B.Sc. in Computer Science and Engineering Thesis

Faster and Improved CD-MAWS with Suffix Automata

Submitted by

Mohammad Tamimul Ehsan
1805022

Supervised by

Dr. Mohammad Saifur Rahman

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

June 2023

CANDIDATES’ DECLARATION

This is to certify that the work presented in this thesis, titled, “Faster and Improved CD-MAWS with Suffix Automata”, is the outcome of the investigation and research carried out by us under the supervision of Dr. Mohammad Saifur Rahman.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Mohammad Tamimul Ehsan
1805022

ACKNOWLEDGEMENT

We are thankful to our thesis supervisor, Dr. Mohammad Saifur Rahman, for his unwavering guidance and support throughout the research journey. Without his invaluable expertise, patience, and encouragement, this thesis would not have come to fruition. We thank our classmate Sk Sabit Bin Mosaddek for helping in writing code and providing valuable insights. We also appreciate the help in providing tools and resources from our seniors including Tanjim Ahmed Khan and Md. Shafqat Talukder. Finally, we express our gratitude to all individuals whose research contributions or resources have directly or indirectly supported our study

Mohammad Tamimul Ehsan

Dhaka
June 2023

Contents

CANDIDATES' DECLARATION **i**

ACKNOWLEDGEMENT **ii**

List of Figures **v**

List of Tables **vi**

List of Algorithms **vii**

ABSTRACT **viii**

1 Introduction **1**

2 Background and Related Works **2**
- 2.1 Background 2
- 2.2 Related Works 2

3 Preliminary Concepts **4**
- 3.1 Phylogeny 4
- 3.2 Multiple Sequence Alignment 4
- 3.3 Alignment-Free Method 4
- 3.4 Minimal Absent Words(MAWs) 5
- 3.5 Composition Vector 5
- 3.6 Cosine Distance for MAW Vectors 6
- 3.7 Length of MAWs 7

4 Proposed Methodology **8**
- 4.1 Two Pointer Method 8
- 4.2 Encoding MAWs 8
- 4.3 Generating MAW using Suffix Automata 9

5 Results **11**
- 5.1 Dataset 11
- 5.2 Comparative Analysis of MAW Generation 12

5.2.1 Execution Time 12
5.2.2 Memory Usage 13
5.3 Comparative Analysis of Cosine Distance Calculation 14
5.3.1 Execution Time 14
5.3.2 Memory Usage 15

6 Discussion **17**

7 Conclusion **18**

References **19**

A Algorithms **21**
A.1 Check MAW 21
A.2 Two Pointer Method 21

List of Figures

3.1 Bulk regions of some species, as studied in [1] 7

4.1 Suffix automaton of ‘ACTGGA’ 9

5.1 Runtime (s) of the original and modified MAW generation method 13
5.2 Memory usage (kB) of the original and modified MAW generation method . . . 14
5.3 Runtime (s) of original and modified distance calculation for different number of Taxa 15
5.4 Memory usage (MB) of original and modified distance calculation for different number of Taxa 16

List of Tables

5.1 Runtime (s) of the original and modified MAW generation method 12
5.2 Memory usage (kB) of the original and modified MAW generation method . . . 13
5.3 Runtime (s) of original and modified distance calculation for different number of Taxa 15
5.4 Memory usage (MB) of original and modified distance calculation for different number of Taxa 16

List of Algorithms

1 Check if a Word is MAW 21
2 Count Number of Equal Elements 22

ABSTRACT

We introduce a refined CD-MAWS method for phylogeny estimation, significantly reducing computational complexity from $\max(O(m^3n), O(m^2n \log n))$ to $\max(O(m^2n), O(mnk))$ while maintaining tree quality. Here, m is the number of species, n is the size of DNA of a species, and k is the maximum MAW length. 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

Chapter 1

Introduction

The study of phylogeny is crucial for understanding evolutionary relationships. The CD-MAWS method [2], an alignment-free approach, has been pivotal in this field but is hampered by 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. The original CD-MAWS method, known for its innovative alignment-free approach to estimating phylogenetic relationships, provided the groundwork for our advancements. By improving the calculation of cosine distances between MAW vectors, introducing suffix automata for MAW generation and storing generated MAWs using hashing techniques and encoding, we reduce computational complexity while maintaining accuracy. This progress opens new avenues for large-scale genomic studies, offering a faster and more scalable tool for unraveling the complexities of evolutionary histories.

