A comparative study of maximal and closed sequential pattern mining algorithms on proteomic sequences in the Fungal kingdom

Victor Terrón-Macias[1], Jesús Ariel Carrasco-Ochoa[2*], José Francisco Martínez-Trinidad[2*], Jezreel Mejía[1*]

[1] Centro de Investigación en Matemáticas, Parque Quantum, Ciudad del Conocimiento, Av. Lassec, Andador Galileo Galilei, Manzana, 3, Lote, 7, 98160, Zacatecas, ZAC, México
{victor.terron, jmejia}@cimat.mx

[2] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, México
{ariel, fmartine}@inaoep.com

Abstract. Advancements in data mining algorithms have significantly progressed the discovery of complex patterns within biological sequences, aiding in understanding genetic bases and detecting MOTIFS. While these algorithms have been extensively applied to general biological data, a research gap exists in the sequential pattern mining of fungal proteomic sequences. This study addresses this gap by evaluating the performance of three key algorithms—MFPS, FP-Max, and BIDE—focusing on their efficiency in mining maximal and closed sequential patterns from fungal proteomic data.

We selected these algorithms for their ability to provide a compact representation of data, aiming to compare their performance in terms of runtime (the time taken to execute the algorithm), memory usage (the amount of computer memory required to run the algorithm), and the quantity and length of patterns identified. Using a dataset of Lachancea meyersii CBS 8951, we conducted experiments with various minimum support thresholds to measure each algorithm's efficiency and resource consumption.

Our findings highlight the trade-offs between time and memory consumption across different algorithms. MFPS was found to be time-intensive but mid-term in memory usage. FPMAX used mid-term execution times and mid-time consumption memory. BIDE demonstrated faster execution times but higher memory usage.

This comparative analysis not only provides insights into the suitability of maximal and closed sequential pattern mining algorithms for fungal proteomic sequences but also directly guides researchers in selecting the most appropriate tools based on their specific needs and constraints. Future research could explore optimization strategies for these algorithms and extend their application to larger datasets, thereby enhancing their utility in computational biology and proteomics.

Keywords: Maximal Sequential and Closed Sequential Pattern Mining, Proteomic Sequence Analysis, Data Mining Algorithm Performance Evaluation.

1 Introduction

Advancements in data mining algorithms have made significant progress in discovering complex patterns within biological sequences. For example, one application is helping to advance the understanding of the genetic basis [1]; another application is the detection of MOTIFS, which are short repetitive patterns within DNA hypothesized to possess a biological function [2, 3, 4].

The large-scale study of proteins is of utmost importance in understanding the functional aspects of biological systems. Analyzing these sequences offers invaluable insights into cellular processes, potential drug targets, and evolutionary relationships [5]. However, a significant research gap exists in the analysis of sequential pattern mining for fungi, a gap that is crucial for deciphering their biological functions, interactions, and potential applications in various fields [6].

From the literature, we have identified the most effective algorithms for mining frequent sequences that could be used to analyze proteins [1]. These include the MFPS algorithm used to find maximal sequential patterns [7], FP-Max used for extraction of maximal frequent sequences [8], BIDE used for extraction of closed frequent patterns [9], FP-Growth used to extract frequent itemsets in data sets containing itemsets [10], ECLAT focused on detecting frequent itemsets [11] and SPADE for detection of sequential patterns. However, our research will focus on the algorithms that mine maximal sequential (MFPS and FP-Max algorithms) and closed patterns (BIDE algorithm) due to their proven ability to provide a more compact representation of the data compared to mining all frequent sequences [9] like other algorithms do.

This research compares the performance of algorithms for mining maximal and closed patterns applied to fungal sequences. The primary aim is to evaluate time efficiency and memory usage when applied to proteomic sequence data for pattern identification. This study enables users to pinpoint the algorithm that best aligns with their requirements, particularly regarding time and memory consumption.

This paper is structured as follows: section two describes the pattern-finding algorithms selected for the comparative study, section three describes the experimental evaluation, and section 4 presents our conclusions.

2 Pattern Mining Algorithms

Mining closed sequences and maximal frequent sequences are both techniques used in sequence mining, which involves discovering interesting subsequences from sequential data. However, they have different objectives and yield different results.

