Brief Announcement: Upper and Lower Bounds for Edit Distance in Space-Efficient MPC

Debarati Das*
Pennsylvania State University
State College, Pennsylvania, USA
debaratix710@gmail.com

Jacob Gilbert†
University of Maryland
College Park, Maryland, USA
jgilber8@umd.edu

MohammadTaghi Hajiaghayi†
University of Maryland
College Park, Maryland, USA
hajiaghayi@gmail.com

Tomasz Kociumaka
Max Planck Institute for Informatics
Saarland Informatics Campus
Saarbrücken, Saarland, Germany
tomasz.kociumaka@mpi-inf.mpg.de

Barna Saha‡
University of California, San Diego
San Diego, California, USA
barnas@ucsd.edu

ABSTRACT

In the Massively Parallel Computation (MPC) model, data is distributed across multiple processors, and we call an algorithm *space-efficient* if each machine has $n^{1-\epsilon+o(1)}$ memory with a machine count of $\Omega(n^{\epsilon})$.

In this paper, we study the string edit distance problem in the MPC model, presenting both a new algorithm and lower-bound results. A space-efficient MPC algorithm computing the exact edit distance using $O(n^{\epsilon})$ communication rounds is known by updating the algorithm of Chowdhury and Ramachandran (SPAA 2008). A key contribution of our work is the introduction of the first space-efficient MPC algorithm, which uses subpolynomial number of rounds and provides an $n^{o(1)}$-approximation of edit distance, where n denotes the length of the input strings.

Further, we complement this algorithm with new lower-bound results. The Orthogonal Vector (O.V.) conjecture states that no truly subquadratic time algorithm exists for the Orthogonal Vector problem, and it follows directly from the Strong Exponential Time Hypothesis (SETH). Drawing inspiration from this, we propose the Strong O.V. Conjecture that posits that there is no space-efficient MPC algorithm capable of solving O.V. using $n^{\epsilon-\Omega(1)}$ communication rounds. The Strong O.V. conjecture has far-reaching consequences, yielding the first (or strengthened) lower bounds for a myriad of problems in the MPC model including graph diameter estimation, computing Fréchet distance, longest common subsequence, and dynamic time warping. Via an MPC reduction from O.V. to edit distance, we give the first conditional lower bound for string edit distance in the MPC model showing that there does not exist any space-efficient, $n^{\epsilon-\Omega(1)}$-round MPC exact edit distance algorithm.

*This project is partially supported by NSF CAREER Award CCF 2337832.
†Partially supported by DARPA QuICC, ONR MURI 2024 award on Algorithms, Learning, and Game Theory, Army-Research Laboratory (ARL) grant W911NF2410052, NSF AF:Small grants 2218678, 2114269, 2347322.
‡This project is partially supported by NSF grants 1652303, 1909046, 2112533, and EnCORE:HDR TRIPODS Phase II grant 2217058.

SPAA '24, June 17–21, 2024, Nantes, France

ACM ISBN 979-8-4007-0416-1/24/06.
https://doi.org/10.1145/3626183.3660265

CCS CONCEPTS

• **Theory of computation → Massively parallel algorithms**; *MapReduce algorithms.*

KEYWORDS

Massively Parallel Computations, Edit Distance, Approximation, Hardness

ACM Reference Format:
Debarati Das, Jacob Gilbert, MohammadTaghi Hajiaghayi, Tomasz Kociumaka, and Barna Saha. 2024. Brief Announcement: Upper and Lower Bounds for Edit Distance in Space-Efficient MPC. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3626183.3660265

1 INTRODUCTION

The Massively Parallel Computations (MPC) model has emerged as a theoretical model of interest in the last decade to reason about popular distributed processing frameworks such as MapReduce, Spark, and Hadoop, which are widely used for large-scale data analysis. The MPC model consists of multiple processors each with local memory sublinear in the size of input. The input is distributed across processors and computation happens in synchronous rounds. Between consecutive rounds, the partial results through local computations are communicated across processors subject to their local memory constraints. This inter-round communication poses a major bottleneck in the MPC setting. More formally, in the MPC model, each machine has $n^{1-\epsilon+o(1)}$ memory for $\epsilon > 0$ where n is the input size, and the number of machines is $\Omega(n^{\epsilon})$ which is sufficient to hold all the data across the machines. An MPC algorithm is called *space-efficient* if the total memory across all the processors is $n^{1+o(1)}$. The primary goal here is to design space-efficient MPC algorithms minimizing the number of rounds, aka round complexity, which is directly proportional to inter-round communications.

