An Efficient Implementation of Cosine Distance on Minimal Absent Word Sets using Suffix Automata

Mohammad Tamimul Ehsan, Sk. Sabit Bin Mosaddek, and M Saifur Rahman

Bangladesh University of Engineering and Technology

Abstract. The Cosine Distance on Minimal Absent Word Sets, CD-MAWS in short, was recently introduced for alignment-free phylogeny estimation. Here we introduce a refined CD-MAWS method, significantly reducing computational complexity from $\mathcal{O}(\max(km^3n, km^2n \log n))$ to $\mathcal{O}(\max(m^2n, kmn))$ while maintaining tree quality. Here, m is the number of species, n is the size of the whole genome of a species, and k is the maximum length of a minimal absent word (MAW). This advancement is achieved through a revised cosine distance calculation method, binary encoding of MAWs, and the adoption of suffix automata for MAW generation, addressing the main computational bottleneck and setting a better runtime for alignment-free phylogenetic analysis.

Keywords: Phylogeny · CD-MAWS · Suffix Automata

1 Introduction

The study of phylogeny is crucial for understanding evolutionary relationships among species. The Cosine Distance on Minimal Absent Word Sets (CD-MAWS) [1] offers an alignment-free approach for phylogeny reconstruction from whole genome analysis. However, CD-MAWS is hampered by its computational inefficiency. Our improvements aim to streamline this process, offering a more practical solution for genomic analysis. Building on the foundational work of the CD-MAWS method, our research extends the application of Minimal Absent Words (MAWs) in phylogenetic analysis by addressing computational bottlenecks and enhancing methodological efficiency.

We introduce suffix automata for MAW generation in lexicographical order and leverage this ordered generation for determining the intersection set of MAWs between species pairs in an efficient way. Furthermore, we propose to encode each nucleotide in two bits so as to process any MAW of up to a certain length in a very efficient way in various operations. We thus reduce computational complexity while maintaining accuracy. Our work opens up new avenues for large-scale genomic studies, offering a faster and more scalable tool for unraveling the complexities of evolutionary histories.

2 Background and Related Works

2.1 Background

Phylogenetic analysis is essential for understanding evolutionary relationships among species. Traditional methods, such as multiple sequence alignment (MSA), have been widely used but come with significant computational challenges and limitations, particularly when dealing with large datasets or highly divergent sequences. To address these issues, alignment-free methods have gained attention in recent years [2,3,4,5]. These methods bypass the need for sequence alignment, thus reducing computational complexity.

Minimal Absent Words (MAWs) are a novel concept in alignment-free phylogenetic analysis. MAWs are sequences that are conspicuously absent from a given genome but present in related genomes. These absent sequences provide unique phylogenetic signals that help in constructing evolutionary relationships. The use of MAWs, combined with mathematical measures such as cosine distance, offers a promising approach for phylogeny estimation, balancing computational efficiency and accuracy.

2.2 Related Works

Several alignment-free methods have been proposed and benchmarked in recent studies. Zielezinski et al. (2019) [10] conducted a comprehensive benchmarking of alignment-free sequence comparison methods, highlighting the strengths and weaknesses of various approaches. Their study provides a foundation for evaluating new methods such as the one proposed in this paper.

One notable alignment-free method is the use of k-mers [11], which involves counting the frequency of fixed-length substrings within sequences. This method has been widely adopted due to its simplicity and effectiveness. However, it may not fully capture the phylogenetic signals present in the sequences, especially for more complex datasets. Another significant contribution to the field is the work of Liu et al. (2009) [9], who developed a rapid and accurate method for large-scale coestimation of sequence alignments and phylogenetic trees. Although not alignment-free, their approach sets a high standard for accuracy and computational efficiency.

In addition to k-mers and large-scale coestimation methods, techniques like spectral methods [12] and information-theoretic measures [13] have been explored for phylogenetic analysis. These methods offer alternative ways to capture evolutionary signals in sequence data without relying on traditional alignment procedures.