- Closed sequences: refer to sequences with no super sequences with the same support. The primary goal of mining closed sequences is to identify compact and non-redundant patterns that are significant in the dataset. These patterns provide concise representations of frequent subsequences, reducing redundant information. Closed sequence mining produces fewer patterns than traditional frequent sequence mining, as it focuses on eliminating redundancy.

- Maximal Frequent Sequences: these sequences cannot be extended further without becoming non-frequent. The objective of mining maximal frequent sequences is to identify all the locally longest frequent subsequences in the dataset. These sequences provide a comprehensive view of the frequent patterns present in the data, including both sequential and non-sequential relationships. Maximal Frequent Sequences produce fewer patterns than closed sequences mining.

As mentioned before, the literature indicates that several algorithms have been proposed for mining frequent sequences; nevertheless, this research will focus only on the most popular and successful algorithms that mine maximal sequential (MFPS and FP-Max algorithms) and closed sequences (BIDE algorithm) due to their ability to provide a more compact representation of the data compared to mining all frequent sequences [5] like other algorithms do.

By assessing these algorithms, the study aims to identify the most efficient algorithm for pattern identification in proteomic sequences of fungal organisms. To our knowledge, no similar study exists on this type of proteomic sequence in the literature. A brief description of each selected algorithm is presented in the following sections.

In fungal sequence analysis, all compared algorithms could be used to identify essential MOTIFS or sequences within DNA or protein sequences. These MOTIFS may represent functional elements such as binding sites, active sites, or regulatory regions [6].

It is noted that, even though the algorithms under comparison mine frequent sequences of the same nature, some do not establish a minimum length for the identified sequences. Instead, these algorithms identify all frequent sequences of the specific type the algorithm search. This variability in the size of the extracted frequent sequences directly affects the quantity of sequences mined. In the context of SEQMINE CI, the algorithms enabled to adjust the desired minimum length are PrefixSpan (excluded from the present comparative study) and BIDE.

2.1 MFPS (Mining Maximal Frequent Patterns)

MFPS is designed to find maximal sequential patterns within sequences [7, 8], utilizes suffix trees, a data structure commonly used in string processing, to identify sequential patterns efficiently, and identifies maximal frequent patterns, which cannot be extended further to find frequent sequences. This ensures that only the longer patterns are retained, reducing redundancy [9].

MFPS uses the pruning rule m*aximal pattern property*. This property states that if an itemset is not frequent, it cannot be part of any frequent pattern set [10]. Thus, when exploring candidate itemsets, if it's determined that an itemset is not maximal frequent, there's no need to explore its supersets further, as they cannot be maximal frequent either.

In our implementation this algorithm does not allow the restriction of the minimum length of the found sequences.

2.2 FP-Max

It focuses on mining maximal itemsets occurring in a sequence dataset.

The FP-Max builds a special data structure called Maximal Frequent Itemset Tree to store all maximal frequent itemsets [11].

FP-Max algorithm initially, the database undergoes scanning, leading to the construction of an FP-tree denoted as T. Subsequently, a linked list named Head is generated for each item in the header table, commencing from the last item. Consequently, the Head encompasses the items constituting the conditional pattern base of the item under consideration. If T comprises a solitary path denoted as P, the union of Head and P (Head U P) is incorporated into the MFI-Tree. Conversely, if T does not represent a single-path tree, the item within the header table is appended to the Head. Subsequently, an assessment is made to determine if the new Head forms a subset of any existing maximal frequent itemsets. If not, FPMAX is invoked recursively [12].

FP-Max employs several pruning rules. Firstly, it utilizes *subset infrequency pruning*: Any candidate itemset that contains an infrequent subset is pruned, meaning it will not be considered for further analysis. Then, FP-Max uses *Lookahead Superset Frequency Pruning*: FP-Max performs a lookahead to identify all single items, denoted as j, that are not present in the frequent itemset X but can be added to X to form a frequent itemset $X \cup \{j\}$. These identified items form set Y. If $X \cup Y$ is found to be frequent, it implies that any subset of $X \cup Y$ is also frequent. Consequently, such subsets can be pruned from further consideration. Also, FP-Max applies *maximality pruning*: During the mining process, if it is determined that the subgroups of a frequent itemset are already non-maximal, FP-Max refrains from examining them further [12].

