A Comparative Study of Needleman-Wunsch and Smith-Waterman Algorithms for DNA Sequence Alignment in Viral Variants

Aurelia Jennifer Gunawan - 13524089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: aurellia.jennifer@gmail.com , 13524089@std.stei.itb.ac.id

***Abstract*—DNA sequence alignment is a key task in bioinformatics. It helps researchers find similarities and differences between biological sequences. This is important for detecting mutations, studying how organisms are related, tracking diseases, and identifying viral variants. Because viral genomes can change through substitutions, insertions, and deletions, reliable computational methods are needed for accurate comparisons. Sequence alignment is usually set up as an optimization problem and often solved with dynamic programming. In this study, we compare the Needleman-Wunsch (NW) and Smith-Waterman (SW) algorithms for aligning DNA sequences from viral variants. The NW algorithm aligns entire sequences, while the SW algorithm finds the best matching parts of sequences. We implemented and tested both algorithms on DNA sequences with different levels of mutation. We compared how well they align sequences, looking at alignment scores, sequence identity, and how long each method takes to run. Our goal is to see how effective global and local alignment methods are under different mutation conditions and to show how dynamic programming is used in sequence alignment in bioinformatics.**

Keywords—Needleman-Wunsch, Smith-Waterman, dynamic programming, DNA sequence alignment, viral variants

I. Introduction

Recent findings in genomic sequencing have allowed scientists to analyze biological sequences with greater accuracy on a larger scale. DNA sequence alignment plays a central role in bioinformatics because it helps identify similarities and differences between nucleotide sequences [1]. This is useful for studying evolution, tracking, diseases, finding mutations, and identifying viral variants. Aligning DNA sequences allows researchers to identify conserved regions, detect mutations, and better understand how different organisms are related.

The appearance of new viral variants highlights the importance of sequence alignment. Viruses change over time through mutations such as substitutions, insertions, and deletions, leading to new variants with unique genetic profiles. By comparing viral DNA or RNA sequences, scientists can track how viruses evolve and identify genetic changes linked to specific variants [2]. As sequences become longer and mutations more complex, there is a great need for faster, more efficient algorithms to manage them. From an algorithmic point of view, sequence alignment is seen as an optimization problem. The goal is to align two biological sequences to achieve the highest score, considering matches, mismatches, and gaps. Since the number of possible alignments increases quickly as sequences get longer, it is impractical to check every option. Efficient algorithms are needed to find the best alignments within a reasonable time. Dynamic programming is often used as it applies the principle of optimality and saves time by storing intermediate results. The Needleman-Wunsch and Smith-Waterman algorithms are well-known dynamic programming methods for sequence alignment [3]. Needleman-Wunsch is used for global alignment, while Smith-Waterman is used for local alignment [4].

This study compares the NW and SW algorithms for aligning DNA sequences from various viral variants. The analysis measures how well each algorithm performs by looking at alignment scores, sequence identity, and execution time. By testing both algorithms with different types of mutations, the study intends to highlight the strengths and weaknesses of global and local alignment methods for analyzing viral sequences.

II. Theoretical Foundations

A. DNA Sequences and Viral Variances

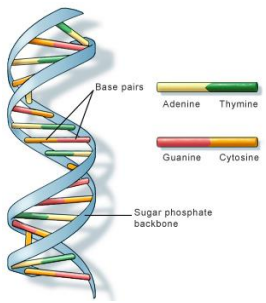


Fig. 1: Structure of DNA
Source: https://praxilabs.com/en/blog/2021/02/08/dna-sequencing-definition-importance-methods-facts-and-more/

Deoxyribonucleic acid (DNA) is the fundamental molecule of heredity in all known living organisms. A DNA sequence is formally represented as a string over the alphabet $\Sigma = \{A, T, G, C\}$, corresponding to the four nucleotide bases: adenine,

thymine, guanine, and cytosine. The ordering of these bases encodes the genetic instructions that determine the structure and function of proteins, and by extension, the biological characteristics of an organism [5].

Viruses play a critical role in genomic analysis because of their rapid mutation rates. Unlike cellular organisms, many viruses, especially RNA viruses such as SARS-CoV-2 and influenza, exhibit elevated mutation rates due to the low fidelity of their replication mechanisms. These mutations accumulate over time, generating distinct viral variants that can differ in transmissibility, immune evasion, or clinical severity [6]. Identifying and characterizing these variants requires computational methods capable of detecting subtle genomic differences.

