DNA Sequencing and Fuzzy Searching

Exploring Fuzzy Searching Algorithms in Bioinformatics

Ej Boakye, Alex Sheckells, Riley Simpson, Jacob Sisson

University of North Carolina Wilmington

Department of Computer Science

May 2024

ABSTRACT

Bioinformatics is an interdisciplinary field that combines biology, computer science, mathematics, and statistics to analyze and interpret biological data. This field is vital in modern biology and medicine, aiding in the deciphering of genetic codes, understanding diseases at a molecular level, and developing new therapies and drugs. A key aspect of bioinformatics is the analysis and comparison of biological sequences, such as DNA, RNA, or proteins, which are essential for understanding the genetic basis of life and disease. DNA sequences are represented as strings of characters (A, T, C, G) that encode genetic information. This paper explores three algorithms used in DNA sequence analysis: the Smith-Waterman algorithm for local sequence alignment, BLAST for rapid sequence similarity searches, and the Approximate Boyer-Moore algorithm for approximate string matching. These algorithms address challenges in bioinformatics, including identifying regions of high similarity between sequences and efficiently searching large databases for matching sequences, even with variations or errors, through the concept of fuzzy searching.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s)

Introduction

This paper focuses on three fuzzy search algorithms that play a vital role in sequence analysis within the field of bioinformatics. The Smith-Waterman algorithm is a dynamic programming approach designed to find the optimal local alignment between two sequences, highlighting the most similar region or substring within the sequence. It allows for gaps and mismatches, providing a detailed comparison essential for identifying functional or evolutionary relationships. BLAST (Basic Local Alignment Search Tool) is a heuristic algorithm that rapidly searches a database of sequences to find regions of similarity to a query sequence. BLAST is instrumental in annotating genes, identifying homologous sequences, and exploring the functional and evolutionary relationships between sequences. The Approximate Boyer-Moore algorithm, a variation of the original Boyer-Moore string-matching algorithm, allows for approximate matching with a certain number of mismatches. This algorithm is useful in biological applications where variations and mutations in DNA sequences are common, enabling efficient searches for similar sequences even with errors or variations. These algorithms address the challenges of computational efficiency, accuracy, and handling variations in sequence analysis.

The following sections are organized as follows: Section I will review related work, focusing on previous studies that have contributed to the development and improvement of sequence analysis techniques in bioinformatics. Section II will detail the specific methodologies explored in this study, including the Smith Waterman algorithm, BLAST, and the Approximate Boyer-Moore algorithm, and their applications in fuzzy searching. Section IV will present experimental results, comparing the efficiency and accuracy of these algorithms in various scenarios of sequence analysis. Section V will discuss the advantages and limitations of each algorithm, along with potential future directions for research in the field of bioinformatics and fuzzy searching.

CCS CONCEPTS

•  Applied computing → Bioinformatics

• Theory of computation → Pattern matching

• Theory of computation → Approximation algorithms

KEYWORDS

Bioinformatics, Sequence alignment, Smith-Waterman algorithm, BLAST (Basic Local Alignment Search Tool), Approximate Boyer-Moore algorithm, DNA sequence analysis, Pattern matching, computational biology, genetic sequence comparison, algorithm efficiency, fuzzy searching algorithms.

1 Related Works

This section explores the research contributions towards the three algorithms explored in this study—BLAST, Smith-Waterman, and Boyer-Moore. Each of these studies provides valuable insights into the capabilities and challenges of these tools, highlighting their unique applications in genomic research.

1.1 BLAST

The BLAST algorithm is well known for its heuristic method of sequence alignment, which enables fast query searches across large genomic databases while keeping run-times reasonable, even with substantial data volumes. Camacho et al. conducted a detailed review of how BLAST's performance can falter as the size of the query word increases, primarily due to the repeated need to look up each letter of the word. To combat this issue, they implemented a more efficient data structure like a hash table, which significantly reduced lookup times and used less space.

Furthermore, Camacho et al. investigated different versions of the BLAST algorithm to better suit various research needs. They created megaBLAST for high-throughput analysis of sequences from the same species, using larger word sizes and requiring exact matches, which makes it exceptionally efficient for comparing highly similar sequences. On the other hand, BLASTN uses smaller word sizes and is designed for finding exact matches among more distantly related sequences, although it operates more slowly.