In our implementation, this algorithm does not allow the restriction of the minimum length of the found sequences.

2.3 Closed Sequential Pattern Mining (BIDE)

BIDE is an algorithm mining closed frequent patterns from sequence databases [5]. This algorithm efficiently identifies closed patterns and subsequences that cannot be extended without decreasing their support count. Unlike traditional frequent pattern mining algorithms, BIDE focuses on discovering closed patterns to provide a more compact and informative dataset representation [5].

Firstly, BIDE utilizes *anti-monotonicity*, which dictates that if a sequence is not frequent, any extension of this sequence is also non-frequent. Consequently, the algorithm discards extensions of infrequent sequences from further consideration. Another technique employed is s*equence merge pruning*, which identifies sequences sharing identical prefixes and eliminates the infrequent ones, thereby reducing redundancy and streamlining the search process. Additionally, *Subsequence Pruning* is implemented, whereby if a subsequence of a sequence is not frequent, the sequence itself is non-frequent. Thus, the algorithm prunes sequences containing infrequent subsequences, further optimizing its performance [8].

In our implementation, this algorithm restricts the minimum length of the found sequences; this parameter value is established to 2 in the experimentation for this algorithm.

3 Comparative Study

In this section, we delve into the comparative study of mining sequential patterns of fungi proteomic data employing the algorithms described in section 2. We document each algorithm's runtime and record memory consumption using specialized libraries like time and psutil.

3.1 Metrics to measure: time and memory consumption

The discovery of frequent itemsets suffers from two main obstacles: (1) the high memory consumption [13, 14] and (2) the large runtime that it takes for each algorithm to mine the sequences [14]. Another parameter to consider is the number of sequences each algorithm produces as output; this is because, although the study focuses on the amount of time and memory used by the algorithm, the number of sequences indicates the algorithm's efficiency and sensitivity [15] and give us a metric to compare its performance, helpful when choosing between different algorithms and according to the nature of the data and task.

Comparing memory consumption, processing time, and the number of patterns found by the algorithms mining proteomic sequences serves different purposes:

- Resource Efficiency: Memory consumption directly impacts the scalability and resource utilization of the algorithms; if the algorithm consumes a large amount of memory, it potentially generates a *"system crash"* [14]. Lower memory consumption implies the algorithm can operate efficiently even on systems with limited resources, depending on whether the algorithm processes the information in a distributed way [16].
- Performance Benchmarking: Processing time is a critical factor in determining algorithms' efficiency [17]. Shorter processing times often indicate faster results, which is essential when dealing with large datasets.
- Algorithm Selection: We can decide which algorithm performs the best by comparing memory consumption and processing time. If we prioritize speed, we might opt for the algorithm with the shortest processing time, even if it consumes slightly more memory. Conversely, if memory resources are constrained, we might prioritize an algorithm with lower memory consumption, even if it takes somewhat longer to process the data.

Overall, comparing memory consumption and processing time provides comprehensive insights into the performance and suitability of those algorithms. A higher number of sequences may imply a greater diversity of patterns and insights that can be extracted from the dataset. However, it's also essential to consider the quality and relevance of these sequences, as generating many sequences doesn't necessarily guarantee valuable insights.

3.2 Origin of information for the experiment

For our study, we conducted an experiment selecting a species of the Fungal kingdom: the Lachancea meyersii CBS 8951 dataset [21], downloaded using the Joint Genome Institute MycoCosm Portal [22]. This dataset has the characteristics indicated in Table 1.

Table 1. Characteristics of Lachancea meyersii CBS 8951 dataset.

Parammeter	Value
Number of sequences	4998
Average length	490.22
Date of Download MycoCosm Portal	April 11, 2020.
Database Origin	Ensembl Fungi.

3.3 Experimental Evaluation

We perform experiments on a desktop computer with a Ryzen 5600H processor, 24GB of memory, and 2TB SSD. The operating system was Windows 11 Pro, and we used Python as the programming language to develop a web application with the Django framework. The application's name is SEQMINE CI.