A central challenge in viral genomics is determining the degree of similarity or divergence between a newly sequenced viral genome and a reference sequence or other known variants. Even minor nucleotide substitutions, insertions, or deletions can have significant biological consequences. Computational sequence alignment provides a systematic framework for identifying these differences, supporting the reconstruction of evolutionary relationships, the detection of clinically significant mutations, and the development of diagnostic tools or vaccines targeting conserved genomic regions [7].

B. *Sequence Alignment*

Sequence alignment is the process of arranging two or more sequences to identify regions of similarity that may reflect functional, structural, or evolutionary relationships. Given two sequences S_1 of length m and S_2 of length n, an alignment is produced by inserting gap characters into one or both sequences such that the resulting strings are of equal length and columns in the alignment correspond to either matched characters, mismatched characters, or a gap opposite a character [1].

Fig. 2: Global and Local Alignment
Source: https://www.majordifferences.com/2016/05/difference-between-global-and-local.html

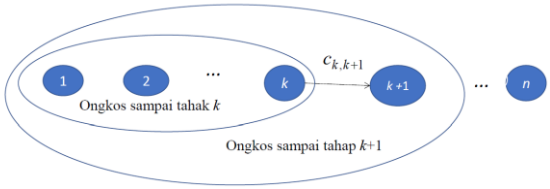
Two principal categories of alignment exist. Global alignment seeks to align the entire length of both sequences end to end and is most appropriate when the sequences under comparison are of similar length and presumed to be homologous throughout [3]. Local alignment, by contrast, identifies the highest-scoring contiguous sub-regions between two sequences, making it suitable for comparing sequences that share only partial or domain-level similarity [8].

Alignment quality is evaluated using a scoring scheme that assigns positive values to matches, negative values (penalties) to mismatches, and gap penalties to insertions or deletions. A common method applies a linear gap penalty, δ, to each introduced gap. With a match score s(a, b) between characters a and b, the optimal alignment maximizes the total score, which corresponds to minimizing the edit distance between sequences [3]. Because the number of possible alignments grows exponentially with sequence length, exhaustive enumeration is computationally infeasible. Dynamic programming overcomes this limitation by decomposing the problem into overlapping subproblems and storing intermediate results.

C. *Dynamic Programming*

Dynamic programming (DP) is an algorithmic paradigm for solving optimization problems by decomposing them into a sequence of interdependent decisions. DP structures a solution as a collection of stages, enabling the overall solution to be interpreted as a series of interrelated decisions. The term "dynamic" refers to the use of a growing table to record intermediate computational results, rather than implying a temporal property [10]. DP differs from the greedy algorithm paradigm by evaluating multiple decision sequences rather than committing to a single locally optimal choice at each step. This property makes DP especially suitable for problems in which locally optimal decisions do not yield a globally optimal solution, a scenario frequently encountered in sequence alignment [9].

1) *Principle of Optimality*



2

Fig. 3: Principle of Optimality Diagram
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/25-Program-Dinamis-(2026)-Bagian1.pdf

The theoretical foundation of dynamic programming rests on the Principle of Optimality, articulated by Richard Bellman: if a total solution is optimal, then every sub-solution constituting it is also optimal [10]. Formally, if the cost incurred from stages 1 through k + 1 is optimal, then the cost from stages 1 through k must also be optimal with respect to the choices made up to that point. In the context of sequence alignment, if the optimal alignment of sequences $S_1[1…m]$ and $S_2[1…n]$ is known, then the alignment of any prefix $S_1[1…i]$ with $S_2[1…j]$ embedded within it must itself be optimal for those prefixes. This recursive structure permits the construction of a globally optimal alignment by incrementally computing and storing optimal prefix alignments.

2) *Memoization and Dynamic Programming Table*

The principal computational mechanism enabling dynamic programming to avoid redundant recomputation is memoization, which stores the value of each subproblem in a table immediately after computation. This strategy ensures that subsequent queries for the same subproblem are resolved in constant time. Because each subproblem is computed only once, the overall time complexity is reduced from exponential to polynomial. In sequence alignment, the dynamic programming

table is a two-dimensional matrix H with dimensions (m + 1) × (n + 1), where H[i][j] denotes the optimal alignment score for the prefix pair (S_1[1…i], S_2[1…j]). The table is filled in either row-major or column-major order, ensuring that all dependencies, specifically H[i−1][j−1], H[i−1][j], and H[i][j−1], are resolved before computing H[i][j].