Chapter 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. These methods bypass the need for sequence alignment, thus reducing computational complexity and error potential.

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) [3] conducted a comprehensive benchmarking of alignment-free sequence comparison methods, highlighting the strengths and weaknesses of various approaches. This 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 [4], 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) [5], 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 [6] and information-theoretic measures [7] 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 [8] [9] [10] [11] [12] They efficiently distill essential genomic sequence information, enabling comparisons without traditional alignment methods. MAWs support distance-based techniques such as Length Weighted Index [11] and Jaccard Distance [9], which aid in reconstructing phylogenies. Additionally, they facilitate the creation of composition vectors used in cosine similarity computation [13]. The CD-MAWS approach [2] specifically demonstrates their effectiveness through experiments on biological and simulated datasets, showcasing their potential in reconstructing species phylogeny.

In CD-MAWS, the selection of MAW lengths was not guided by objective criteria but influenced by factors like MAW frequency and memory limitations. As the number of MAWs increases, memory constraints can prevent the method from generating vectors needed for cosine distance calculations. In this study, we integrated suffix automata and hashing techniques to address this limitation, achieving improved runtime and memory usage successfully.

Chapter 3

Preliminary Concepts

3.1 Phylogeny

Phylogeny refers to the evolutionary history and relationships among species or other taxonomic groups. These relationships are often depicted as a phylogenetic tree, a branching diagram that illustrates how various species are related through common ancestors. Phylogenetic analysis involves comparing genetic, morphological, or other types of data to infer these relationships and understand how species have evolved over time. Phylogenetic trees help in understanding the divergence of species, tracing the lineage of genes, and studying the patterns of evolution.

3.2 Multiple Sequence Alignment

Multiple Sequence Alignment (MSA) is a method used to align three or more biological sequences (DNA, RNA, or protein) to identify regions of similarity that may indicate functional, structural, or evolutionary relationships. The aligned sequences are arranged in a matrix, where columns contain characters that are homologous (derived from a common ancestor). MSAs are crucial for identifying conserved motifs, inferring phylogenetic relationships, predicting secondary and tertiary structures of proteins, and for various comparative genomics studies. Common tools for performing MSA include Clustal Omega, MUSCLE, and MAFFT.

3.3 Alignment-Free Method

Alignment-Free Methods are techniques used in bioinformatics to compare and analyze biological sequences without the need for sequence alignment. Unlike traditional alignment-based methods, which require aligning sequences to identify similarities and differences, alignment-free methods use alternative strategies such as k-mer frequencies, sequence composition, and other statistical measures to compare sequences. These methods are particularly useful for large-scale sequence analysis, where alignment can be computationally expensive, and for sequences that are highly divergent or contain complex rearrangements. Alignment-free methods are used in various applications, including phylogenetic analysis, genome comparison, and metagenomics.

3.4 Minimal Absent Words(MAWs)

A string y is called a factor or substring of another string x if y is a contiguous sequence of characters within x. If y is a substring of x and length of y is less than x then it y is called a proper substring of x.

A string y which is not present as a substring of x is called a absent word of x.

A prefix y is a substring of x which can be found by deleting several or zero characters from the end of a string. A proper prefix y is a prefix of x whose length is smaller than the string x.

A suffix y is a substring of x which can be found by deleting several or zero characters from the beginning of a string. A proper suffix y is a suffix of x whose length is smaller than the string x.