All reported times are accurate, obtained at the beginning and end of the algorithm execution, as received from the Python time library. We also recorded the memory consumption for each algorithm using the *psutil* Python library; it is clarified that only this application was executed during application executions, and unnecessary background processes were killed. Fig. 1 shows the user interface of SEQMINE CI.

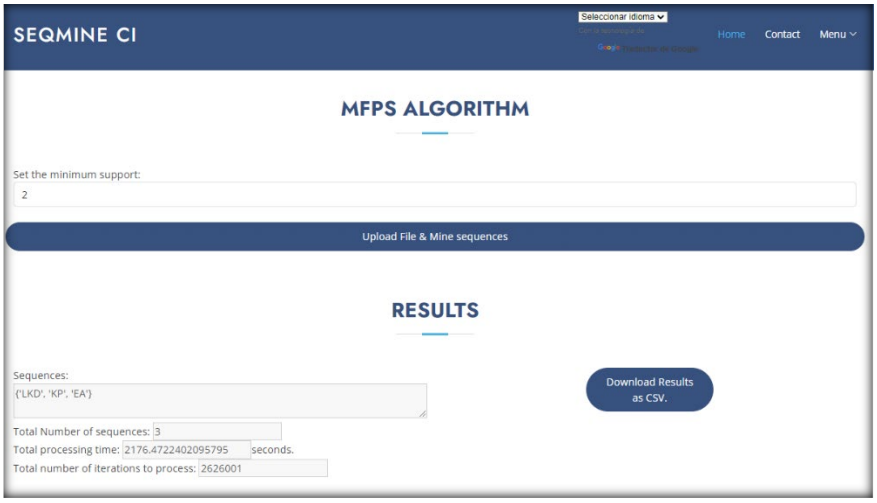

Fig. 1. UI of SQMINE CI.

All mined sequences were stored in a CSV file using the pandas library to a file to allow the user to download and analyze them.

For the experiments, each algorithm was allocated a maximum of 30 minutes to finish execution, after which it was killed. This is to finish the evaluation in a reasonable time. Although SEQMINE CI has available algorithms like PrefixSpan and ECLAT, these algorithms were not included in this research since they mine all sequential patterns (too many). Therefore, this study, only includes algorithms that mine maximal or close sequential patterns as mentioned before.

Memory management becomes critical for this research, mainly when dealing with large datasets. This study selected a dataset with 4998 sequences, with an average length of 490.22. Despite possessing a memory capacity of 24GB, attempting to analyze the entire dataset leads to memory overflow issues due to the volume of data. Consequently, a pragmatic approach is adopted whereby only the first 175 sequences are selected for analysis. Although even this subset challenges memory limits, the computational resources available are adequate to proceed with the analysis and yield meaningful results, as described in the next section.

3.4 Mining Maximal Frequent Itemsets and Closed Frequent Itemsets

In this section, we show the results for mining maximal frequent itemsets and closed frequent itemsets. In each experiment with the algorithms, we adjusted the minimum support threshold with four different values, keeping the same minimum support value for each experiment in all algorithms.

The minimum support threshold is defined by the user, this parameter defines the number of times a sequence must appear to be considered a pattern. A lower support value will lead to the detection of more sequences, with a higher probability of finding longer sequences. Also, this implies that the time and memory consumption will increase. Some algorithms, such as MFPS, set this threshold to an absolute value, meaning that a sequence is considered a pattern if it appears a certain number of times. On the other hand, FP-Max sets it to a relative scale, for example, 0.01, indicating that the sequence is considered a pattern if it appears in at least 1% of the data set. BIDE also uses a relative scale, but since it focuses on closed sequences, it also allows setting the maximum size of the sequence to be found. This leads us to establish an equivalence between the absolute and relative value of the minimum support, for which we first need to know the total number of sequences in the dataset (N). We establish Eq. 1 and Eq. 2, where the variable MST is an abbreviation of $Minimum\ Support\ Threshold$.

$$Relative\ MST = \frac{Absolute\ MST}{N} * 100\% \qquad (Eq.1)$$

$$Absolute\ MST = \frac{Relative\ MST}{100\%} * N \qquad (Eq.2)$$

Once the equivalence between Absolute MST and Relative MST was established, we performed the experiments; these results are presented in Fig. 2 and described below. We used the first 175 proteomic sequences of the fungi dataset for all the compared algorithms.