3) Recursive Formulation

The optimal value function at each stage is defined recursively in terms of optimal values from prior stages. For a general multistage graph problem, the recurrence takes the form:

$$f_k(s) = opt_{x_k}\{c_{s,x_k} + f_{k+1}(x_k)\}$$

where s denotes the current state, x_k is the decision variable at stage k, c_{s,x_k} is the transition cost, and $f_{k+1}(x_k)$ is the optimal accumulated value from the next stage onward [10]. The basis condition is given by $f_n(s) = c_{s,destination}$ for terminal-stage states. In sequence alignment, the state corresponds to the pair (i, j), representing positions in the two sequences; the decision at each state is whether to match/mismatch the current characters or introduce a gap; and the transition cost reflects the scoring scheme. The recurrence for the general alignment problem is thus:

$$H[i][j] = opt\{H[i-1][j-1] + s(S_1[i], S_2[j]), H[i-1][j] - \delta, H[i][j-1] - \delta\}$$

where s(a,b) is the substitution score and δ is the gap penalty. The specific recurrence, and its initial conditions and optimization direction, differ between global and local alignment.

4) Solution Reconstruction (Traceback)

Computing only the optimal value is typically insufficient; it is also necessary to recover the sequence of decisions that constitute the optimal alignment. Solution reconstruction is performed by tracing back through the dynamic programming table from the terminal state to the initial state. At each step, the predecessor cell is chosen so that its value, together with the transition cost, matches the current cell's value. In sequence alignment, a pointer matrix P[i][j] ∈ {↖, ↑, ←} is maintained alongside H during the filling process to record the origin of each H[i][j] value. Traceback begins at a designated cell, (m, n) for global alignment or the cell with the maximum score for local alignment, and follows the pointers until the table boundary is reached. Diagonal pointers (↖) indicate a match or mismatch, upward pointers (↑) indicate a gap in S_2, and leftward pointers (←) indicate a gap in S_1. The resulting path, when read in reverse, yields the optimal alignment.

D. Needleman-Wunch Algorithm

The Needleman-Wunsch (NW) algorithm, introduced by Saul B. Needleman and Christian D. Wunsch in 1970, was among the first dynamic programming algorithms applied to biological sequence alignment. Its objective is to compute the optimal global alignment between two sequences by maximizing a cumulative similarity score over the entire length of both sequences [3]. The algorithm initializes a scoring matrix H of size (m + 1) × (n + 1). The first row and column are filled according to the linear gap penalty:

$$H[i][0] = -i \cdot \delta, \text{ for i = 0, 1, …, m}$$

$$H[0][j] = -j \cdot \delta, \text{ for j = 0, 1, …, m}$$

This initialization reflects the cost of aligning i characters of S_1 against j gaps, or vice versa, before any character comparison has occurred. For i ∈ {1, …, m} and j ∈ {1, …, n}, the matrix is filled using:

$$H[i][j] = \max\{H[i-1][j-1] + s(S_1[i], S_2[j]), H[i-1][j] - \delta, H[i][j-1] - \delta\}$$

The three candidate values correspond, respectively, to: (1) aligning S_1[i] with S_2[j] (match or mismatch), (2) inserting a gap in S_2 (deletion in S_1), and (3) inserting a gap in S_1 (deletion in S_2). The substitution score s(a, b) is typically set to a positive value for a match and a negative value for a mismatch, as defined by the scoring matrix in use.

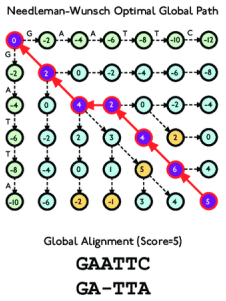


Fig. 4: Example of Needleman-Wunsch Global Alignment
Source: https://www.researchgate.net/figure/Dynamic-programming-path-graphs-for-Needleman-Wunsch-and-Smith-Waterman-alignments_fig2_228986143

After the matrix is fully computed, the optimal global alignment score is found at H[m][n]. Traceback proceeds from H[m][n] to H[0][0] by following the pointer matrix, yielding the aligned sequence pair. The time and space complexities of the Needleman-Wunsch algorithm are both O(mn), where m and n represent the lengths of the two input sequences. Memory-efficient variants, such as Hirschberg's algorithm, reduce the space requirement to O(min(m, n)), though at the cost of increased computational cost.