Minimal absent words (MAWs) are increasingly important in alignment-free sequence analysis [14,15,16,17,18] They efficiently distill essential genomic sequence information, enabling comparisons without traditional alignment methods. MAWs support distance-based techniques such as Length Weighted Index [17] and Jaccard Distance [15], which aid in reconstructing phylogenies. Additionally, they facilitate the creation of composition vectors used in cosine similarity computation [19]. The CD-MAWS approach [1] specifically demonstrates their effectiveness through experiments on biological and simulated datasets, showcasing their potential in reconstructing species phylogeny. However, as the number of MAWs increases, memory constraints can prevent the method from generating vectors needed for cosine distance calculations, and the running time can also become prohibitively expensive. In this study, we integrated suffix automata and encoding techniques to address these limitations, achieving significantly improved runtime and memory usage.

3 Preliminary Concepts

3.1 Minimal Absent Words

Minimal Absent Words (MAWs) are words that do not appear in a string, but all their proper substrings do. For example, *acg*, *gac*, and *gct* are examples of MAWs for the string $x = actgcga$. However, *acta* is an absent word but not a MAW of x. Notably, proper substrings are derived by deleting one or more characters from the beginning or end of the word. Proper prefixes and suffixes are similarly defined. The number of MAWs of a string of size n is $O(n)$.

3.2 Cosine Distance for MAW Vectors

Cosine distance measures the cosine of the angle between two vectors, representing the presence or absence of MAWs in genomic sequences. It ranges from 0 (identical orientation, implying high similarity) to 1 (orthogonal orientation, implying low similarity). This metric is advantageous for its normalization, accounting for differences in sequence lengths and compositions. The distance can be shown as:

$$\mathit{CD\text{-}MAWS}(x, y) = 1 - \frac{V^x.V^y}{|V^x||V^y|}$$

Here V^x (V^y) is a vector representation of the sequence x (y), such that the value assigned to vector V^x (V^y) along dimension w identifies whether w is a MAW of x (y) or not. These vectors are also known as composition vectors.

3.3 Length of MAWs

Aurell et al. [6] derived lower and upper limit on the length within which the bulk of MAWs of a string of length N lie, assuming a random model:

$$l_{min} = \frac{\ln N - \ln \ln N}{\ln 4}$$

$$l_{max} = \frac{2 \ln N + \ln 9}{\ln 4}$$

Each letter of a genome can be encoded using 2 bits (see Section 4.2). So, using a 64 bit integer we can store a MAW of size upto 32. Thus if we only consider the MAWs from this bulk region, it is possible to analyze whole genomes as large as $\approx 1.4 \times 10^9$ base pairs.

4 Proposed Methodology

We break down our work into two primary segments. First part being the distance calculation ie CD-MAWs. This segment assumes that the MAW set is already calculated and works on that. The second part concerns the calculation of the MAW set of a given string using suffix automata. The description of the proposed methodology and comparison are done seperately for the two segments.

4.1 Two Pointer Method

In the original CD-MAWS method, the proposed approach was to create a composition vector using MAWs for each species, and then calculate the pairwise cosine distances. However, by examining the structure of the cosine similarity, we observe that the numerator of the ratio implies the number of common MAWs, while the denominator is the product of the number of MAWs in each sequence. Without composing the actual vectors, we can directly find the denominator value as the size of the MAW set for each species. For the numerator, we can use a two-pointer method to find the common MAWs between two species. For this, the MAWs need to be sorted. After sorting, we iterate over them using the two-pointer technique to find the number of common MAWs. With our current proposition, this part has a complexity of $\mathcal{O}(kn \log n)$, where k is the maximum MAW length.

If we have m species in our dataset, finding the cosine distance between each pair of them would require $\mathcal{O}(kmn \log n + km^2 n)$. In the original CD-MAWS, the complexity is $O(\max(km^2 n \log n, km^3 n))$.