MOTIFS are usually short, ranging from 5 to 20 characters [2, 18]; given this characteristic of a MOTIF, it was determined that the value for the maximum length parameter in the BIDE algorithm was 6.

MFPS Algorithm

- With a minimum support threshold of 2:
  - Time: 88.31 seconds.
  - Memory consumption: 10.1 GB.
- With a minimum support threshold of 3:
  - Time: 73.42 seconds.
  - Memory consumption: 11.2 GB
- With a minimum support threshold of 4:
  - Time: 91.48 seconds.
  - Memory consumption: 11.5 GB.
- With a minimum support threshold of 5:
  - Time: 74.87 seconds.
  - Memory consumption: 11.4 GB.

FP-Max Algorithm

- With a minimum support threshold of 2:
  - Time: 76.13 seconds.
  - Memory consumption: 1.6 GB.
- With a minimum support threshold of 3:
  - Time: 58.42 seconds.
  - Memory consumption: 1.7 GB
- With a minimum support threshold of 4:
  - Time: 45.42 seconds.
  - Memory consumption: 1.7 GB.
- With a minimum support threshold of 5:
  - Time: 38.59 seconds.
  - Memory consumption: 1.8 GB.

BIDE Algorithm

- With a minimum support threshold of 2:
  - Time: 183.41 seconds.
  - Memory consumption: 1.2 GB.
- With a minimum support threshold of 3:

    - Time: 169.58 seconds.
    - Memory consumption: 1.2 GB
- With a minimum support threshold of 4:
    - Time: 157.94 seconds.
    - Memory consumption: 1.2 GB.
- With a minimum support threshold of 5:
    - Time: 172.11 seconds.
    - Memory consumption: 1.2 GB.

Maximal vs. Closed Algorithms

The results of the algorithms that mine maximal and closed frequent sequences reveal that the total number of sequences found differs between each type of algorithm. It is important to remember that the differences between maximal and closed sequences have been presented in Section 2. The results are described below:

- With a minimum support threshold of 2:
    - Number of found sequences.
        - Maximal: 29 sequences.
        - Closed: 57 sequences.
    - Max length of found sequences:
        - Maximal: 5.
        - Closed: 5.
- With a minimum support threshold of 3:
    - Number of found sequences.
        - Maximal: 18 sequences.
        - Closed: 29 sequences.
    - Max length of found sequences:
        - Maximal: 4.
        - Closed: 4.
- With a minimum support threshold of 4:
    - Number of found sequences.
        - Maximal: 20 sequences.
        - Closed: 20 sequences.
    - Max length of found sequences:
        - Maximal: 3.
        - Closed: 3.
- With a minimum support threshold of 5:
    - Number of found sequences.
        - Maximal Algorithms: 10 sequences.
        - Closed Algorithm: 18 sequences.
    - Max length of found sequences:
        - Maximal algorithms: 3.
        - Closed Algorithm: 3.

The results show that the number of found sequences decreases as the minimum support threshold grows because a higher support threshold value requires more oc-

currences of the sequence in the data to be considered frequent. The mined close frequent sequences are more frequent than the mined maximal frequent sequences.

These results are presented in Fig. 2, in which two graphs are placed at the top of the figure. The one on the left presents the memory consumption with different threshold values, and the one on the right presents the consumption in time with different threshold settings. The lower left corner shows the size of the sequences found with their respective threshold values, and finally, the lower right corner shows the total number of sequences found with the different algorithms.