Minimal Absent Words (MAWs) are words that do not appear in a given DNA sequence, but all their proper substrings are present in x. MAW can also be defined with respect to prefix and suffix. Minimal Absent Words (MAWs) are words that do not appear in a given DNA sequence, but its proper prefix and proper suffix are present in x.

For example, ACG, GAC, and GCT are examples of MAWs for the string $x = ACTGGA$. However, $ACTA$ is an absent word but not a MAW of x.

3.5 Composition Vector

For a sequence x, its vector representation V_x is defined such that $V_x[w]$ indicates whether w is a MAW of x or not. Specifically:

$$V_x[w] = \begin{cases} 1, & \text{if } w \in \text{MAW}(x) \\ 0, & \text{otherwise} \end{cases}$$

The composition vectors are created from the union of MAW set of every sequence.

Let’s consider an example for 3 sequence

x_1 = ”ACTGGA”
x_2 = ”ATGGAC”
x_3 = ”ACGGTG”

And the MAW set for each sequence will be:

MAW(x_1) = “GGG”, “GAC”, “TGA”
MAW(x_2) = “TGA”, “GAT”, “GGG”
MAW(x_3) = “GGG”, “CGT”, “TGT”, “TGG”

Thus the compostion vectors, V_x for each sequence will be:

	GGG	GAC	TGA	GAT	CGT	TGT	TGG
V_{x_1}	1	1	1	0	0	0	0
V_{x_2}	1	0	1	1	0	0	0
V_{x_3}	1	0	0	0	1	1	1

3.6 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.

In the original CD-MAWS, the distance is shown as:

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

So, we can calculate the distance between the composition vectors using this formula. The distance between $V_x 1$ and $V_x 2$ can be calculated as
CD-MAWS(V_{x1}, V_{x2}) = $\frac{1*1+1*0+1*1+0*1+0*0+0*0+0*0}{\sqrt{1^2+1^2+0^2+0^2+0^2+0^2+0^2}*\sqrt{1^2+0^2+1^2 10^2+0^2+0^2+0^2}} = 0.333$

Therefore, the CD-MAWS distance between x_1 and x_2 is 0.333.

3.7 Length of MAWs

Aurell et al. [1] derived equations for calculating the lower and upper range of the bulk region of a sequence of length N assuming a random model:

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

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

The bulk of MAWs can be described using simple statistical properties of the sequence, such as the single nucleotide composition.

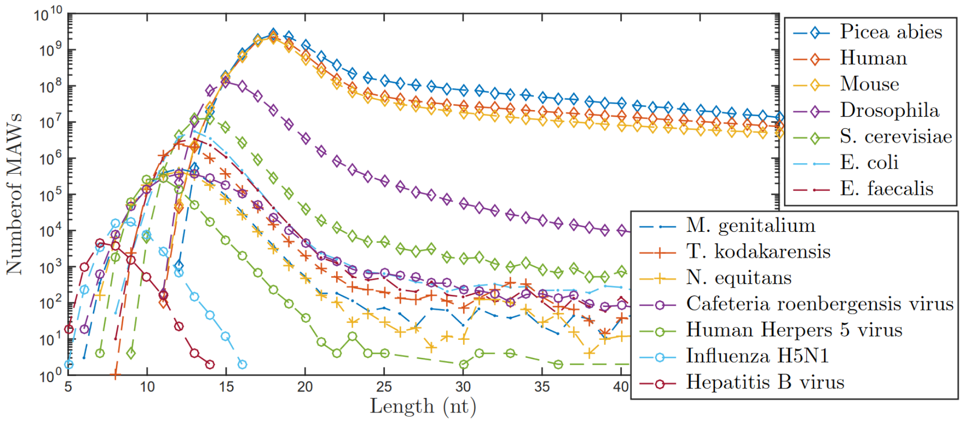

Figure 3.1: Bulk regions of some species, as studied in [1]

For an input size of $N = 10^8$, the bulk range lies between $l_{min} = 15$ to $l_{max} = 29$. Hence, the maximum length of a MAW is assumed to be 30.