E. Smith-Waterman Algorithm

The Smith-Waterman (SW) algorithm, introduced by Temple F. Smith and Michael S. Waterman in 1981, addresses the problem of optimal local alignment. Rather than aligning two sequences in their entirety, SW identifies the pair of substrings, one from each input sequence, that achieves the highest similarity score under the given scoring scheme. This makes it especially powerful for detecting conserved functional domains or mutation hotspots within otherwise divergent sequences.

A key distinction from Needleman-Wunsch lies in the matrix initialization. In Smith-Waterman, the entire first row and first column are set to zero:

$$H[i][0] = 0, \text{ for i = 0, 1, …, m}$$

$$H[0][j] = 0, \text{ for j = 0, 1, …, m}$$

This ensures that every possible sub-alignment can begin at any position in either sequence without incurring a penalty for upstream unaligned regions. For i ∈ {1, …, m} and j ∈ {1, …, n}:

$$H[i][j] = \max\{0, H[i-1][j-1] + s(S_1[i], S_2[i]), H[i-1][j] - \delta, H[i][j-1] - \delta\}$$

The inclusion of 0 as a candidate value allows any cell to serve as the starting point for a new local alignment, effectively resetting the score when it would otherwise fall below zero. Consequently, H[i][j] represents the score of the best local alignment ending at position i in S_1 and position j in S_2.

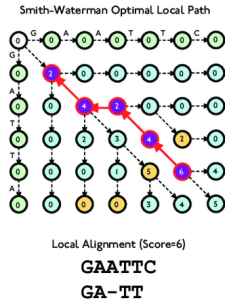


Fig. 5: Example of Smith-Waterman Local Alignment
Source: https://www.researchgate.net/figure/Dynamic-programming-path-graphs-for-Needleman-Wunsch-and-Smith-Waterman-alignments_fig2_228986143

Unlike Needleman-Wunsch, traceback in Smith-Waterman does not begin at a fixed corner. Instead, the algorithm first identifies the cell H[i*][j*] containing the global maximum score across the entire matrix. Traceback then follows pointers from H[i*][j*] until a cell with value 0 is reached, at which point the local alignment is complete [8]. This yields the highest-scoring local alignment between the two sequences. The Smith-Waterman algorithm exhibits time and space complexities of O(mn), where m and n represent the lengths of the two input sequences.

III. Methodology

This study employs and experimental comparative design to evaluate the performance of the Needleman-Wunsch (NW) and Smith-Waterman (SW) algorithms for DNA sequence alignment. The primary objective is to compare global and local alignment strategies across varying sequence similarities and input sizes. The comparison emphasizes alignment quality and algorithmic performance. Three experiments are conducted in this study. The first utilizes real-world viral genome sequences to assess practical alignment performance. The second examines the effect of sequence length on execution time. The third investigates how increasing mutation rates influence alignment quality and runtime.

A. *Experimental Variables*

The independent variables are the alignment algorithm (NW or SW), sequence length, mutation rate, and the specific sequence pair being aligned, while the dependent variables are alignment score, identity percentage, alignment length, gap count, and execution time. Controlled variables include the scoring scheme, implementation language, execution environment, and experimental procedures. Identical scoring parameters and evaluation metrics are applied to both algorithms to ensure comparability.

B. *Dataset*

1. Real-world viral sequences

Complete SARS-CoV-2 genome sequences are obtained from the NCBI virus database in FASTA format. Three representative variants are selected: the Wuhan reference strain, the Delta variant, and the Omicron variant. These variants represent distinct stages of viral evolution and provide sequence pairs with varying levels of genetic divergence.

2. Synthetic sequences

Synthetic DNA sequences are generated to provide controlled experimental inputs. Each sequence consists of nucleotides from the set {A, T, C, G}. Controlled mutations are introduced via random nucleotide substitutions at predefined mutation rates. These synthetic datasets are intended to create measurable levels of sequence divergence, rather than to model biological evolution.

C. *Evaluation Metrics*

Both algorithms are evaluated using a consistent scoring scheme across all experiments. Runtime measurements are obtained using a high-resolution timing function. Experiments involving runtime analysis are repeated multiple times to reduce measurement noise. All experiments are conducted using identical implementation settings and evaluation procedures to ensure that observed differences arise from the alignment strategies rather than from variations in experimental conditions.