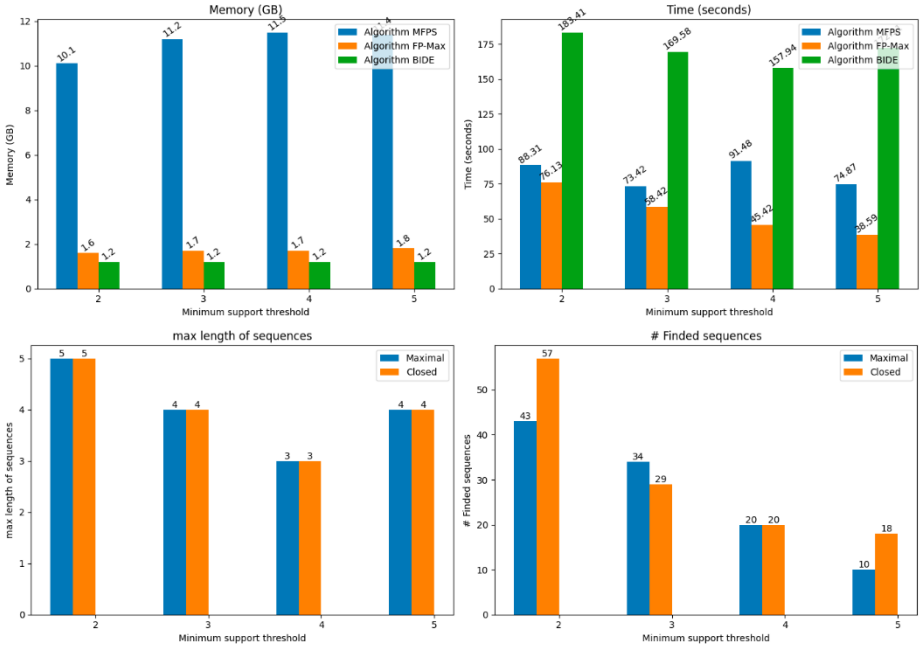

Fig. 2. Comparison between the evaluated algorithms.

Based on these results, we calculated the average memory and time consumption with all the thresholds for each algorithm (see Fig. 3). The X axis of the plots presents the algorithms, and the Y axis presents the average of the parameters.

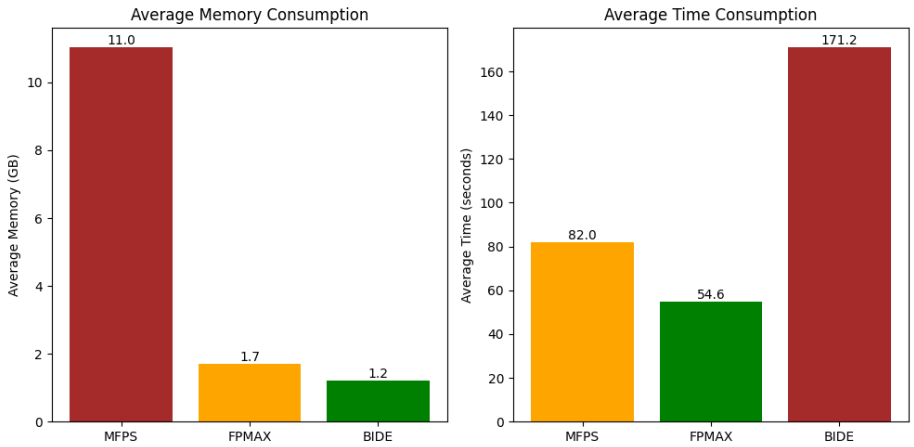

Fig. 3. Average Memory vs Time Consumption of Algorithms.

Fig. 3 highlights that the MFPS algorithm consumes the most time. On the other hand, the BIDE algorithm consumes the most memory compared to the other algorithms. The FP-Max algorithm follows closely, which consumes 1.7GB of memory and respects the time consumption, which takes an average of 54.6 seconds.

Based on the results, we recommend using the MFPS algorithm when mining maximal sequences, and memory consumption is not a significant factor. FP-Max is recommended when mining maximal sequences and memory is limited. Finally, BIDE is recommended if mining closed sequences is desired, particularly when memory resources are crucial, and the user has time availability.

4 Conclusions

In this study, we conducted a comparative analysis of maximal and closed sequential pattern mining algorithms applied to proteomic sequences within the Fungal kingdom, intending to provide the user with a vision of which algorithm should be selected according to what they want to achieve and the establishment of priorities regarding time and memory consumption mainly.

Our investigation focused on evaluating the performance of MFPS, FP-Max, and BIDE algorithms in terms of runtime, memory usage, and pattern identification capabilities. Through the experimentation, distinct characteristics and outcomes were observed for each algorithm.