Chapter 4

Proposed Methodology

4.1 Two Pointer Method

In the original CD-MAWS method, the proposed approach was to create an augmented vector. In the complexity calculation, the expected size of MAWs was excluded as it was nearly constant. The maximum length of MAWs was considered to be no greater than 30 in any case. This is also a crucial factor for calculation in our proposed method. If the maximum length of a MAW is k, then creating the augmented vector requires a length of $O(mn)$, and each MAW has a length of $O(k)$. Thus, the expected runtime is $O(kmn)$. Comparing each pair would require $O(m^2 \cdot mn)$, i.e., $O(m^3n)$.

However, by examining the structure of the cosine distance, 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 augmenting anything, 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. Please refer to Algorithm 2 for pseudocode of the two pointer method. With our current proposition, this part has a complexity of $O(n \log n)$.

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

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 $O(k)$, where k is their length. In CD-MAWS, the maximum length of MAWs is not greater than 30. We can leverage this information to our advantage. 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 of length not exceeding 60 bits. In many programming languages, including C++, this length can be represented as 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.

If the length of MAW becomes more than that, then we can't use encoding. We propose rolling hash to store the hased MAWs. To decrease the hash collision probability we used double hashing on the MAW set. Thus still maintaining the constant time equality checking irrespective of the length of MAW.
Encoding or double hashing the MAWs requires $O(kmn)$ time, and creating the cosine distance now requires $O(m^2 n \log n)$. Therefore, the total time complexity is $O(kmn + mn \log n + m^2 n)$.

4.3 Generating MAW using Suffix Automata

A suffix automaton is a trie-like structure. In Figure 4.1 the forward links (solid line) act as normal trie and the suffix links or backward links (dashed lines) point to the longest suffix of a node that is not the node itself. Please refer to Algorithm 1 for pseudocode for checking if a word is MAW or not.

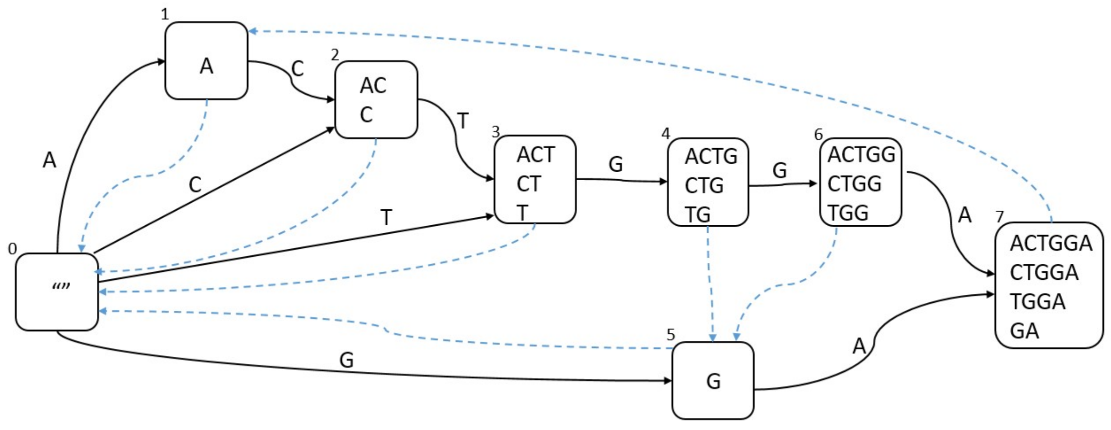

Figure 4.1: Suffix automaton of 'ACTGGA'

For example, consider the word 'x = ACTGGA'. We check if two strings 's = GAC' and 't = TGC' are MAWs or not. For 'GAC', we traverse the trie up to 'GA' to reach node 7. The longest prefix is a substring of the original word, but from there we can't move forward with edge 'C'. Therefore, 's' is not included in 'x'. We then move backward using the suffix link to reach node 1. Here, the length of the word in that node is at least one, making it a candidate for being the 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 ‘TG’ but can’t move forward with edge ‘C’. It matches all criteria so far, but after moving to node 5 following the suffix link, we still can’t move forward with edge ‘C’. 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 Solon et el. [14]. 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. Additionally the encoding of the MAW can be done during this generateion stage.
Thus our current implementation now has a final complexity of $O(kmn + mn + m^2n)$ ie $O(kmn + m^2n)$.