D. *Alignment Algorithms*

1) Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm performs global sequence alignment using dynamic programming. A scoring matrix of size $(m + 1) \times (n + 1)$ is constructed, where (m) and (n) denote the lengths of the input sequences. The matrix is initialized with cumulative gap penalties and filled using recurrence relations that account for match, mismatch, and gap operations. Traceback is then performed from the final cell to reconstruct the optimal global alignment.

2) Smith-Waterman Algorithm

The Smith-Waterman algorithm performs local sequence alignment using dynamic programming. Its recurrence relation includes zero as a candidate score, allowing alignment to begin and end at arbitrary positions within the sequences. Traceback starts from the highest-scoring cell and continues until a cell with score zero is reached, producing the highest-scoring local alignment region.

E. *Experimental Procedures*

1) Variant Comparison Experiment

The first experiment aligns three pairs of SARS-CoV-2 variants: Wuhan-Delta, Wuhan-Omicron, and Delta-Omicron. Each pair is processed using both NW and SW algorithm. Alignment score, identity percentage, alignment length, gap count, and execution time are recorded.

2) Runtime Experiment

The second experiment evaluates computational performance using synthetic sequences of increasing length. Sequence lengths are systematically varied while all other parameters remain constant. The runtime of each algorithm is measured repeatedly and summarized using descriptive statistics. The resulting trends are analyzed in relation to the theoretical O(mn) time complexity of dynamic programming-based sequence alignment.

3) Mutation Experiment

The third experiment evaluates algorithm behavior as sequence divergence increases. Mutated sequences are generated from a reference sequence at predefined mutation rates. Each mutated sequence is aligned against the initial reference using both algorithms. Alignment quality and runtime metrics are collected and summarized to assess the effect of mutation rate on algorithm performance.

F. Performance Measures

Alignment quality is assessed with alignment score, identity percentage, alignment length, and gap count. The identity percentage is calculated as:

$$Identity = \frac{Number\ of\ matching\ positions}{Alignment\ length} \times 100\%$$

Computational performance is assessed based on execution time measured during the alignment process.

G. Data Analysis

Experimental results are exported for analysis and visualization. For experiments involving repeated trials, summary statistics such as mean and standard deviation are computed. Comparisons are conducted across algorithms, sequence lengths, mutation rates, and viral variants. The analysis focuses on three questions: (1) how global and local alignment strategies differ in alignment quality, (2) how sequence divergence affects alignment performance, and (3) whether empirical runtime trends are consistent with the theoretical complexity of dynamic programming-based sequence alignment algorithms.

IV. Result and Analysis

A. Viral Variant Alignment Analysis

The alignment outcomes for the three SARS-CoV-2 variant pairs, each comprising 1000 bp sequences and evaluated with s(match) = +2, s(mismatch) = −1, and δ = −2, are presented in the table below:

Table 1. Alignment Outcome by Variant Pair

Pair	Algorithm	Score	Identity	Gaps	Align. Length
Wuhan-Delta	NW	1844	0.9493	50	1025
	SW	1944	0.9979	0	975
Wuhan-Omicron	NW	1673	0.8966	108	1054
	SW	1889	0.9989	0	946
Delta-Omicron	NW	1823	0.9427	58	1029
	SW	1939	0.9990	0	971

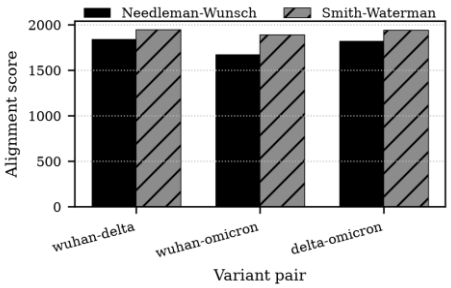


Fig. 6: Variant Score Comparison Chart
Source: The Author's experiment

Under the Needleman-Wunsch (NW) algorithm, the ordering of pairwise identity (Wuhan-Delta > Delta-Omicron > Wuhan-Omicron) corresponds to the gap count assigned to each pair: 50, 58, and 108 gap positions, respectively. Because global alignment requires traversing both sequences from end to end, every region containing an insertion or deletion (indel) necessitates a corresponding gap insertion in the opposing sequence. The resulting penalty (δ = −2 per gap) is unavoidable, regardless of the favorability of surrounding matches. The Wuhan-Omicron pair, which has more than twice the gap burden of the other two pairs, consequently records both the lowest NW score (1673) and the lowest NW identity (0.8966), consistent with the comparatively greater genomic divergence attributed to the Omicron lineage.