4.2 Encoding MAWs

To compare two MAWs, which are character arrays or strings, we need to iterate over their length, resulting in a complexity of $O(k)$. However, as there are 4 characters in the DNA set: A, C, G, and T, we can encode them in binary as A (00), C (01), G (10), and T (11). Thus, each MAW can be encoded in a binary string. In many programming languages, including C++, a MAW of length 32 can thus fit in a 'long long' integer. On a 64-bit computer, the comparison can be done in constant time, and on a 32-bit computer, it takes only twice the time, which is also constant. Encoding the MAWs across all species requires $O(kmn)$ time, which subsequently provides $O(k)$ savings in the sorting and comparison steps. Therefore, the total time complexity is $O(kmn + mn \log n + m^2 n)$.

4.3 Generating MAW using Suffix Automata

Suffix automata is a trie-like structure with suffix links. A suffix link from a node points to a separate node that represents the maximum length suffix of the current node. We can leverage this to efficiently compute MAWs of a string. For instance, if we are currently in a node that implies the substring 'xabcd' and we attempt to proceed using ‘y’ but find no path, we check if ‘xabcdy’ is a MAW by moving back using the suffix link. If the string size in the node following the suffix link is less than 4 (i.e., the size of ‘abcd’), then ‘xabcdy’ is not a MAW. Otherwise, we try to proceed using ‘y’ from that node. If we can, it means there is a substring ‘abcdy’. Thus, by the definition of MAW, ‘xabcdy’ is a MAW.

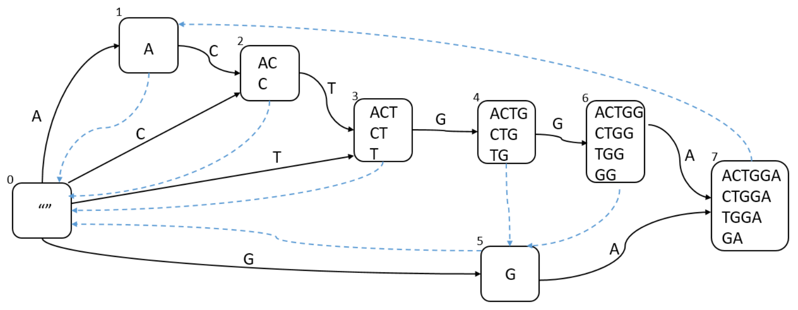

Fig. 1. Suffix automaton of ‘ACTGGA’

For example, consider the word $x = ACTGGA$. We check if two strings $s = GAC$ and $t = CTGA$ are MAWs or not. For GAC, we traverse the trie up to GA to reach node 7. The longest prefix of s is thus a substring of the original word. But from there we cannot move forward with edge C. Therefore, s is not included in x. We then move backward using the suffix link to reach node 1. The length of the word in that node is one, thus having the potential of constructing the longest proper suffix of s. Node 1 has an outgoing edge C, so AC is included in x. Thus, both the longest proper prefix and longest proper suffix of s are included in x but not s itself, making s a MAW of x.

Similarly, for t, we move to node 4 to find CTG by traversing through node 0, 2, 3 using edges C, T, G respectively. Next, we try to move forward using edge A but it is absent. So, $CTGA$ is not present in x but CTG is, which is the longest proper prefix of t. Now we move back using suffix link to node 5. However, this node only holds a length 1 string. Although it has a forward edge of A to construct a suffix of t, it is not going to be the longest proper suffix. Therefore, t is not a MAW of x.

The runtime required to generate MAWs using this method is $O(nk)$, whereas it was previously $O(n)$ using the method proposed by Barton et al. [20]. Although we sacrifice runtime at this stage, we utilize the property of graph traversal. If we traverse the graph in lexicographical order, the MAW set generated will also be lexicographically sorted. Thus, we don’t need to sort the MAWs in the second stage. So the complexity of CD-MAWS become $O(kmn + m^2n)$.

5 Comparative Analysis of MAW Generation

In this section we compare the execution time and memory usage of our proposed method of MAW generation based on Suffix Automata with the MAW generation algorithm proposed by Barton et al. [20] All the comparisons were conducted in a system with Microsoft Windows 10 Pro x64-based PC with WSL Kernel Version: 5.15.133.1-microsoft-standard-WSL2. The Linux Distribution in WSL was Ubuntu 22.04.2 LTS. The processor of the machine is AMD Ryzen 5, 3600 Mhz, 6 Core(s), 12 Logical Processor(s), and RAM was 8 GB.