One of the challenging and extensively studied problems in the domain of fine-grained complexity is the string edit distance problem, which also serves as a primary focus in this paper. The edit

distance between two strings is the minimum number of insertions, deletions, and substitutions needed to transform one string into the other. Edit distance is one of the fundamental measures of string similarity, serving as the foundational principle that propels applications from small-scale applications such as `diff` and `awk` to large-scale problems including database queries, genomics, search engines, and even social networks. Computing edit distance in the MPC model has also received significant attention [6, 7, 14]. Hajiahgayi, Seddighin, and Sun [14] and Boroujeni, Ehsani, Ghodsi, Hajiaghayi, and Seddighin [6] give MPC algorithms for edit distance in constant and logarithmic rounds, respectively, achieving $(1+\epsilon)$, and $(3+\epsilon)$ approximations. However, their algorithms are not space-efficient due to the utilization of superlinear total memory. Later works [7, 14] improved the total memory requirement but still failed to achieve space-efficiency. A space-efficient exact edit distance algorithm in MPC can be achieved via the tiling approach of [11], which was initially proposed as a cache-efficient algorithm, but takes up to $O(n^{\epsilon})$ rounds given n^{ϵ} MPC machines. Presently, there does not exist any algorithm to compute edit distance exactly or approximately in the MPC model with total space $O(n^{1+o(1)})$ and subpolynomial rounds. The existing MPC lower bounds [12, 15] do not shed any light on this stark absence, and prior PRAM results [4] cannot be simulated in MPC as the number of MPC machines is limited to sublinear.

2 OUR CONTRIBUTION AND TECHNIQUES

In this paper we investigates the edit distance problem in the MPC model while presenting a new space-efficient algorithm and lower bound results.

Approximate MPC Edit Distance. We give the first space-efficient MPC algorithm to approximate the edit distance in subpolynomial communication rounds. We say an algorithm $\mathcal{A}$ *α-approximates* a problem X if, for all instances v of problem X, it holds that $X(v)/\alpha \leq \mathcal{A}(v) \leq \alpha X(v)$. Formally we show the following.

Theorem 2.1. *Let X and Y be strings with $n = |X|$. For $\epsilon \in (0,1)$ and $\Theta(1) \leq r \leq \Theta(\frac{\log n}{\log\log n})$ where $n^{1-\epsilon} \geq n^{1/r}$, there is an MPC algorithm that $O(r \cdot n^{1/r})$-approximates $\mathrm{ed}(X,Y)$ in $O(r \cdot \epsilon^{-1})$ rounds of MPC using $O(n^{\epsilon})$ processors and $\tilde{O}(n \cdot (\log n)^{O(r)})$ total memory.*

In the statement of Theorem 2.1, the parameter r governs a trade-off between the round complexity, the total space complexity, and the approximation ratio of our MPC algorithm. As the focus of this work is on space-efficient MPC algorithms, we observe that the total memory becomes $n^{1+o(1)}$ as long as $r = o(\frac{\log n}{\log\log n})$. In particular, when $r = \Theta(\sqrt{\log n / \log\log n})$, the algorithm achieves a $2^{O(\sqrt{\log n \log\log n})}$ approximation ratio using sublogarithmic communication rounds and $n \cdot 2^{O(\sqrt{\log n \log\log n})}$ total memory. It remains an open problem to find a poly-logarithmic-factor approximation algorithm that still uses $n^{1+o(1)}$ total memory and $n^{o(1)}$ communication rounds.

Next, we provide a brief overview of the algorithm that is an MPC simulation of the approximate edit distance algorithm of Andoni, Krauthgamer, and Onak [3] (or, more precisely, its recent simplified implementation by Bringmann, Cassis, Fischer, and Nakos [9]). We approximate edit distances between substrings of X and Y using a tree with a small height and low degree. Each node in the tree corresponds to a substring of X, and each level represents a partition of X. At each node, we determine the minimum edit distance between its corresponding substring of X and various substrings of Y. These fragment approximations are recursively combined, starting from the leaf nodes and ending at the root, which produces our approximation for $\mathrm{ed}(X,Y)$.

During the above computation, we may need to compute the edit distance between up to $k \geq \mathrm{ed}(X,Y)$ substrings of Y for each substring of X. Therefore, one primary challenge in simulating this algorithm in MPC arises when k is large. In such cases, assigning a single processor with local memory $O(S)$ to each node of the tree proves inadequate, as a more sophisticated method is required to compute k approximations per node. To address this, we allocate k/S processors for each node, with each processor computing a specific subset of the k different approximations. Additionally, when recursively combining the approximation results, a careful and synchronized approach is necessary to keep the number of communication rounds bounded. A pivotal step in this approach involves an elegant reduction to the *prefix sums* problem (with respect to various associative operators), that is solvable in constant rounds and with nearly linear total memory in MPC [13]. We show that leveraging the prefix sum problem enables the combination of k approximations across multiple processors in constant communication rounds and nearly linear total memory. For a more comprehensive understanding of the algorithm and this non-trivial reduction, please refer to the full version of the work.

