-1)}$ (\tilde{I}_0) . Let D be a distance bound satisfying $D \geq \tilde{I}_0 - \tilde{J}_0$. Consider the first time T' that either $(\tilde{I}_{T'}, \tilde{J}_{T'})$ is a stable state or $\tilde{I}_{T'} - \tilde{J}_{T'} > D$. Let P' be the number of progress steps before time T' and let $S' = \text{sing}[\tilde{L}_0, \tilde{I}_{T'}]$. Then*

$$\mathbb{E} \left[P' - 50 \cdot \left((\tilde{R}_0 - \tilde{J}_0)(\tilde{J}_0 - \tilde{L}_0) + S' \cdot D + S'^2 \right) \right] \leq 0.$$

Given previous lemmas to control progress steps, we are now ready to prove Proposition 18.

Proof of Proposition 18. Since $X[1]$ and $Y[L]$ are matched to dummy characters after we extend X, Y , there are $K := \text{sing}[-\infty, +\infty] = 2 \cdot (L - M - 1)$ singletons in total. Let $d := \hat{I} - \hat{J}$ be the initial distance between the two pointers and let $D := 2 \cdot d$. For a state $(I, J), I \geq J$,

- if $I = J \leq L$ or $I - J > D$, then we say it is a *failure state*;
- if it is not a failure state and $I > L$, then we say it is a *success state*.

We stop the CGK random walk when it reaches a success state or a failure state. The former case implies that the random walk misses (L, L) . So it suffices to prove that we stop at a success state with probability at least 0.5.

Phases in the CGK random walk. Let $I_0 = \hat{I}, J_0 = \hat{J}$ and $t_0 = 0$. Let $t_1 \geq 0$ be the first time that either (I_{t_1}, J_{t_1}) is a stable state or $I_{t_1} - J_{t_1} > D$.

For every $i \geq 2$, if $(I_{t_{i-1}}, J_{t_{i-1}})$ is neither a success state nor a failure state, we know $J_{t_{i-1}} < I_{t_{i-1}} \leq L$ and $I_{t_{i-1}} - J_{t_{i-1}} \leq D$. Then we recursively define $t_i > t_{i-1}$ to be the first time that either $I_{t_i} - J_{t_i} > D$, or the following three conditions hold simultaneously: $I_{t_i} \geq 2I_{t_{i-1}} - J_{t_{i-1}}$, and $J_{t_i} \geq I_{t_{i-1}}$, and (I_{t_i}, J_{t_i}) is a stable state.

Assume we stop at (I_{t_m}, J_{t_m}) , which is either a success state or a failure state. Let P_i be the number of progress steps made during the time interval $[t_i, t_{i+1})$. Then $P := \sum_{i=0}^{m-1} P_i$ is the total number of progress steps before we stop.

Bounds on $\mathbb{E}[P_0]$. Let \mathcal{P} be an arbitrary stable partition and let b be such that $\text{pred}_{\mathcal{P}}^{(b)}(I_0) \leq J_0 < \text{pred}_{\mathcal{P}}^{(b-1)}(I_0)$. Let $L_0 = \text{pred}_{\mathcal{P}}^{(b)}(I_0)$, $R_0 = \text{pred}_{\mathcal{P}}^{(b-1)}(I_0)$. Since $X[1]$ is matched to $Y[0]$, we know $\text{pred}_{\mathcal{P}}(1) = 0$. Hence applying Lemma 27 with $I' = R_0, I = 1$, we have

$$(R_0 - L_0) - (1 - 0) \leq \text{sing}[0, R_0] \leq \text{sing}[0, I_0].$$

Therefore, let $S_0 = \text{sing}[0, I_{t_1}]$ and we have

$$(J_0 - L_0)(R_0 - J_0) \leq \left\lfloor \frac{R_0 - L_0}{2} \right\rfloor \cdot \left\lceil \frac{R_0 - L_0}{2} \right\rceil \leq (\text{sing}[0, I_0])^2 \leq S_0^2.$$

Thus by Lemma 29, we have $\mathbb{E}[P_0 - 50 \cdot (2 \cdot S_0^2 + S_0 \cdot D)] \leq 0$.

Bounds on $\mathbb{E}[P_i], 1 \leq i \leq m - 1$. Let $S_i := \text{sing}[J_{t_i}, I_{t_{i+1}}]$. By Lemma 28, we have $\mathbb{E}[P_i - 2000 \cdot (S_i \cdot D + S_i^2)] \leq 0$.