5.1 Execution Time

Table 1 and Figure 2 compare the execution times of the MAW generation algorithms proposed by us and Barton et al. with an extra sorting step for varying input sizes. Our proposed algorithm shows asymptotic improvements, particularly for large datasets.

Table 1. Execution time comparison (in seconds) between the MAW generation algorithms of Barton et al.[20] and our proposed one with Suffix Automata

Input Size (n)	Barton et al.	Sorting	Total	Proposed
10^1	0.0280	0.0065	0.0345	0.3275
10^2	0.0190	0.0068	0.0258	0.3695
10^3	0.0200	0.0090	0.0290	0.3485
10^4	0.0295	0.0218	0.0513	0.3553
10^5	0.1298	0.1133	0.2430	0.4573
10^6	1.2658	1.0448	2.3105	1.9138
10^7	14.5448	11.1833	25.7280	18.0760

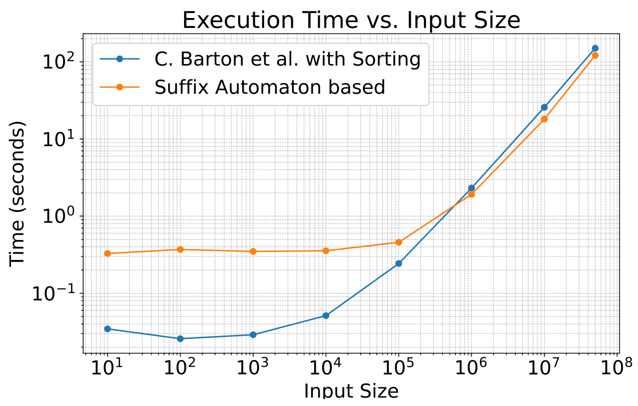

Fig. 2. Execution time comparison between the MAW generation algorithms of Barton et al. [20] and our proposed one with Suffix Automata.

The original CD-MAWS algorithm did not have any constraint on the order of the MAWs required in the MAW set. The algorithm of Barton et al. [20] also does not produce the MAW set in any particular order. However, given that our proposed algorithm processes a sorted list, the MAW set is to be sorted to facilitate the distance calculation step. As the suffix automata already outputs the MAW set in sorted order, it is asymptotically faster for this particular use case. For an input size of 10^7, our method shows a clear asymptotic advantage.

5.2 Memory Usage

Table 2 and Figure 3 summarize the peak memory consumption, showing significant reductions in memory usage with the modified algorithm.

Table 2. Peak memory usage (in MB) comparison between the MAW generation algorithms of Barton et al. [20] and our proposed one with Suffix Automata.

Input Size (n)	Barton et al.	Proposed
10^1	3.77	3.33
10^2	3.82	3.36
10^3	3.84	3.43
10^4	4.46	3.76
10^5	16.13	12.36
10^6	253.26	78.89
10^7	2479.24	620.97

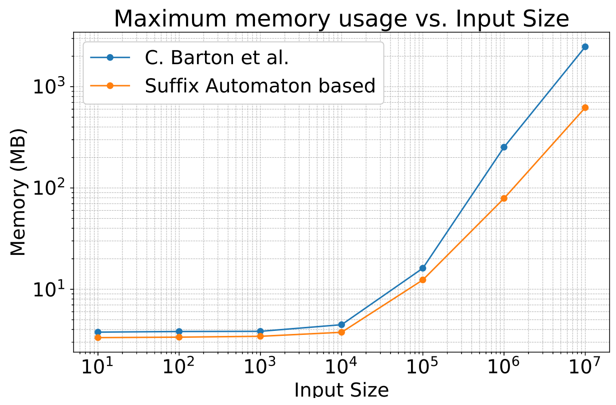

Fig. 3. Peak memory usage (in MB) comparison between the MAW generation algorithms of Barton et al. [20] and our proposed one with Suffix Automata.