The Smith-Waterman (SW) algorithm produces a markedly different outcome. For all three variant pairs, local alignment yields a zero gap count and an identity above 0.997, regardless of the number of gaps required under NW. Figure 6 illustrates that the score gap between the two algorithms increases from 100 points (Wuhan-Delta) to 216 points (Wuhan-Omicron), which is the pair with the highest gap burden under NW. This pattern arises directly from the SW recurrence: since each cell can reset to zero, the traceback can exclude the diverging flanks of the global alignment and report only the longest internal segment that aligns without indels. Consequently, the algorithm measures not whole-genome similarity but the best-matching conserved core shared by the two variants. This explains why SW identity remains close to 1.0 even for the most divergent pair, while NW identity falls below 0.90.

These two alignment outcomes address distinct biological questions. NW identity reflects the extent to which the full genomic length is preserved between variants, including the cost of accommodating insertions and deletions accumulated over evolutionary history. This measure is relevant when assessing whole-genome divergence or phylogenetic distance. In contrast, SW identity isolates a high-confidence conserved region, which is more applicable when searching for a stable primer-binding site or a domain that remains intact despite mutations elsewhere in the genome. Neither score is inherently superior; rather, each answers a different question. The divergence between NW and SW identities increases with the amount of structural variation

(gaps) separating the two sequences, as demonstrated most clearly for the Wuhan-Omicron pair.

B. Runtime Performance Analysis

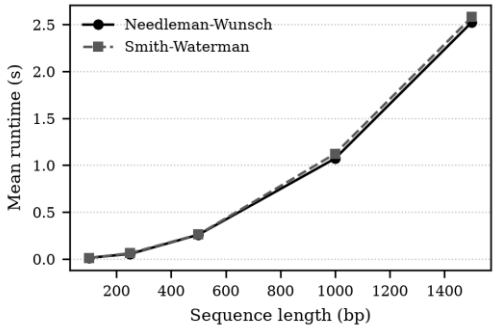


Fig. 7: Runtime vs. Sequence Length Chart
Source: The Author's experiment

Figure 7 and the corresponding summary statistics indicate that both algorithms exhibit similar growth patterns as sequence length increases from 100 base pairs to 1500 base pairs, with mean runtime rising from approximately 0.011-0.013 seconds to 2.5-2.6 seconds. Because the test pairs were of equal length ($m = n$), the $O(mn)$ complexity for both algorithms simplifies to $O(n^2)$, which can be directly verified using the measured runtime ratios. Doubling the sequence length from 500 base pairs to 1000 base pairs increases mean runtime by a factor of approximately 4.1 for NW and 4.3 for SW. Increasing the length from 1000 base pairs to 1500 base pairs (a 1.5× increase) raises runtime by a factor of approximately 2.4, compared to a quadratic prediction of 2.25. Both results closely match the n^2 scaling expected from a dynamic programming table of $(m + 1) \times (n + 1)$ cells, each requiring constant-time evaluation of the recurrence.

A deviation from this pattern is observed at the smallest sequence length tested. Increasing the length from 100 base pairs to 250 base pairs (a 2.5× increase) raises runtime by a factor of only 4.3, which is significantly below the 6.25× predicted by a purely quadratic model. This discrepancy is attributable to fixed overhead, such as matrix allocation, function call setup, and interpreter-level constants, which do not scale with n and therefore constitute a larger proportion of total runtime at small sequence lengths. As n increases, this constant term becomes negligible relative to the $O(n^2)$ term, and the measured growth rate approaches the theoretical quadratic relationship, as observed in the 500–1500 base pair. NW and SW exhibit nearly identical runtimes at every tested sequence length, with the largest relative difference (approximately 4.7%) observed at 1500 base pairs. This similarity is expected due to the algorithms' shared recurrence structure: both fill an identical $(m + 1) \times (n + 1)$ matrix with a three-way comparison per cell. The operations that differ between them, such as initialization of the boundary row or column and the starting point for traceback, are $O(m + n)$ and $O(1)$, respectively, and contribute negligibly to the total computational cost compared to the $O(mn)$ matrix-fill step. The runtime experiment thus confirms that the distinction between global and local alignment has no measurable effect on computational cost; it only determines which alignment is reported, not the computational expense.