45:14 An Improved Sketching Algorithm for Edit Distance

Final bounds. Note that $\sum_{0 \leq i < m} S_i \leq 2 \cdot \text{sing}[0, I_{t_m}] \leq 2 \cdot K$. This is because $J_{t_{i+1}} \geq I_{t_i}$ for all $i \geq 1$, implying each singleton is counted at most twice. Hence

$$\begin{aligned} \mathbb{E}[P] &= \mathbb{E} \left[\sum_{i=0}^{m-1} P_i \right] \leq \mathbb{E} \left[2000 \sum_{i=0}^{m-1} (S_i D + S_i^2) \right] \leq 2000 \cdot \mathbb{E} \left[D \sum_{i=0}^{m-1} S_i + \left(\sum_{i=0}^{m-1} S_i \right)^2 \right] \\ &\leq 8000 \cdot (K \cdot d + K^2). \end{aligned}$$

For $1 \leq j < +\infty$, let r_j be the deviation brought by the j -th progress step.¹⁰ Then r_j are i.i.d. random variables with $\Pr[r_j = 0] = 1/2$, $\Pr[r_j = +1] = \Pr[r_j = -1] = 1/4$. Hence by Cauchy-Schwarz inequality, we have

$$\begin{aligned} \mathbb{E} \left[\left| \sum_{j=1}^P r_j \right| \right] &= \mathbb{E} \left[\left| \sum_{j=1}^{+\infty} r_j \cdot 1_{\{j \leq P\}} \right| \right] \leq \sqrt{\mathbb{E} \left[\left(\sum_{j=1}^{+\infty} r_j \cdot 1_{\{j \leq P\}} \right)^2 \right]} = \sqrt{\mathbb{E} \left[\frac{P}{2} \right]} \\ &\leq \sqrt{4000 \cdot (K \cdot d + K^2)}. \end{aligned}$$

Observe that in the end we have $I_{t_m} - J_{t_m} = d + \sum_{j=1}^P r_j$. By setting $\rho = C_4 \cdot (L - M)$ for some large enough constant C_4 , we have $d \geq \rho \geq C_4 \cdot K/2$ and

$$d \geq 4 \cdot \sqrt{4000 \cdot (K \cdot d + K^2)} \geq 4 \cdot \mathbb{E} \left[\left| \sum_{j=1}^P r_j \right| \right].$$

Then by Markov's inequality, with probability at least 0.5 we have $|I_{t_m} - J_{t_m} - d| \leq d/2$, which indicates (I_{t_m}, J_{t_m}) is not a failure state. Hence we stop at some success state with probability at least 0.5. \blacktriangleleft

5 Discussion

Building upon [7], we present an improved sketching algorithm for edit distance with sketch size $\tilde{O}(k^3)$. Although the algorithm itself is essentially the same as in [7], the analysis is more involved. We conclude the paper with a few remarks on further problems.

- **Lower bounds.** We conjecture the lower bound for this problem (i.e., $\mathcal{Q}_{n,k,\delta}$) is $\tilde{\Omega}(k^2)$, since $\Theta(k^2)$ is the distortion of the CGK random walk embedding [13]. However, to the best of our knowledge, there is no lower bound beyond $\tilde{\Omega}(k)$. (Since we do not find any paper formally stating the lower bounds, we present them in Appendix B of full version.)
- **Edit distance.** It is natural to wonder if current framework can be pushed further. For example, is it possible that we only run $\tau = O(1)$ rounds of CGK random walks and there will be an optimal matching going through all edges that are common to these walks? Unfortunately this is not true, and we can show $\tau = \Omega(\sqrt{k})$ with the following example:

$$\begin{aligned} x &= Ac_1c_2 \cdots c_{k-1} Bc_1c_2 \cdots c_{k-1} \underbrace{d \cdots d}_{2k} Ac_1c_2 \cdots c_{k-1}, \\ y &= Bc_1c_2 \cdots c_{k-1} \underbrace{d \cdots d}_{2k} Ac_1c_2 \cdots c_{k-1} Bc_1c_2 \cdots c_{k-1}. \end{aligned}$$

¹⁰Though we will only use r_1, \dots, r_P , we define it in this way to make the next Cauchy-Schwarz inequality easier to understand.

Then with probability $1 - \Theta(1/\sqrt{k})$, a CGK random walk walks through (k, k) . Note that $\text{ed}(x, y) \leq 2 \cdot k$ by deleting $x[1..k]$ and inserting $y[4k + 1..5k]$. However any edit sequence leaving (k, k) matched will have at least $(2 \cdot k + 1)$ edits, where the one more edit comes from substituting $x[1]$ with $y[1]$. Moreover, this example may generalize to the binary alphabet by replacing each symbol with a short random binary string.