6 Comparative Analysis of Cosine Distance Calculation

Previous CD-MAWs implementation introduced a method of composition vector where the length of the vector will be the number of MAW across all DNA. This vector can easily get very large in size. Thus it took more time to create those vectors and also to work with them. As shown in this section, the proposed two-pointer method and encoding techniques significantly reduce the runtime and memory footprint compared to the original CD-MAWS method. All the experiments in this section were conducted in the same computing system as mentioned in Section 5.

6.1 Data Generation

We generated a simulated dataset with varying numbers of species using the Seq-gen Monte Carlo simulation tool [8]. The data comprised 1000 DNA sequences, each having 10000 nucleotides. We then prepared various sub-samples and ran our experiments 5 times on each of them and averaged the execution time and peak memory usage.

6.2 Execution Time

A comparative runtime analysis of the original CD-MAWS method and our proposed method using suffix automata can be found in Table 3. We show two variants of our method: CD-MAWS using suffix automata (CD-MAWS-SA) and another one which also applies bit encoding for nucleotides (CD-MAWS-SA-Encoded). The corresponding graph of this data is presented in Figure 4.

Table 3. Execution Time comparison (in seconds) for different versions of CD-MAWS for varying number of Taxa.

Taxa Count	CD-MAWS	CD-MAWS-SA	CD-MAWS-SA-Encoded
100	2.34	0.37	0.31
200	6.83	1.52	0.91
300	13.55	3.33	1.79
400	23.18	5.89	3.06
500	33.97	9.15	4.63
600	47.72	13.29	6.46
700	63.41	17.87	8.60
800	82.62	23.52	11.10
900	104.12	29.71	14.03
1000	129.76	36.53	17.01

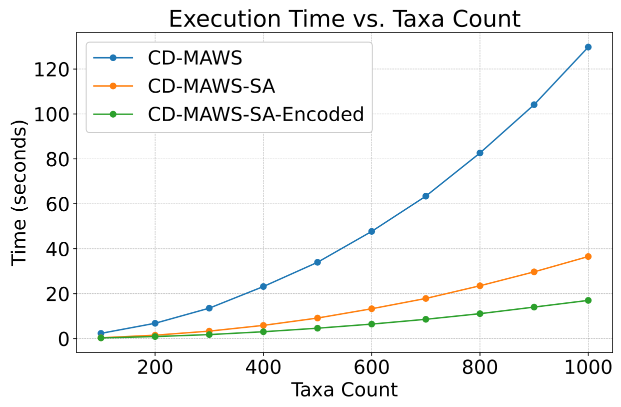

Fig. 4. Execution Time comparison (in seconds) for different versions of CD-MAWS for varying number of Taxa.

6.3 Peak Memory Usage

The composition vector in CD-MAWS generated heavy memory usage in the earlier implementations. By only working with the MAWs (and avoiding the composition vectors), we have significantly decreased the memory footprint. Additionally, encoding the MAWs further reduces memory usage. The memory usage comparison is provided in Table 4, with the corresponding graph presented in Figure 5.

Table 4. Peak memory consumption (in MB) for different versions of CD-MAWS for varying number of Taxa

Taxa Count	CD-MAWS	CD-MAWS-SA	CD-MAWS-SA-Encoded
100	90.07	14.81	8.79
200	199.08	25.99	15.00
300	371.25	37.54	20.96
400	449.67	48.80	26.85
500	526.91	59.90	32.89
600	914.80	71.50	38.66
700	1008.31	82.65	45.53
800	1102.43	98.06	52.14
900	1198.46	111.13	57.99
1000	1286.33	116.76	62.69

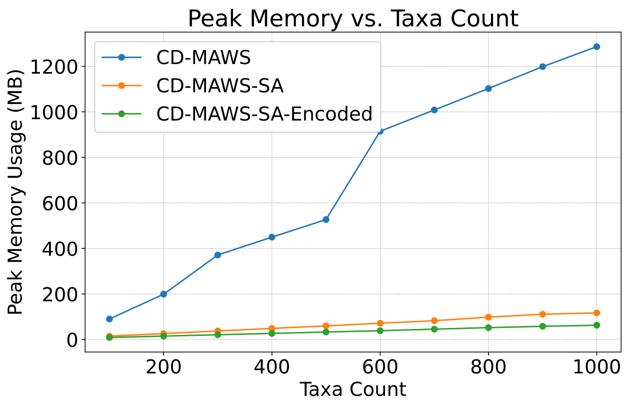

Fig. 5. Peak memory consumption (in MB) for different versions of CD-MAWS for varying number of Taxa