These variations are tailored to specific uses: megaBLAST is ideal for studying very similar sequences, while BLASTN is better for more exploratory searches where matches are not guaranteed. The work of Camacho et al. has greatly improved how BLAST software is implemented, notably by introducing customizable "tasks" in the BLAST command-line applications. These tasks let users adjust search parameters and optimization settings to meet the specific needs of their research, such as changing word sizes and match criteria, thereby making BLAST tools more flexible and user-friendly.

Their efforts also included rewriting the command lines in C++ for the NCBI software, allowing for more individualized adjustments of BLAST functions. This was a key advancement over older software versions, which didn’t support such modifications. By enabling these customizations, Camacho et al. have greatly enhanced the functionality and adaptability of BLAST applications, accommodating a broader spectrum of genomic research requirements. [1]

1.2 Smith-Waterman

Zhao and Zhang (2015) reported on the development of an accelerated version of the Smith-Waterman algorithm, designed for effective large-scale sequence matching, which is critical for phylogenetic analysis. Their study incorporated strategies such as inter-pair pruning and adaptive band optimization, which significantly reduce the computational load required for exhaustive pairwise sequence alignment. Particularly effective on GPU platforms, these methods facilitate rapid parallel processing and minimize unnecessary computations within the dynamic programming matrix, enhancing performance and resource management during the alignment of sequences with similar lengths and compositions.

Zhao and Zhang also noted that runtimes could dramatically increase with larger datasets, indicating the necessity for further optimizations to make bioinformatics applications practical. Their research emphasizes the need for performance enhancements beyond what is typically expected without such improvements. By utilizing GPU architecture, they achieved a speed-up factor of up to 160 times, making the algorithm suitable for extensive bioinformatics analyses. This significant performance boost results from strategic reductions in computational demand through inter-pair pruning and band optimization techniques, which focus processing power on the most promising segments of the alignment matrix without sacrificing accuracy.

These technological advancements not only accelerate processing times but also broaden the potential applications of the Smith-Waterman algorithm in extensive genomic studies. This breakthrough is vital for the bioinformatics community as it offers a faster and more efficient method for sequence alignment, essential for understanding genomic structure and function. By drastically reducing computational times, Zhao and Zhang's study opens new avenues for research and enables more comprehensive and in-depth genomic analyses that were previously too costly or time-consuming. [2]

1.3 Boyer-Moore

The Boyer-Moore algorithm is best known for its efficiency in pattern matching, particularly useful in DNA genomic sequence matching due to its ability to find exact matches of a pattern character by character, especially in larger datasets. Building on this, Nadia Ben Nsira, Thierry Lecroq, and Mourad Elloumi tackled the specific challenge of comparing highly similar, latter genome sequences that are more than 99% identical.

In their research, Nsira, Lecroq, and Elloumi introduced the "ExtendedFastSearch" algorithm, which enhances the traditional Boyer-Moore approach by incorporating techniques from the Fast-Search algorithm. This new algorithm is specifically optimized for high-performance on these latter genome sequences. The effectiveness of ExtendedFastSearch was assessed by comparing its runtime against several other algorithms designed to address similar challenges, including FJS, ExtendedNaive, and ExtendedMP.

The comparative analysis was conducted using two types of datasets: pseudo-random data and real genomic data. On pseudo-random data, ExtendedFastSearch demonstrated superior performance, particularly as the size of the pattern increased, as noted on page 283, section 6.1 of their publication. This trend continued with real data, where ExtendedFastSearch again outperformed the competing algorithms.

The findings, detailed in their conclusion assert that ExtendedFastSearch specifically excels in scenarios involving latter sequence datasets that exhibit more than 99% similarity. This specialization makes it a significant improvement on the existing Boyer-Moore algorithm for these specific contexts, according to Nsira, Lecroq, and Elloumi’s study. [3]

2 Experimentation

In this section of the study, three distinct experiments were conducted to evaluate the performance of different sequence matching algorithms under various conditions, emphasizing both exact and fuzzy logic matching approaches. The algorithms tested include BLAST, Smith-Waterman, and Boyer-Moore, each tailored to specific aspects of DNA sequence analysis and optimized to address unique challenges within the field of bioinformatics.