Chapter 5

Results

5.1 Dataset

In our analysis of the running time and memory usage of CD-MAWS, we used PASTA algorithm [15] to generate a phylogenetic tree in Newick format (with branch lengths) and the Seq-gen Monte Carlo simulation tool [16] to generate DNA sequences from that tree. The PASTA tool takes a fasta file as input and produces a tree as output. To create the tree on the 1000L2 R0 dataset from the SATe [5] paper, we have used the following command:

```
python run pasta.py -i rose.aln.true.fasta
```

Here, ”rose.aln.true.fasta” is the fasta file for the 1000L2 R0 dataset. To generate a set of 1000 sequences, each consisting of 2000 base pairs, from the output tree of the PASTA tool, we have used the following command of the Seq-gen tool:

```
seq-gen -mGTR -r0.25, 1, 4, 4, 1.380952381, 0.3452380952 -l2000 -n1
-s1.0 -or input.tree out.phy
```

Here,

- The parameter ”m” is used to indicate the nucleotide model, and in our case, we employed the GTR (Generalized time-reversible) model.
- The parameter ”r” is responsible for specifying the six substitution rate parameters of the GTR model. These parameters include Alpha, Beta, Gamma, Delta, Epsilon, and Eta, representing the rates of nucleotide substitutions between specific pairs.
- The parameter ”l” denotes the desired sequence length.

- The parameter ”n” specifies the number of replicate datasets we want to generate.
- The parameter ”s” allows us to scale the branch lengths in the input tree.
- ”input.tree” is the output of the previously executed PASTA tool
- Finally, the parameter ”or” instructs the command to produce the output file in PHYLIP format, with the name ”out.phy”.

5.2 Comparative Analysis of MAW Generation

In this section we compare the execution time and memory usage of the proposed method with the previous MAW generation algorithm.

5.2.1 Execution Time

Table 5.1 and Figure 5.1 compare the execution times of the original and modified MAW generation algorithms for various input sizes. The modified algorithm shows asymptotic improvements, particularly for large datasets.

Table 5.1: Runtime (s) of the original and modified MAW generation method

Input Size	Original	Sorting	Total	Modified
10	0.0280	0.0065	0.0345	0.3275
100	0.0190	0.0068	0.0258	0.3695
1000	0.0200	0.0090	0.0290	0.3485
10000	0.0295	0.0218	0.0513	0.3553
100000	0.1298	0.1133	0.2430	0.4573
1000000	1.2658	1.0448	2.3105	1.9138
10000000	14.5448	11.1833	25.7280	18.0760