C. Mutation Sensitivity Analysis

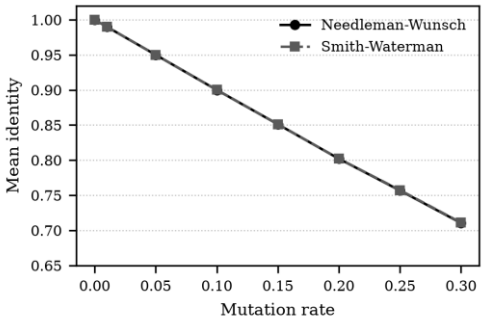


Fig. 8: Identity vs. Mutation Rate Chart
Source: The Author's experiment

The figure above shows that the mean identity declines from 1.0 at a mutation rate of 0.0 to approximately 0.71 at 0.30 for both algorithms. At low to moderate mutation rates (≤ 0.10), identity closely follows the $(1 -$ mutation rate) relationship, with values of 0.99, 0.95, and 0.90 at rates of 0.01, 0.05, and 0.10, respectively. The mean gap count remains zero across this range because the mutation model used in this experiment introduces only substitutions, not altering sequence length. Under these conditions, the NW and SW scores, identities, and gap statistics are numerically indistinguishable, differing only in the fourth significant digit (for example, a mean score of 1850.0 for both algorithms at a mutation rate of 0.05).

This convergence between NW and SW results directly from the dataset's lack of structural variation. Local alignment provides an advantage over global alignment only when excluding part of the sequence improves the score, such as when one or both sequence ends are poorly matched, as observed with the indel-bearing variant pairs in Section A. When mutations are distributed as isolated substitutions across otherwise fully homologous sequence pairs, no sub-region exists whose exclusion would yield a higher score than the full-length alignment. In this scenario, the SW traceback recovers the same start-to-end path that NW computes by design. Thus, while the two algorithms are mathematically distinct, they are behaviorally equivalent under this mutation model.

A small but consistent gap count appears once the mutation rate exceeds 0.10, increasing from a mean of 2.0 gaps at 0.15 to 14.4 gaps at 0.30 for both algorithms. Under the linear scoring scheme applied here, two adjacent mismatches (-1 each, total -2) and a single gap (-2) are cost-equivalent in the recurrence. As mutation density increases, runs of consecutive substitutions occasionally create cells where a gap-bearing path scores equally well as an all-substitution path. The traceback rule then selects one of these tied optima, resulting in a nonzero gap count even though the mutation model did not introduce any insertions or deletions. This also explains why the mean identity slightly exceeds the naive $(1 -$ rate) line at higher mutation rates (for example, 0.7106 rather than 0.70 at a rate of 0.30): the mean alignment length increases beyond 1000 (1007.2 for NW) as gap columns are introduced, and the resulting identity ratio is computed over a slightly longer denominator that does not penalize the additional gap columns as mismatches.

D. Synthesis and Implications

Across the three experiments, alignment quality and computational performance cost are largely independent. Runtime is determined entirely by the O(mn) matrix-fill operation common to both algorithms and is unaffected by whether the alignment is global or local, as demonstrated by the near-identical timing curves in Figure 7 and the matching *nw_runtime_sec* and *sw_runtime_sec* columns in the variant dataset. In contrast, alignment quality depends on the structural relationship between the algorithms and the type of divergence present in the input sequences. When divergence consists of isolated substitutions distributed across full-length, equal-length sequences, as in Section C, NW and SW yield statistically equivalent scores and identities, since no partial alignment outperforms the full-length one. When divergence includes insertions or deletions concentrated in specific regions, as in the variant pairs of Section A, the two algorithms diverge sharply: NW incurs the full gap penalty required to reconcile the entire sequence length, while SW reports a higher-identity result by focusing on the conserved core and excluding the indel-bearing flanks.

This distinction determines the practical suitability of each algorithm. Needleman-Wunsch is appropriate when the analytical objective concerns the full-length relationship between two presumed-homologous sequences, such as estimating overall genomic divergence between viral lineages, because its score and identity directly reflect the cost of reconciling the complete sequences, including gaps. Smith-Waterman is preferable when the goal is to identify a high-confidence conserved region, regardless of the surrounding sequence, such as when selecting a stable target for primer design or detecting a shared domain between only partially homologous sequences. The experiments reveal that this trade-off is qualitative rather than computational. Choosing between global and local alignment changes the biological question being addressed, not the computational cost, since both algorithms share the same O(mn) time and space complexity derived from the underlying dynamic programming table.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization,” or “Magnetization, M,” not just “M.” If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization (A (m(1),” not just “A/m.” Do not label axes with a ratio of quantities and units. For example, write “Temperature (K),” not “Temperature/K.”