In section 2.1 the BLAST algorithm was implemented without accommodating gaps, focusing on optimal comparison of DNA subsequences. By setting a standard word size and manipulating nucleotide matching scores, the experiment sought to identify high-scoring subsequences from a genomic database, observing the algorithm's performance across linear searches and noting the effect of sequence size on computational time.

In section 2.2 this experiment tested the Smith-Waterman algorithm across three dataset sizes to assess its scalability and computational efficiency. The performance analysis highlighted the algorithm's polynomial runtime growth, particularly evident in larger datasets, and discussed the implications for practical bioinformatics applications where optimization becomes crucial.

Section 2.3 focuses on the Boyer-Moore algorithm and its variations, specifically the ExtendedFastSearch, which enhances fuzzy matching capabilities. By adjusting the max\_mismatches parameter, the experiment compared the exact and approximate matching performance, revealing how fuzzy logic can impact search efficiency and runtime, particularly in larger datasets.

2.1 BLAST

In this study, the BLAST algorithm was programmed in Python and implemented for DNA sequence searching, using a standard word size of 12 as is typical for DNA sequencing applications in BLAST. This implementation did not accommodate gaps in either the query or the database; instead, it was designed for optimal comparison of DNA subsequences without gaps. The coding implementation used in this experiment can be seen in Figure 1 and 2.