7 Discussion

The enhanced CD-MAWS method presented in this study significantly improves computational efficiency as well as memory consumption. By employing a two-pointer method on the sorted MAW set, encoding MAWs in binary, and utilizing suffix automata for MAW generation, we have reduced the computational complexity from $\mathcal{O}(\max(km^3n, km^2n\log n))$ to $\mathcal{O}(\max(m^2n, kmn))$. These improvements address the primary computational bottlenecks of the original CD-MAWS method, making it more suitable for large-scale genomic studies.

Compared to previous alignment-free methods, such as k-mers and spectral methods, our approach provides a more nuanced understanding of evolutionary relationships through the use of MAWs. This method not only retains the advantages of being alignment-free but also improves the scalability of phylogenetic estimations. However, our study has limitations. The assumption of a maximum MAW length of 32, while practical, may not be optimal for all datasets. For longer MAWs, we propose using hashing. To minimize the probability of hash collisions, we propose employing double hashing. Because of use of hashing, equality checking can still be performed in constant time. We plan to experiment with this approach in future. Adaptive approaches to determining the appropriate MAW length based on the specific characteristics of the data shold also be investigated. Additionally, while the binary encoding of MAWs enhances comparison speed, it may require further optimization for different hardware architectures.

Future research could focus on extending the CD-MAWS method to other types of genomic data, such as RNA or protein sequences, and integrating it with other phylogenetic tools to provide a more comprehensive analysis framework. Exploring the use of machine learning techniques to predict evolutionary relationships based on MAWs could also be a promising direction.

8 Conclusion

In this paper, we presented an enhanced version of the CD-MAWS method for phylogenetic estimation, achieving a significant reduction in computational complexity while maintaining high accuracy. Our improvements, including refined cosine distance calculations, binary encoding of MAWs, and the use of suffix automata for MAW generation, have addressed the major computational challenges of the original method.

These advancements make the CD-MAWS method a more practical and scalable tool for large-scale genomic analyses, capable of efficiently handling modern datasets. The impact of our work lies in its potential to facilitate more accurate and faster phylogenetic estimations, contributing to a deeper understanding of evolutionary relationships. We have made our implementation of CD-MAWS open-source and freely available at `https://github.com/TamimEhsan/cd-maws-sa`. Looking forward, we anticipate that our method will be integrated into broader phylogenetic analysis pipelines and adapted for various types of genomic data. The continued development and refinement of alignment-free methods like CD-MAWS will be crucial for advancing the field of phylogenetic analysis and unlocking new insights into the evolutionary history of life.

Disclosure of Interests. The authors state that there is no competing interest for this study.

References

1. Anjum, N, Nabil, R. L., Rafi, R. I., Bayzid, M. S. and Rahman, M. S.: CD-MAWS: An Alignment-Free Phylogeny Estimation Method Using Cosine Distance on Mini-

mal Absent Word Sets. Journal **IEEE/ACM Trans. Comput. Biol. Bioinformatics** 20, 1 (Jan.-Feb. 2023), 196–205. https://doi.org/https://doi.org/10.1109/TCBB.2021.3136792