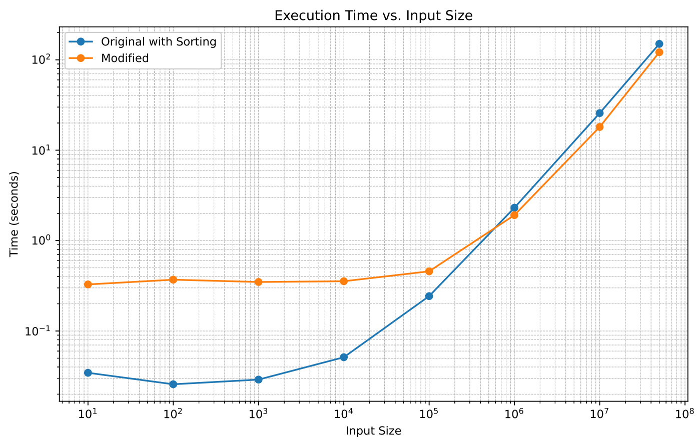

Figure 5.1: Runtime (s) of the original and modified MAW generation method

The maw generation proposed by C. Barton et al [4] and used by CD-MAWS [1] produces the MAW set it random order. Our proposed method requires the MAWs to be sorted. The MAW generation method using suffix automata outputs the MAW set in sorted order. Thus removes the sorting step for two pointer method. For an input size of 10,000,000, where $(\log_2(10,000,000) \approx 23)$, close to the maximum MAW length of 30, our method shows a clear asymptotic advantage as the input size increases.

5.2.2 Memory Usage

The memory usage of both the method is asymptotically same that is O(n) but due to simplicity of our approach the memory footprint is a bit less than the original proposal. The memory consumption of MAW generation is provided in 5.2 and Figure 5.2 summarize the peak memory consumption of previous and current method.

Table 5.2: Memory usage (kB) of the original and modified MAW generation method

Taxa Count	Original	Modified
10	3860	3408
100	3916	3444
1000	3932	3512
10000	4572	3848
100000	16516	12660
1000000	259336	80780
10000000	2538740	635872

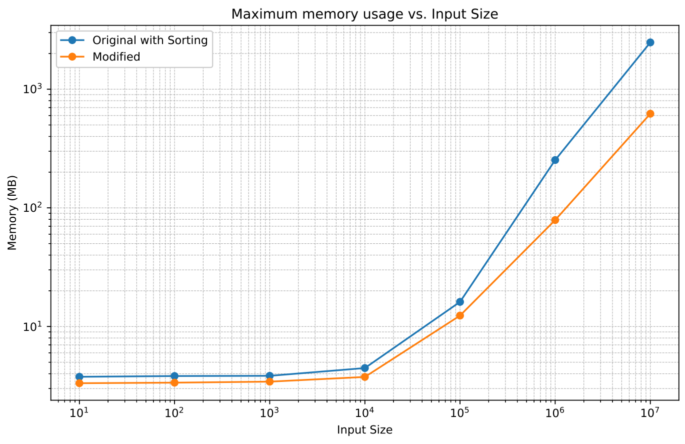

Figure 5.2: Memory usage (kB) of the original and modified MAW generation method

5.3 Comparative Analysis of Cosine Distance Calculation

Previous CD-MAWs implementation introduced a method of augmented 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. The proposed two-pointer method and encoding techniques significantly reduce the runtime and memory footprint compared to the original CD-MAWS method.

5.3.1 Execution Time

By employing a two-pointer method on the sorted MAW set, we can identify common MAWs between two species in linear time, significantly reducing the runtime. However, comparing two MAWs directly requires string comparison, which depends on the length of the MAWs. To address this, we introduced a method to encode the MAWs into 64-bit integers, making the comparison a constant-time operation, independent of the MAW length. This encoding can be performed during the generation stage as proposed earlier. For longer MAWs, we utilize hashing, and to minimize the probability of hash collisions, we employed double hashing. Despite the use of hashing, equality checking is performed in constant time. A comparative runtime analysis of the original CD-MAWs method and our proposed method can be found in Table 5.3. The corresponding graph of this data is presented in Figure 5.3.

Table 5.3: Runtime (s) of original and modified distance calculation for different number of Taxa

Taxa Count	Original	Modified	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