Another big challenge in simulating the serial approximate edit distance algorithm arises when each of the $O(n)$ leaf nodes (and subsequent internal nodes) demands k computations. This becomes particularly problematic when k is large, posing a risk of surpassing our total linear memory budget in MPC. To this end, the serial algorithm [3, 9] employs a randomized pruning mechanism. While this slightly degrades the approximation guarantees, it effectively manages the challenge. The expected number of remaining nodes becomes $n^{1+o(1)}/k$, thereby requiring nearly linear total memory for the entire algorithm. The challenge lies in the fact that since the randomness is determined after the MPC algorithm starts, our algorithm needs to adaptively allocate processors to nodes without the ability to schedule assignments deterministically (the schedule remains unknown before the algorithm runs). To address this, we create a request set for processors to be assigned to node computations and utilize a reduction to the prefix sum problem to allocate processors and provide them with the necessary data for their tasks.

MPC Orthogonal Vector Conjectures. We complement the above algorithm by introducing new lower-bound results for computing edit distance in the MPC model. To achieve this, we present the Strong O.V. conjecture and subsequently provide a reduction from O.V. to edit distance in the MPC model. Motivated by the fine-grained complexity and lack of lower bounds in the MPC model, our proposed Strong Orthogonal Vector Conjecture serves as the first super-logarithmic lower bound in MPC as well as the first lower bound conjecture for a non-graph problem in MPC.

Conjecture 2.2 (Strong (MPC) Orthogonal Vector Conjecture)**.** *For every constant $\epsilon \in (0,1)$, there is no MPC algorithm that solves O.V.*

using $n^{\epsilon-\Omega(1)}$ communication rounds, $\Theta(n^{\epsilon})$ processors, and $n^{1+o(1)}$ total memory.

Next, we provide some insights guiding our conjecture. A direct solution to O.V. involves computing the product of all $O(n^2)$ pairs of vectors, which poses a considerable communication challenge in MPC. The computation of the dot product necessitates the vectors to be co-located on the same processor at a given time step. Consider a processor in our MPC model with local memory denoted as $S = n^{1-\epsilon+o(1)}$, where $\epsilon \in (0,1)$. If the total memory across all processors is $n^{1+o(1)}$, implying $O(n^{\epsilon})$ processors, then in a single communication round, at most $O(n^{2-\epsilon+o(1)})$ pairs of vectors can be stored together on some processor. Consequently, any O.V. algorithm required to check all vector pairs will need $\Omega(\frac{n^2}{n^{2-\epsilon+o(1)}}) = n^{\epsilon-\Omega(1)}$ communication rounds in MPC. Despite a processor having the capability for unlimited local computations in a single round, the restricted local memory mandates that an O.V. algorithm employs multiple communication rounds to ensure the vectors are on the same processor at some point. For further discussion on intuition for the conjecture, please refer to the full version of the work.

From MPC O.V. Conjectures to Edit Distance in the MPC model. Next, we introduce a reduction from O.V. to edit distance that, together with the Strong O.V. conjecture, establishes the first hardness for space-efficient edit distance computation in the MPC model.

Theorem 2.3. *For every constant $\epsilon \in (0,1)$, if there is an MPC algorithm that solves exact string edit distance in MPC using R communication rounds, $\Theta(n^{\epsilon})$ processors, and $n^{1+o(1)}$ total memory, then there is an MPC algorithm that solves O.V. in $O(R)$ rounds with the same space complexity.*

Our MPC reduction builds on the reduction of Backurs and Indyk [5] converting an instance of O.V. to an instance of edit distance. The key technique in their reduction involves vector gadgets, that converts each input vector into a string with size linear in the dimension d of the vectors. In this construction, orthogonal vectors have an edit distance of d, while non-orthogonal vectors have an edit distance of $3m + (d - m)$, where m is the dot product of the vectors. These vector gadgets are concatenated with additional padding to ensure that any optimal edit sequence must align the gadgets of orthogonal vectors rather than non-orthogonal ones. Consequently, if the edit distance between the final pair of strings is below a certain threshold, the original sets of vectors must have included an orthogonal pair. This reduction can be performed in MPC with $\tilde{O}(n)$ total memory and $O(1)$ communication rounds and provide the reduction for completeness.