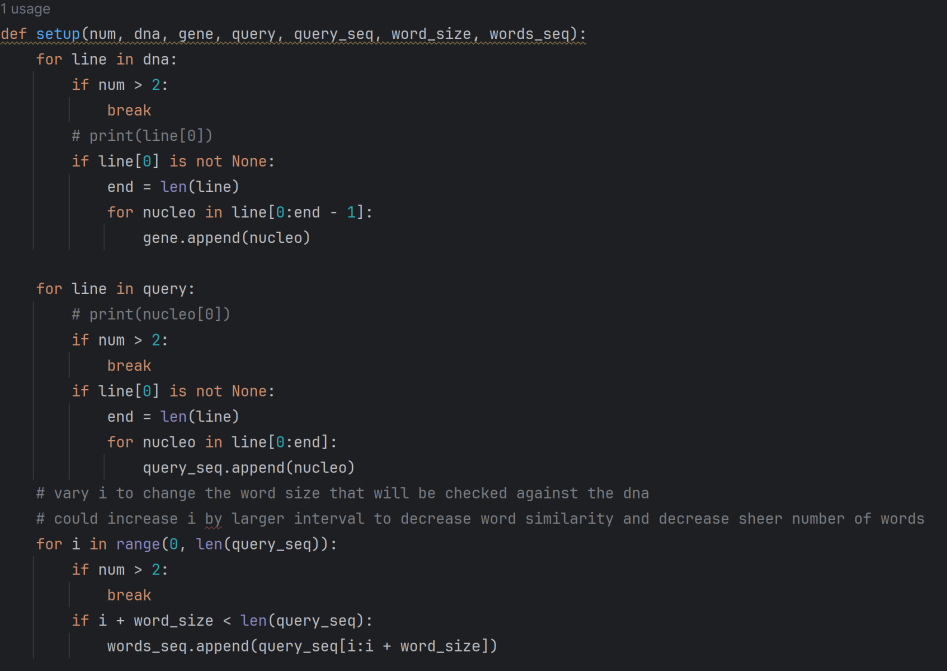


Figure 1: Python Smith-Waterman Algorithm

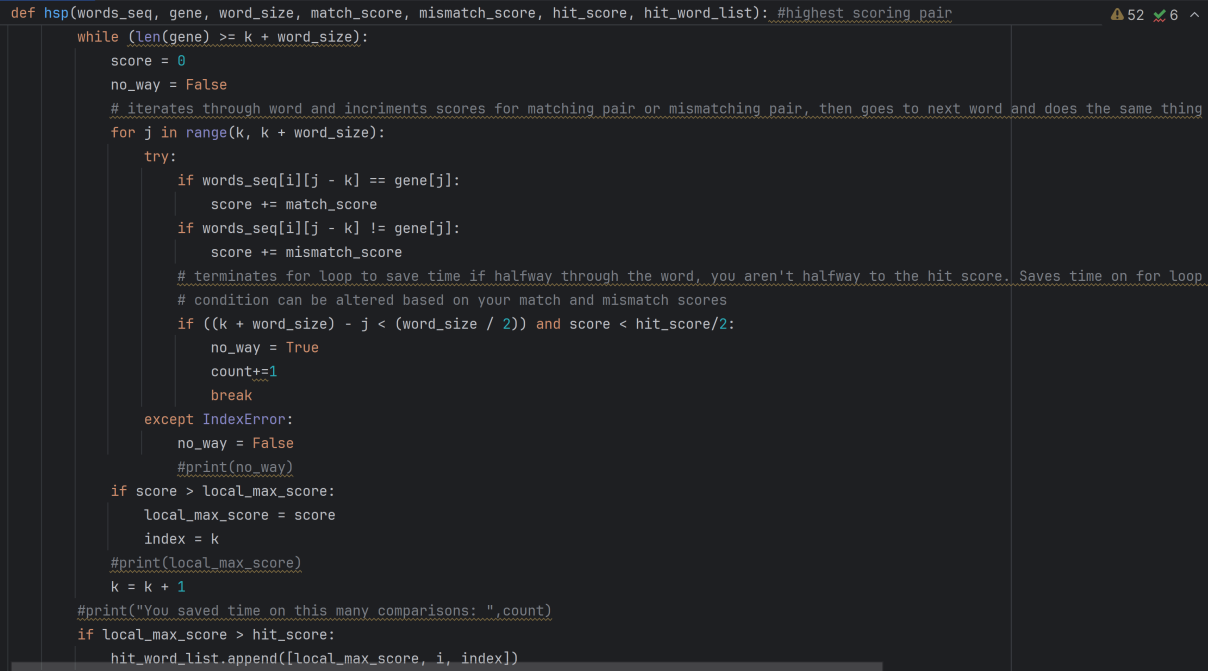


Figure 2: Python Smith-Waterman Algorithm

Initially, nucleotides from the database and the query strand (approximately 70 letters) were compiled into separate lists. From these, 12-letter words were generated from the query list. Matching and mismatching scores were then determined, set at five and negative four respectively, while the threshold (T) and minimum score (S) were calculated as a fraction of the optimal score. The goal was to identify database subsequences that scored at least the threshold score T. This involved comparing each nucleotide in the query words against nucleotides in the database, adding scores based on matches or mismatches. If the cumulative score exceeded the set minimum for T, the word was added to a new high-scoring list.

The process was then extended to determine if the words from this high-scoring list met the minimum scoring pair (MSP) criteria. Extensions were limited to one direction, with the extension limit set to a third of the word size. As each subsequent nucleotide in the database was compared to the next in the query, scores were adjusted for matches or mismatches accordingly. If a new word's score met or exceeded the MSP, the location and the appended nucleotide were recorded. This procedure was repeated for each word in the list, and any final word that met the MSP score was added to a new list, noting the location of the matching strand in the database.

Further exploration of the algorithm's performance with different word sizes was graphically represented in Figure 3. As shown, increasing the word size from zero to fifty resulted in a roughly linear increase in runtime, suggesting that even minor changes in word size can significantly affect the time efficiency of the BLAST searches. This behavior is likely due to the query size still being small compared to the database being searched, demonstrating that as word size increases linearly, so does the time to execute BLAST, especially when the only variable changed is the word size. This pattern underscores the need for careful selection of word size in optimizing BLAST for specific genomic research applications.

A graph with a line going up

Description automatically generated

**Figure 3: BLAST Word Size Vs Run Time**

Several technical adjustments were necessary to optimize the functionality of the code. Rather than directly manipulating strings of DNA, each nucleotide was added as an individual element to a list, simplifying lookups to constant time when the index is known. Unlike some related works that utilize a hash table for efficient data retrieval, this approach used list indexing for immediate access.

This program’s design reflects the descriptions of BLASTN discussed in the related works by Camacho C. et al. [1] Like BLASTN, the implementation used a word size of approximately 12, which typically results in longer execution times compared to other BLAST variants due to the smaller word size and heavier penalties for mismatches, ensuring that only sequences with high genetic similarity are considered. If a megaBLAST approach were employed, execution times could potentially be reduced significantly as it would involve fewer total comparisons and allow for the use of larger query sequences due to its less stringent match requirements.

2.2 Smith-Waterman

The Smith-Waterman algorithm was implemented to assess its performance across three varying dataset sizes, typical of genomic data applications, ranging from small sequences (150-200 characters) to very large sequences (260,000-280,000 characters).