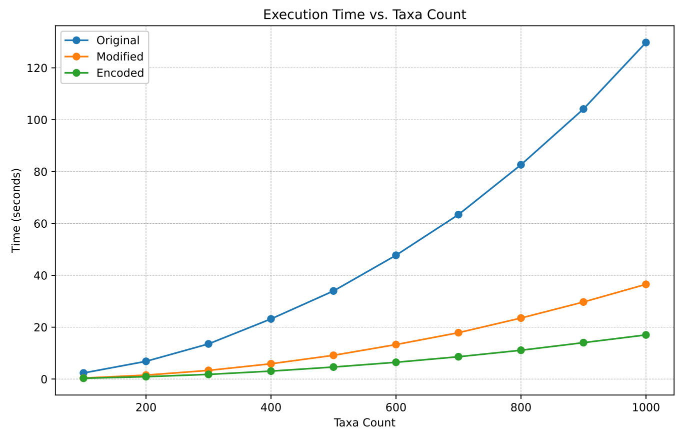

Figure 5.3: Runtime (s) of original and modified distance calculation for different number of Taxa

5.3.2 Memory Usage

The augmented vector in CD-MAWs generated heavy memory usage in past implementations. By only working with the MAWs, we have significantly decreased the memory footprint. Additionally, encoding or hashing the MAWs further reduces memory usage. The memory usage comparison is provided in Table 5.4, with the corresponding graph presented in Figure 5.4.

Table 5.4: Memory usage (MB) of original and modified distance calculation for different number of Taxa

Taxa Count	Original	Modified	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

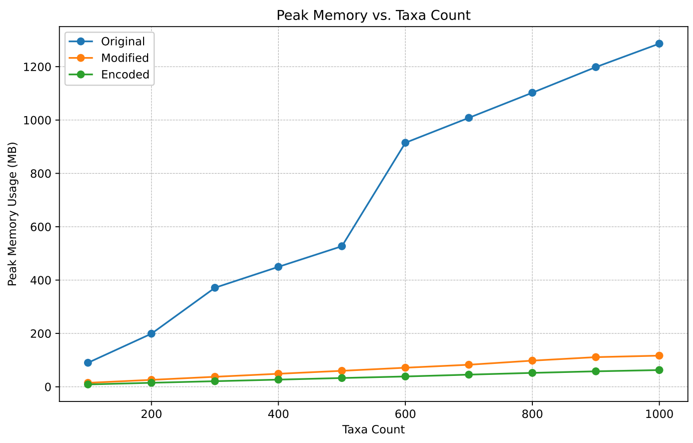

Figure 5.4: Memory usage (MB) of original and modified distance calculation for different number of Taxa

Chapter 6

Discussion

The enhanced CD-MAWS method presented in this study significantly improves computational efficiency and accuracy in phylogenetic analysis. By refining the cosine distance calculation, encoding MAWs in binary, and utilizing suffix automata for MAW generation, we have reduced the computational complexity from $O(mn^2 \log n)$ to $O(mn \log n)$. 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 accuracy and scalability of phylogenetic estimations.

However, our study has limitations. The assumption of a maximum MAW length of 30, while practical, may not be optimal for all datasets. Future work could explore adaptive approaches to determining the appropriate MAW length based on the specific characteristics of the data. 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.

Chapter 7

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.

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.

References

[1] E. Aurell, N. Innocenti, and H.-J. Zhou, “The bulk and the tail of minimal absent words in genome sequences,” *Physical Biology*, vol. 13, p. 026004, apr 2016.

[2] N. Anjum, R. L. Nabil, R. I. Rafi, M. S. Bayzid, and M. S. Rahman, “Cd-maws: An alignment-free phylogeny estimation method using cosine distance on minimal absent word sets,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 20, no. 1, pp. 196–205, 2023.

[3] A. Zielezinski, H. Z. Girgis, G. Bernard, C.-A. Leimeister, K. Tang, T. Dencker, A. K. Lau, S. Röhling, J. J. Choi, M. S. Waterman, M. Comin, S.-H. Kim, S. Vinga, J. S. Almeida, C. X. Chan, B. T. James, F. Sun, B. Morgenstern, and W. M. Karlowski, “Benchmarking of alignment-free sequence comparison methods,” *Genome Biology*, 2019.

[4] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy, “Mash: fast genome and metagenome distance estimation using minhash,” *Genome Biology*, 2016.