2. Yi, H. and Jin, L.: Co-phylog: an assembly-free phylogenomic approach for closely related organisms," Journal **Nucleic Acids Research**, vol. 41, p. e75, Apr 2013.
3. Klotzl, F. and Haubold, B.: Phylonium: fast estimation of evolutionary distances from large samples of similar genomes. Journal **Bioinformatics**, vol. 36, no. 7, pp. 2040–2046, 2020.
4. Ondov, B. D., Starrett, G. J., Sappington, A., Kostic, A., Koren, S., Buck, C. B. and Phillippy, A. M.: Mash screen: high-throughput sequence containment estimation for genome discovery. Journal **Genome Biology**, vol. 20, no. 1, p. 232, 2019.
5. Sarmashghi, S., Bohmann, K., Gilbert, M. T. P., Bafna, V. and Mirarab, S.: Skmer: assembly-free and alignment-free sample identification using genome skims. Journal **Genome Biology**, vol. 20, no. 1, pp. 1–20, 2019.
6. Aurell, E., Innocenti, N. and Zhou, H.-J.:"The bulk and the tail of minimal absent words in genome sequences". Journal **Physical Biology**, vol. 13, no. 2, p. 026004, 2016.
7. Collins, K. and Warnow, T.: "Pasta for proteins". Journal **Bioinformatics**, vol. 34, pp. 3939–3941, 11 2018.
8. Rambaut, A. and Grass, N. C.: Seq-gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Journal **Bioinformatics**, vol. 13, no. 3, pp. 235–238, 1997.
9. Liu, K., Raghavan, S., Nelesen, S., Linder, C. R. and Warnow, T.: Rapid and accurate largescale coestimation of sequence alignments and phylogenetic trees. Journal **Science**, vol. 324, no. 5934, pp. 1561–1564, 2009.
10. Zielezinski, A., Girgis, H.Z., Bernard, G. et al.: Benchmarking of alignment-free sequence comparison methods. Journal **Genome Biology** 20, 144 (2019). https://doi.org/https://doi.org/10.1186/s13059-019-1755-7
11. Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S. and Phillippy, A. M.: "Mash: fast genome and metagenome distance estimation using minhash," Journal **Genome Biology**, vol. 17, no. 1, p. 132, 2016
12. Abeysundera, M., Field, C., Gu, H.: Phylogenetic Analysis Based on Spectral Methods, Journal **Molecular Biology and Evolution**, Volume 29, Issue 2, February 2012, Pages 579–597, https://doi.org/https://doi.org/10.1093/molbev/msr205
13. Liu, Z., Meng, J., Sun, X.: A novel feature-based method for whole genome phylogenetic analysis without alignment: Application to HEV genotyping and subtyping, Journal **Biochemical and Biophysical Research Communications**, Volume 368, Issue 2, 2008, Pages 223-230, ISSN 0006-291X, https://doi.org/https://doi.org/10.1016/j.bbrc.2008.01.070
14. Akon, M., Akon, M., Kabir, M., Rahman, M. S., and Rahman, M. S.: "ADACT: a tool for analysing (dis)similarity among nucleotide and protein sequences using minimal and relative absent words," Journal **Bioinformatics**, 2020. In Press.
15. Rahman, M. S., Alatabbi, A., Athar, T., Crochemore, M.and Rahman, M. S.: "Absent words and the (dis)similarity analysis of DNA sequences: an experimental study," Journal **BMC Research Notes**, vol. 9, p. 186, Mar 2016.
16. Silva, R. M., Pratas, D., Castro, L., Pinho, A. J., and Ferreira, P. J.: "Three minimal sequences found in ebola virus genomes and absent from human DNA," Journal **Bioinformatics**, vol. 31, no. 15, pp. 2421–2425, 2015.
17. Chairungsee, S. and Crochemore, M.: "Using minimal absent words to build phylogeny", Journal **Theoretical Computer Science**, vol. 450, pp. 109–116, Sep 2012.

18. Garcia, S. P. and Pinho, A. J.: "Minimal absent words in four human genome assemblies". Journal **PLoS One**, vol. 6, p. e29344, Dec 2011.
19. Belazzougui, D. and Cunial, F.: "A framework for space efficient string kernels". Journal **Algorithmica**, vol. 79, no. 3, pp. 857–883, 2017.
20. Barton, C., Heliou, A., Mouchard, L. et al: Linear-time computation of minimal absent words using suffix array. Journal **BMC Bioinformatics** 15, 388 (2014). `https://doi.org/https://doi.org/10.1186/s12859-014-0388-9`