The Python implementation can be seen in Figures 4 and continued in Figure 5.

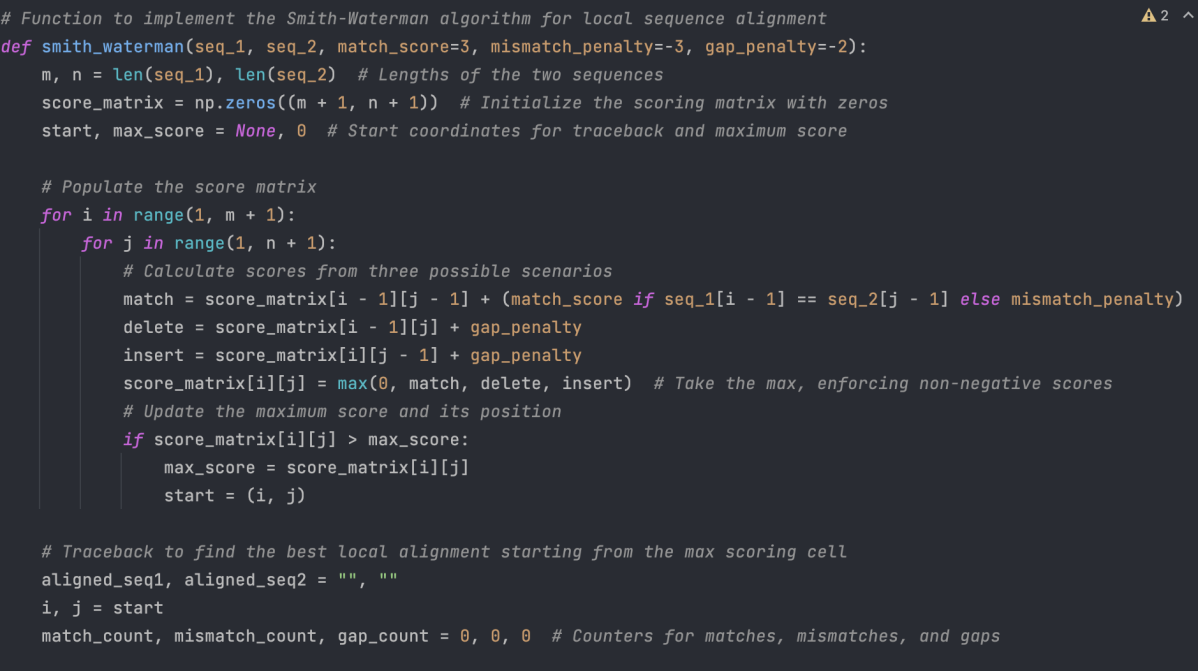


Figure 4: Python Smith-Waterman Algorithm

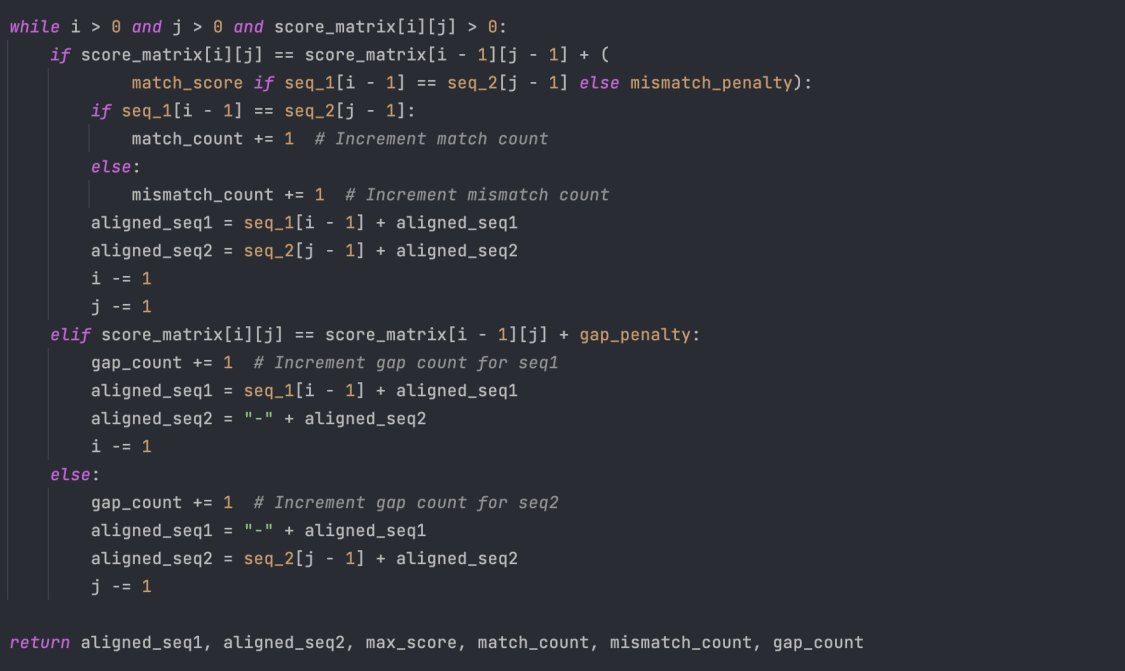


Figure 5: Python Smith-Waterman Algorithm Cont.

The respective processing times recorded for these datasets were 0.01 seconds, 10.29 seconds, and 1259.5 seconds, illustrating a significant increase in runtime with larger data sizes. This trend can be seen in Figure 6, where data gathered during experimentation shows a steep increase in runtime as the dataset size escalates.

A graph showing a line

Description automatically generated

Figure 6: Smith-Waterman Algorithm Data Size Vs. Runtime

This runtime escalation aligns with the Smith-Waterman algorithm's computational complexity, which is theoretically described by the Big O notation O(m × n), where 𝑚*m* and 𝑛*n* represent the lengths of the two aligned sequences. As the dataset size increased, the runtime growth followed a polynomial trend, specifically quadratic, which correlates with the increasing number of comparisons required by the algorithm. This pattern is particularly evident when runtime data is plotted against dataset sizes, showing a sharp increase in runtime as sequence lengths expand.

Comparative analysis with existing studies, such as those by Rognes and reviewed by Zhao and Zhang [2], indicated similar trends where runtime dramatically escalates with larger datasets. Rognes’ findings underscore the necessity for further optimizations to adapt the Smith-Waterman algorithm for practical bioinformatics applications, supporting the baseline performance expected without such enhancements. [2]

Through this experiment, it became evident that there are substantial limitations with the Smith-Waterman algorithm when analyzing large datasets. The runtime for the largest dataset tested highlighted the computational impracticality of employing this algorithm for routine bioinformatics tasks without significant optimization. This challenge aligns with the broader literature, which suggests either enhancing the algorithms or shifting to approximate methods for managing large datasets.