The comparative analysis provides valuable insights into the performance and suitability of maximal and closed sequential pattern mining algorithms for proteomic sequence analysis in the Fungal kingdom. While each algorithm has strengths and limitations, understanding its characteristics enables researchers to make informed decisions based on specific requirements and constraints. Further research could explore optimization strategies to improve efficiency and explore other types of mining

algorithms for larger datasets, thereby advancing their utility in computational biology and proteomics research.

5 Acknowledgements

The first author thanks Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCyT) of Mexico for his MSc. scholarship.

References

[1] T. T. B. C. G. A. C. G. a. C. S. G. Graciela H. Gonzalez, «Recent Advances and Emerging Applications in Text and Data Mining for Biomedical Discovery,» de *Brief Bioinform*, BMC, 2016, pp. 33-42.

[2] P. D'haeseleer, «What are DNA sequence motifs?,» *Nature Biotechnology,* nº 24, pp. 423-425, 2006.

[3] M. S. M. W. A.-A. Fatma A. Hashim, «Review of Different Sequence Motif Finding Algorithms,» *Avicenna J Med Biotechnologyy,* vol. 11, nº 2, pp. 130-148, 2019.

[4] T. L. Bailey, «STREME: accurate and versatile sequence motif discovery,» *Bioinformatics,* vol. 37, nº 18, pp. 2834-2840, 2021.

[5] J. H. Jianyoung Wang, «BIDE: Efficient Mining of Frequent Closed Sequences,» 2004, pp. 1-12.

[6] N. B. B. R. H. K. S. d. Z. R. E. P. F. N. D.-Q. D. Y. L. N. S. J. K. T. C. B. Nalin N. Wijayawardene, «OMICS and Other Advanced Technologies in Mycological Applications,» de *J Fungi (Basel)*, Journal of Fungi MDPI, 2023, p. 688.

[7] U. Y. Gangin Lee, «Analysis of Recent Maximal Frequent Pattern Mining Approaches,» de *Advances in Computer Science and Ubiquitous Computing 2016*, Singapore, SpringerLink, 2016, pp. 873-877.

[8] Y. W. C. Z. Y. S. Jingsong Zhang, «Mining Contiguous Sequential Generators in Biological Sequences,» de *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2016, pp. 855-867.

[9] J. F. M.-T. J. A. C.-O. René A. García-Hernández, «A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text,» *CIARP,* pp. 478-486, 2004.

[10] J. F. M.-T. J. A. C.-O. René Arnulfo García-Hernández, «A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection,» de *Computational Linguistics and Intelligent Text Processing*, D. Hutchison, Ed., Ciudad de México, México: SpringerLink, 2006, pp. 514-523.

[11] P. G. Anshu Singla, «A Comprehensive Study and Analysis Of Frequent Itemsets Mining Algorithms Using Diverse Real Datasets,» *International Conference on Advances in Computation, Communication and Information Technology,* pp. 174-180, 2023.

[12] Y. B.Ziani, «Mining Maximal Frequent Itemsets: A java implementation of FPMAX Algorithm,» *International Conference on Innovations in Information Technology,* pp. 330-334, 2009.

[13] M. J. Z. Bart Goethals, «Workshop on Frequent Itemset Mining Implementations,» *FIMI'03,* pp. 1-13, 2003.

[14] J.-L. H. M.-S. C. Kun-Ta Chuang, «Mining top-k frequent patterns in the presence of the memory constraint,» *The VLDB Journal,* vol. 17, pp. 1321-1344, 2008.

[15] M. K. J. P. Jiawei Han, Data Mining, Amsterdam: Elsevier, 2016.

[16] Z. B.-J. SAKET NAVLAKHA, «Distributed Information Processing in Biological and Computational Systems,» *COMMUNICATIONS OF THE ACM,* vol. 58, nº 1, pp. 94-102, January 2015.

[17] C. E. L. R. L. R. C. S. Thomas H. Cormen, Introduction to Algorithms, Massachusets: MIT Press, 2022.

[18] M. A.-H. G. D.-C. M. S.-B. S. C. S. C. G. Rogelio Alcántara-Silva, «PISMA: A Visual Representation of Motif Distribution in DNA Sequences,» *Bioinform Biol Insights,* vol. 11, 2017.