[5] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow, “Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees,” *Science*, vol. 324, no. 5934, pp. 1561–1564, 2009.

[6] M. Abeysundera, C. Field, and H. Gu, “Phylogenetic Analysis Based on Spectral Methods,” *Molecular Biology and Evolution*, vol. 29, pp. 579–597, 08 2011.

[7] Z. Liu, J. Meng, and X. Sun, “A novel feature-based method for whole genome phylogenetic analysis without alignment: Application to hev genotyping and subtyping,” *Biochemical and Biophysical Research Communications*, vol. 368, no. 2, pp. 223–230, 2008.

[8] M. Akon, M. Akon, M. Kabir, M. S. Rahman, and M. S. Rahman, “ADACT: a tool for analysing (dis)similarity among nucleotide and protein sequences using minimal and relative absent words,” *Bioinformatics*, vol. 37, pp. 1468–1470, 11 2020.

[9] M. S. Rahman, A. Alatabbi, T. Athar, M. Crochemore, and M. S. Rahman, "Absent words and the (dis)similarity analysis of dna sequences: an experimental study," *BMC Research Notes*, 2016.

[10] R. M. Silva, D. Pratas, L. Castro, A. J. Pinho, and P. J. S. G. Ferreira, "Three minimal sequences found in Ebola virus genomes and absent from human DNA," *Bioinformatics*, vol. 31, pp. 2421–2425, 04 2015.

[11] S. Chairungsee and M. Crochemore, "Using minimal absent words to build phylogeny," *Theoretical Computer Science*, vol. 450, pp. 109–116, 2012. Implementation and Application of Automata (CIAA 2011).

[12] S. P. Garcia and A. J. Pinho, "Minimal absent words in four human genome assemblies," *PLOS ONE*, vol. 6, pp. 1–11, 12 2011.

[13] D. Belazzougui and F. Cunial, "A framework for space-efficient string kernels," *Algorithmica*, 2017.

[14] C. Barton, A. Heliou, L. Mouchard, and S. P. Pissis, "Linear-time computation of minimal absent words using suffix array," *BMC Bioinformatics*, 2014.

[15] K. Collins and T. Warnow, "PASTA for proteins," *Bioinformatics*, vol. 34, pp. 3939–3941, 06 2018.

[16] A. Rambaut and N. C. Grass, "Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees," *Bioinformatics*, vol. 13, pp. 235–238, 06 1997.

Appendix A

Algorithms

A.1 Check MAW

Algorithm 1 Check if a Word is MAW

```
Require: Suffix Automaton $S$ , Word $w$
  $x \leftarrow$ root of $S$
  for $i$ in range(len(w)) do
    $c \leftarrow w[i]$
    $x \leftarrow S[x].ForwardLink(c)$
    if $x$ is $NULL$ then
      return $False$
    end if
  end for
  if $S[x].ForwardLink(c)$ is not $NULL$ then
    return $False$
  else if $S[x].BackwardLink(c)$ is $NULL$ then
    return $False$
  end if
  if $len(S[x]) \leq len(w) - 1$ then
    return $False$
  end if
  return $True$
```

A.2 Two Pointer Method

Algorithm 2 Count Number of Equal Elements

```
Require: Two sorted lists $A$ and $B$
Ensure: The number of equal elements between $A$ and $B$
 1: Initialize pointer $i \leftarrow 0$
 2: Initialize pointer $j \leftarrow 0$
 3: Initialize count $c \leftarrow 0$
 4: while $i <$ length$(A)$ and $j <$ length$(B)$ do
 5:     if $A[i] = B[j]$ then
 6:         $c \leftarrow c + 1$
 7:         $i \leftarrow i + 1$
 8:         $j \leftarrow j + 1$
 9:     else if $A[i] < B[j]$ then
10:         $i \leftarrow i + 1$
11:     else
12:         $j \leftarrow j + 1$
13:     end if
14: end while
15: return $c$
```