Looking ahead, the findings from this experiment suggest several potential avenues for future research. There is a clear benefit to exploring parallel computing techniques and specialized hardware, such as GPUs, to make the algorithm more practical for large-scale genomic analyses. Additionally, comparative studies with alternative algorithms that are computationally less complex could provide crucial insights. These studies might pave the way for more efficient designs for sequence alignment tools, which could significantly benefit the bioinformatics community.

2.3 Boyer-Moore

The Boyer-Moore algorithm is widely applied in DNA sequence matching, including variations such as the ExtendedFastSearch algorithm by Naira, Lecroq, and Elloumi, which enhances its capability in fuzzy matching as discussed in the related works. This paper focuses on testing both the exact matching and the approximate, fuzzy logic matching capabilities of the Boyer-Moore algorithm by manipulating the max\_mismatches variable. Setting max\_mismatches to 0 transforms it into a generic exact sequence matching algorithm, while setting it to a fifth of the pattern length allows for mismatches, making it a fuzzy logic algorithm. The hypothesis posits that the exact Boyer-Moore algorithm will run in 𝑂(n) time, and the approximate version will run in 𝑂(𝑛×𝑚) due to the inclusion of the max\_mismatches.

Figure 7 provides the Python implementation of the algorithm used in this experiment.

A screenshot of a computer program

Description automatically generated

Figure 7: Boyer-Moore Python Algorithm

To verify this hypothesis, the experiment first analyzes the runtime differences across various dataset sizes. The results confirm the hypothesis: both versions of the Boyer-Moore algorithm exhibit linear runtimes, though the approximate version has a longer runtime due to the added complexity of the max\_mismatches. This is visualized in Figure 8, which shows a clear distinction in runtime escalation between the two algorithms as dataset size increases.