V. Conclusion

The Needleman-Wunsch (NW) and Smith-Waterman (SW) algorithms were compared for DNA sequence alignment using real SARS-CoV-2 variants and synthetic sequences of varying lengths and mutation rates. The observed differences between NW and SW were influenced by the types of sequence changes. In cases involving insertions and deletions, as seen in real variants, NW produced lower identity scores due to greater penalties for these changes, whereas SW maintained higher identity by emphasizing the longest matching segments. When differences were limited to substitutions, both algorithms yielded nearly identical alignment scores. Both methods demonstrated similar runtime patterns consistent with the expected O(mn) complexity, with minor differences at shorter sequence lengths attributed to fixed implementation overhead.

These results suggest that choosing between NW and SW depends on the biological question, not on speed. NW works best for comparing the overall similarity of full-length related sequences. SW is better for finding highly conserved regions, like primer-binding sites, in sequences that are only partly similar. Both algorithms take about the same amount of computing power, with the main differences in how they handle sequence ends and trace back through the alignment.

A limitation of this study is that the synthetic experiments incorporated only substitutions and excluded insertions or deletions. This constraint may limit the applicability of the findings to real-world scenarios where indels are prevalent. Future research should incorporate explicit indel events, evaluate longer genomic sequences to assess scalability, and explore alternative gap penalty models, such as affine gap schemes, to more accurately represent biological alignment challenges.

Video Link at Youtube

The presentation video explaining this paper, including the methodology and experimental results, can be accessed through the following YouTube link: https://youtu.be/3Pxk1tcJv6U

Appendix

The full source code of the experiment can be found on this github repository: https://github.com/aureliajenn/Paper-IF2211

Acknowledgment

The completion of this paper would not have been possible without the guidance of Prof. Dr. Ir. Rinaldi, M.T., whose commitment to teaching and thoughtful support throughout the Algorithm Strategy course have been truly invaluable. The writer is deeply grateful for the knowledge and direction provided during the development of this work.

References

[1] Mount, D. W. (2004). *Bioinformatics: Sequence and genome analysis* (2nd ed.). Cold Sprint Harbor Laboratory Press

[2] Hadfield, Hadfield, J., Megill, C., Bell, S. M., Huddleston, J., Potter, B., Callender, C., Sagulenko, P., Bedford, T., & Neher, R. A. (2018). Nextstrain: real-time tracking of pathogen evolution. *Bioinformatics*, *34*(23), 4121–4123. https://doi.org/10.1093/bioinformatics/bty407

[3] Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, *48*(3), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

[4] Muhamad, N. A., Yusof, R., & Md Yatim, N. F. (2015). Global vs. local alignment: What’s the difference? In *Proceedings of the 2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)* (pp. 95–99). IEEE. https://doi.org/10.1109/ACSAT.2015.37

[5] B. Alberts, A. Johnson, J. Lewis, D. Morgan, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 6th ed. New York, NY, USA: Garland Science, 2014.

[6] E. Domingo and J. J. Holland, "RNA virus mutations and fitness for survival," *Annual Review of Microbiology*, vol. 51, pp. 151–178, 1997, doi: 10.1146/annurev.micro.51.1.151.

[7] P. Lemey, M. Salemi, and A.-M. Vandamme, Eds., *The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2009.

[8] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, Mar. 1981, doi: 10.1016/0022-2836(81)90087-5.

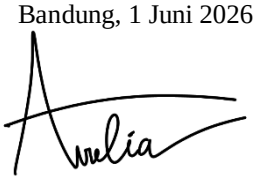
[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. Cambridge, MA, USA: MIT Press, 2022.

[10] R. Munir, "Program Dinamis (Dynamic Programming) Bagian 1 dan Bagian 2," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI-ITB, Bandung, Indonesia, 2026.

STATEMENT OF ORIGINALITY

I hereby declare that the work presented in this paper is entirely my own. It is not a copy, translation, or adaptation of any other author's work, and it does not constitute plagiarism.

Bandung, 1 Juni 2026



Aurelia Jennifer Gunawan
13524089