Using the above theorem, we can then give the first conditional hardness for edit distance in MPC. This justifies the significance of our approximation algorithm and provides a matching lower bound for the exact algorithm of [11].

Corollary 2.4. *If the Strong MPC Orthogonal Vector Conjecture is true, then any MPC exact string edit distance algorithm using $O(n^{\epsilon})$ processors for $\epsilon \in (0,1)$ and $n^{1+o(1)}$ total memory requires $n^{\epsilon-o(1)}$ communication rounds.*

Further Consequences of Strong MPC O.V. Conjecture. The Strong MPC Orthogonal Vector Conjecture has significant implications for proving hardness results in the MPC model. This is because, in fine-grained complexity, SETH-based lower bounds typically require reducing k-SAT to the O.V. problem. Typical properties seen in these reductions are (i) the gadgets employed in the reduction are of small size and (ii) they can be computed independently for each vector. This allows the reductions to be implemented in the MPC model using a constant number of rounds and $O(n^{1+o(1)})$ total space. Consequently, there is a broad range of problems e.g., LCS [1], dynamic time warping [10], Fréchet distance [8], graph diameter [16], single source max flow [2] etc., for which there does not exist any $n^{\epsilon-o(1)}$-round MPC algorithm using $O(n^{\epsilon})$ processors and $n^{1+o(1)}$ total memory space assuming the Strong MPC O.V. conjecture. Formally one can show the following.

Theorem 2.5. *For any problem X if there exists a reduction f from O.V. (with binary vector sets $A, B \subseteq \{0,1\}^d$) to X such that (i) for each vector $v \in A \cup B$, size of the gadget $f(v)$ is polynomial in d, and (ii) the gadgets $f(v)$ can be computed locally, then assuming the Strong MPC O.V. conjecture, no MPC algorithm can solve X with $O(n^{\epsilon})$ processors for $\epsilon \in (0,1)$, $n^{1+o(1)}$ total memory and $n^{\epsilon-\Omega(1)}$ communication rounds.*

REFERENCES

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. 2015. Tight Hardness Results for LCS and Other Sequence Similarity Measures. In *FOCS 2015*. 59–78.

[2] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. 2015. Matching Triangles and Basing Hardness on an Extremely Popular Conjecture. In *STOC*. 41–50.

[3] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. 2010. Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity. In *FOCS 2010*. 377–386.

[4] Alberto Apostolico, Mikhail J. Atallah, Lawrence L. Larmore, and Scott McFaddin. 1990. Efficient Parallel Algorithms for String Editing and Related Problems. *SIAM J. Comput.* 19, 5 (1990), 968–988.

[5] Arturs Backurs and Piotr Indyk. 2018. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.* (2018), 1087–1097.

[6] Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, and Saeed Seddighin. 2018. Approximating Edit Distance in Truly Subquadratic Time: Quantum and MapReduce. In *SODA 2018, 2018*. 1170–1189.

[7] Mahdi Boroujeni and Saeed Seddighin. 2019. Improved MPC algorithms for edit distance and Ulam distance. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*. 31–40.

[8] Karl Bringmann. 2014. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *FOCS 2014*. IEEE Computer Society, 661–670.

[9] Karl Bringmann, Alejandro Cassis, Nick Fischer, and Vasileios Nakos. 2022. Almost-optimal sublinear-time edit distance in the low distance regime. In *STOC, 2022*. 1102–1115.

[10] Karl Bringmann and Marvin Künnemann. 2015. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. In *FOCS, 2015*. 79–97.

[11] Rezaul Alam Chowdhury and Vijaya Ramachandran. 2008. Cache-efficient dynamic programming algorithms for multicores. In *SPAA, 2008*. 207–216.

[12] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. 2019. Conditional Hardness Results for Massively Parallel Computation from Distributed Lower Bounds. In *FOCS*. 1650–1663.

[13] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. 2011. Sorting, Searching, and Simulation in the MapReduce Framework. In *ISAAC 2011*. 374–383.

[14] MohammadTaghi Hajiaghayi, Saeed Seddighin, and Xiaorui Sun. 2019. Massively Parallel Approximation Algorithms for Edit Distance and Longest Common Subsequence. In *SODA, 2019*. 1654–1672.

[15] Danupon Nanongkai and Michele Scquizzato. 2020. Equivalence Classes and Conditional Hardness in Massively Parallel Computations. *CoRR* abs/2001.02191 (2020).

[16] Liam Roditty and Virginia Vassilevska Williams. 2013. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC 2013*. 515–524.

Received 23 January 2024