A graph of a graph of a price

Description automatically generated with medium confidence

Figure 8: Boyer-Moore and Boyer-Moore-Approximate Runtime Vs. Dataset Size

Further analysis focuses on the impact of pattern length on runtime. For this part of the experiment, the dataset size was limited to the first 100 million characters to manage runtime. As the pattern length increased, finding exact matches with the standard Boyer-Moore algorithm proved more efficient, consistent with its design for exact matching. Despite both algorithms running in linear time, the Boyer-Moore Approximate exhibited longer runtimes with longer pattern lengths, as detailed in Figure 9. Figure 9 illustrates how runtime extends with longer patterns, particularly under the fuzzy logic approach.

A graph with red line and blue line

Description automatically generated

Figure 9: Boyer-Moore and Boyer-Moore-Approximate Vs. Pattern Length

Additionally, Figure 10 provides insights into the performance implications of different levels of mismatches within the fuzzy logic context, highlighting the computational toll considerations for mismatches impose.

A graph with a line

Description automatically generated

Figure 10: Boyer-Moore-Approximate Mismatches Comparison

The experiment demonstrates distinct applications for each algorithm variant. The standard Boyer-Moore algorithm is particularly effective for searching exact, smaller pattern instances within larger datasets—a common requirement in identifying specific mutations in DNA sequences where the target sequence is precisely known. Conversely, the Boyer-Moore Approximate maintains a linear runtime but tends to perform with less efficiency in larger datasets when fuzzy logic is applied, as the allowance for mismatches leads to more computational iterations. This delineation of strengths and weaknesses underscores the tailored application of each algorithm variant in genomic research.

2.4 Fuzzy Search Logic

Each experiment connects to fuzzy search logic by exploring how algorithms handle sequence mismatches and alignment ambiguities—core aspects of fuzzy matching. The BLAST and Boyer-Moore experiments, for instance, directly engage with fuzzy search principles by adjusting mismatch penalties and exploring the thresholds at which sequences are considered similar enough to be deemed a match. This is particularly important in genomic studies where exact matches are rare, and the ability to efficiently identify near-matches can significantly impact the understanding of genetic variations and mutations. The Smith-Waterman experiment, while focused more on scalability and computational demands, also indirectly supports the need for fuzzy logic by highlighting the limitations of traditional exact matching approaches in handling large-scale genomic data.

Together, these experiments provide a comprehensive view of how different sequence alignment algorithms perform under varying conditions and establish a foundation for further research into more efficient and effective bioinformatics tools, particularly those that leverage fuzzy logic to enhance their practical utility in real-world applications.

3 Conclusion

This study delved into the performance and application of three key algorithms in bioinformatics: BLAST, Smith-Waterman, and Boyer-Moore. Each algorithm serves unique functions in DNA sequence analysis, and our extensive testing shed light on how they handle both exact and fuzzy logic matching scenarios. Our results reveal how each algorithm's effectiveness interacts with dataset characteristics and research demands, pointing to several areas ripe for further study.

Our experiments showed that while BLAST and Boyer-Moore perform exceptionally well for certain types of sequence matching, they require thoughtful adjustment of settings such as word size and mismatch thresholds to effectively balance speed with accuracy, particularly with larger datasets. The Smith-Waterman algorithm, with its high precision in local alignment, faces scalability issues that could be addressed with techniques like parallel processing and the use of GPUs.

Fuzzy logic proved essential for managing the variability and imperfections inherent in genomic data. Allowing for some uncertainty in matches, fuzzy searching offers a more flexible and thorough way of analyzing sequences, leading to discoveries that might be overlooked with strict exact matching. Yet, this flexibility also adds a layer of complexity that can affect computational efficiency, as seen in our experiments adjusting mismatch settings in the Boyer-Moore algorithm.

Looking ahead, it's vital to further refine these algorithms, exploring advanced fuzzy logic models and the potential of parallel computing architectures. Developing hybrid algorithms that can adjust their parameters based on specific queries and data characteristics could also lead to more versatile sequence alignment tools. Such advancements could significantly enhance the speed and precision of bioinformatics applications, giving researchers better tools to unravel complex biological data and push forward the fields of genetics, personalized medicine, and disease research.