- **Ulam distance.** The Ulam distance is the edit distance on two permutations, i.e., $x \in [n]^n$ (resp., $y \in [n]^n$) and $x_i \neq x_j$ (resp., $y_i \neq y_j$) for distinct i, j . Our algorithm (as well as the algorithm in [7]) works for Ulam distance with an improved bound $\tilde{O}(k^{2.5})$. This comes from the following observation: there is no matched edge in the stable zone, hence the length of stable zone is at most k , which means we can set $\rho = O(\sqrt{L})$ in Proposition 18. It would be interesting to improve the algorithm for Ulam distance.
- **Only the distance.** Though our algorithm computes edit distance as well as an optimal edit sequence, it is reasonable to relax the problem by simply asking for the distance or even a constant approximation of the distance. However, we are not aware of any result achieving better sketch size in this setting.

References

- 1 Alexandr Andoni and Robert Krauthgamer. The smoothed complexity of edit distance. *ACM Trans. Algorithms*, 8(4):44:1–44:25, 2012. doi:10.1145/2344422.2344434.
- 2 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 377–386, 2010. doi:10.1109/FOCS.2010.43.
- 3 Alexandr Andoni and Negev Shekel Nosatzki. Edit distance in near-linear time: it’s a constant factor. *CoRR*, abs/2005.07678, 2020. To appear in FOCS 2020. arXiv:2005.07678.
- 4 Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM J. Comput.*, 41(6):1635–1648, 2012. doi:10.1137/090767182.
- 5 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.
- 6 Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2004. doi:10.1109/FOCS.2004.14.
- 7 Djamal Belazzougui and Qin Zhang. Edit distance: Sketching, streaming, and document exchange. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 51–60. IEEE Computer Society, 2016. Full version at arXiv:1607.04200. doi:10.1109/FOCS.2016.15.
- 8 Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, and Saeed Seddighin. Approximating edit distance in truly subquadratic time: Quantum and MapReduce. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1170–1189, 2018. doi:10.1137/1.9781611975031.76.
- 9 Mahdi Boroujeni, Masoud Seddighin, and Saeed Seddighin. Improved algorithms for edit distance and LCS: beyond worst case. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 1601–1620. SIAM, 2020. doi:10.1137/1.9781611975994.99.
- 10 Joshua Brakensiek, Moses Charikar, and Aviad Rubinfeld. A simple sublinear algorithm for gap edit distance. *CoRR*, abs/2007.14368, 2020. arXiv:2007.14368.
- 11 Joshua Brakensiek and Aviad Rubinfeld. Constant-factor approximation of near-linear edit distance in near-linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 685–698, 2020. doi:10.1145/3357713.3384282.

- 12 Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucký, and Michael E. Saks. Approximating edit distance within constant factor in truly sub-quadratic time. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 979–990. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00096.
- 13 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 712–725. ACM, 2016. doi:10.1145/2897518.2897577.
- 14 Moses Charikar and Robert Krauthgamer. Embedding the ulam metric into ℓ_1 . *Theory Comput.*, 2(11):207–224, 2006. doi:10.4086/toc.2006.v002a011.
- 15 Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic document exchange protocols, and almost optimal binary codes for edit errors. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 200–211, 2018. doi:10.1109/FOCS.2018.00028.
- 16 Kuan Cheng and Xin Li. Efficient document exchange and error correcting codes with asymmetric information. *CoRR*, abs/2007.00870, 2020. To appear in SODA 2021. arXiv:2007.00870.
- 17 Elazar Goldenberg, Robert Krauthgamer, and Barna Saha. Sublinear algorithms for gap edit distance. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1101–1120, 2019. doi:10.1109/FOCS.2019.00070.
- 18 Bernhard Haeupler. Optimal document exchange and new codes for insertions and deletions. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 334–347, 2019. doi:10.1109/FOCS.2019.00029.
- 19 Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. *Mathematische Annalen*, 334:821–852, 2006.
- 20 Tomasz Kociumaka and Barna Saha. Sublinear-time algorithms for computing & embedding gap edit distance. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2020.
- 21 Michal Koucký and Michael E. Saks. Constant factor approximations to edit distance on far input pairs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 699–712. ACM, 2020. doi:10.1145/3357713.3384307.
- 22 Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into l_1 . *SIAM J. Comput.*, 38(6):2487–2498, 2009. doi:10.1137/060660126.
- 23 William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980. doi:10.1016/0022-0000(80)90002-1.
- 24 Noam Nisan. Pseudorandom generators for space-bounded computation. *Comb.*, 12(4):449–461, 1992. doi:10.1007/BF01305237.
- 25 Alon Orlitsky. Interactive communication: Balanced distributions, correlated files, and average-case complexity. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 228–238, 1991. doi:10.1109/SFCS.1991.185373.
- 26 Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *J. ACM*, 54(5):23, 2007. doi:10.1145/1284320.1284322.