In summary, each algorithm has proven effective, but integrating and optimizing them for bioinformatics continues to be a challenging endeavor that bridges computational techniques with biological research.

REFERENCES

[1] C. Camacho, G. Coulouris, V. Avagyan, et al. 2009. BLAST+: architecture and applications. BMC Bioinformatics, 10, 421. <https://doi.org/10.1186/1471-2105-10-421>

[2] Y. M. Zhao, & K. Zhang. 2015. SW#-GPU: Smith-Waterman algorithm on GPUs for massively parallel DNA database search and its applications in biomedicine and evolution. BMC Bioinformatics, 16(1).

[3] N. Ben Nsira, T. Lecroq, and M. Elloumi. 2015. A fast Boyer-Moore type pattern matching algorithm for highly similar sequences. International Journal of Data Mining and Bioinformatics (IJDMB), Vol. 13, No. 3, 266–288. <https://doi.org/10.1504/IJDMB.2015.072101>.

[4] J. Allmer. 2016. Exact pattern matching: Adapting the Boyer-Moore algorithm for DNA searches. PeerJ PrePrints. <https://doi.org/10.7287/peerj.preprints.1758v1>.

[5] W. R. Pearson. 1991. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. Genomics, 11(3), 635–650. <https://www.proquest.com/scholarly-journals/searching-protein-sequence-libraries-comparison/docview/16057603/se-2>.

[6] P. Kalsi, L. Salmela, and J. Tarhio. 2007. Tuning Approximate Boyer-Moore for Gene Sequences. In N. Ziviani and R. Baeza-Yates (Eds.), String Processing and Information Retrieval (SPIRE 2007), Lecture Notes in Computer Science, vol 4726. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-540-75530-2_16>.

[7] K. C. Samal, J. P. Sahoo, L. Behera, and T. Dash. 2021. Understanding the BLAST (basic local alignment search tool) Program and a step-by-step guide for its use in life science research. Bhāratīya krṣi anusandhan patrika. <https://doi.org/10.18805/BKAP283>.

[8] I. Gábor, G. Iván, D. Bánky, and V. Grolmusz. 2016. Fast and exact sequence alignment with the Smith–Waterman algorithm: The SwissAlign webserver. Gene Reports, 4, 26–28. <https://doi.org/10.1016/j.genrep.2016.02.004>.

[9] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. 1990. Basic local alignment search tool. Journal of Molecular Biology, 215(3), 403–410. <https://www.proquest.com/scholarly-journals/basic-local-alignment-search-tool/docview/15819331/se-2>.

DATASETS

[1] I. H. Barrette-Ng, K. K.-S. Ng, B. L. Mark, D. van Aken, M. M. Cherney, C. Garen, Y. Kolodenko, A. E. Gorbalenya, E. J. Snijder, and M. N. G. James. 2002. NSP4 proteinase from Equine Arteritis Virus. Protein Data Bank. <https://doi.org/10.2210/pdb1MBM/pdb>

[2] M. Paoli, R. Liddington, J. Tame, A. Wilkinson, G. Dodson. 2002. Oxy T State Haemoglobin - Oxygen bound at all four haems. Protein Data Bank. <https://doi.org/10.2210/pdb1GZX/pdb>.

[3] National Center for Biotechnology Information. 2023. Homo sapiens fibroblast growth factor 21 (FGF21), RefSeqGene on chromosome 19. NCBI Reference Sequence: NG\_033945.1. Available at <https://www.ncbi.nlm.nih.gov/nuccore/NG_033945.1>.

[4] National Center for Biotechnology Information. 2023. Escherichia coli 16S ribosomal RNA, complete sequence. GenBank: J01859.1. Available at <https://www.ncbi.nlm.nih.gov/nuccore/J01859.1>

[5] National Center for Biotechnology Information. 2023. Danio rerio strain Tuebingen chromosome 3, GRCz11 Primary Assembly. NCBI Reference Sequence: NC\_007114.7. Available at <https://www.ncbi.nlm.nih.gov/nuccore/NC_007114.7>.

[6] National Center for Biotechnology Information. 2023. Danio rerio strain Tuebingen chromosome 3, GRCz11 Primary Assembly. NCBI Reference Sequence: NC\_007114.7. Available at <https://www.ncbi.nlm.nih.gov/nuccore/NC_007114.7>.

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00