Pan-genome Sequence Alignment Algorithm Research

LuFang Yu

School of Big Data Statistics, Guizhou University of Finance and Economics, Guiyang, Guizhou Province

Received: 04 September 2025/ Revised: 12 September 2025/ Accepted: 21 September 2025/ Published: 30-09-2025

Abstract— *With the rapid development of sequence alignment technologies, their role in responding to public health emergencies and epidemic prevention has become increasingly prominent. Compared to traditional linear reference sequences, the pan-genome effectively reduces reference bias and misalignment. However, when facing long reference sequences with abundant repetitive regions, candidate positions may proliferate, thereby compromising alignment efficiency and accuracy. This paper proposes a novel alignment algorithm that integrates elastic degenerate strings with the longest common substring strategy. Single nucleotide polymorphisms (SNPs) are linearly encoded using elastic degenerate symbols, while minimizer indexing is combined with a longest common substring interval–based seed filtering strategy to effectively reduce candidate positions. Experimental validation on simulated datasets demonstrates that the proposed method significantly improves both recall and precision rates, while also achieving high accuracy in fragment localization. Further experiments on real datasets reveal that this method outperforms existing mainstream alignment tools in terms of alignment sensitivity, indicating that the longest common substring–based filtering strategy is well suited for complex genomic regions. Overall, this approach provides an alternative technical pathway for enhancing the accuracy of pan-genome sequence alignment and offers a feasible algorithmic framework for subsequent research on pan-genome alignment and viral mutation analysis.*

Keywords— Sequence Alignment; Seed-and-Extend; Elastic Degenerate String; Longest Common Substring.

I. INTRODUCTION

At the end of 2019, the outbreak of pneumonia caused by the novel coronavirus (SARS-CoV-2) rapidly spread worldwide, leading to severe economic losses and significant healthcare burdens[1][2]. Existing studies have identified single-nucleotide polymorphism (SNP) sites and related genes potentially associated with pneumonia symptoms, and some mutation sites have even been shown to influence vaccine efficacy, thereby posing a potential risk of further enhancing the transmissibility of COVID-19[3][4]. When dealing with highly variable viruses such as SARS-CoV-2, researchers must rapidly collect a large number of samples from infected individuals. Using high-throughput sequencing and sequence alignment technologies, they are able to analyze the pathogenic mechanisms, transmission routes, and evolutionary relationships of the virus in depth. These studies provide crucial scientific evidence for vaccine development and epidemic control. However, performing pairwise sequence alignment between short reads and long reference genomes presents substantial computational challenges, requiring not only algorithmic efficiency but also accuracy in short-read alignment. Therefore, improving the efficiency and accuracy of existing sequence alignment algorithms is of significant research value and practical importance for advancing viral genomics research and epidemic surveillance.

With the advancement of sequencing technologies and in consideration of the length and complexity of genomic sequences, an increasing number of alignment algorithms construct auxiliary index structures for reference genomes to accelerate the alignment process. Based on the type of index used, alignment algorithms can generally be divided into two categories: suffix-based indexing and hash-based indexing. Suffix-based methods typically employ suffix arrays[5], Burrows-Wheeler Transform BWT) [6], and FM-index[7] for alignment, and are suitable for large-scale genomic data such as the human reference genome. Although such methods do not require additional positioning operations, they consume substantial memory resources. For relatively short reference genomes such as those of viruses, hash-based indexing is often preferred for rapid sequence analysis

in order to obtain more accurate alignment results. In this process, however, the seed selection strategy plays a pivotal role in enhancing both the efficiency and accuracy of sequence alignment algorithms.

Regarding seed selection strategies, numerous studies have proposed effective improvements. Roberts et al. introduced the concept of the minimizer[8], which selects the k-mer with the smallest hash value within a consecutive window of seeds as the alignment seed. This approach significantly reduces the storage space required for indexing and has led to the development of alignment tools such as Giraffe[9] and GraphAligner[10]. While this method greatly reduces memory usage and computational overhead in alignment, the abundance of repetitive sequences in long reference genomes often leads to false positives caused by high-frequency minimizers. To address this, the weighted-minimizer approach[11] was proposed, assigning weights to minimizers according to their frequency in the reference genome, thereby prioritizing seeds with higher weights for read placement. However, relying solely on single-seed extension can result in locally optimal alignments, necessitating consideration of relationships among multiple seeds. By employing collinear chaining[12] in the intermediate seed extension step, seeds that satisfy collinearity constraints are selected, producing higher-quality seed sets and improving alignment accuracy. One of the most widely used pairwise sequence alignment algorithms, Minimap2[13], adopts this strategy, substantially enhancing long-read alignment efficiency. Similarly, the graph-based alignment tool GED-MAP[14] leverages collinear chaining to accelerate candidate position filtering, achieving both faster alignment and reduced memory usage. For short reads, however, directly computing collinear chains does not provide significant alignment optimization benefits and can even increase computational costs during the filtering phase. Ding Shengnan et al. proposed a region-based coarse-grained filtering method, which accelerates alignment using sliding window indexing tables and filtering formulas. Nevertheless, this method faces challenges of computational and memory consumption when applied to large-scale datasets[15]. Song Siyi et al. introduced a low-frequency seed voting strategy to accelerate alignment, but its scalability is limited and requires improvements in indexing structures to accommodate large genome datasets[16]. Gao Jia and Xu Yun further combined minimizer-based seeding strategies to reduce false-positive alignments, though this approach may still encounter mismatches in repetitive regions[17]. While these methods have made progress in enhancing alignment efficiency, they continue to face limitations when applied to pan-genomes or large-scale complex datasets.

To address the above issues, this paper proposes a semi-global alignment algorithm based on longest common substring intervals. Specifically, the improvements of this study focus on two aspects: first, by employing elastic degenerate symbols, SNP variations are incorporated into the reference genome, providing a concise linear representation of the pan-genome. Second, given the critical role of the longest common substring in biological sequences, this study introduces a seed filtering strategy based on longest common substring intervals.

II. BASIC CONCEPTS AND METHODS

2.1 Basic Concepts:

In the analysis of high-throughput sequencing data, particularly the massive short fragments generated by second-generation sequencing technologies, the limited read length often carries insufficient biological information to independently reveal their functions or significance. Therefore, it is necessary to align these short reads against a reference genome to determine their potential genomic locations, a process referred to as sequence alignment[18]. For sequence alignment problems, Hamming distance and edit distance are two commonly used similarity measures.

Hamming distance[19] refers to the number of single-character substitution operations required to transform one string into another. Edit distance[20], on the other hand, denotes the minimum number of edit operations needed to convert one string into another, where the allowed edit operations include substitution, insertion, and deletion. Considering that sequencing errors in reads often involve substitutions, insertions, and deletions, and that genetic variation across different samples also introduces insertions and deletions, edit distance is generally a more realistic measure.

With the advancement of sequence analysis technologies, using a single reference genome in alignment often fails to adequately capture population-specific variations, thereby increasing the difficulty of identifying disease-related genetic associations. To address this challenge, the concept of the pangenome was introduced in the context of studying the dynamic nature of bacterial genomes. A pangenome refers to the collection of multiple genomic sequences from the same species[21]. In recent years, an

increasing number of studies have shifted toward graph-based pangenomes, which help mitigate reference bias and misalignment issues. Specifically, a pangenome graph is a graph representation of the pangenome, where each node corresponds to a nucleotide sequence, and each path through the graph represents a genotype, as illustrated in Figure 1.

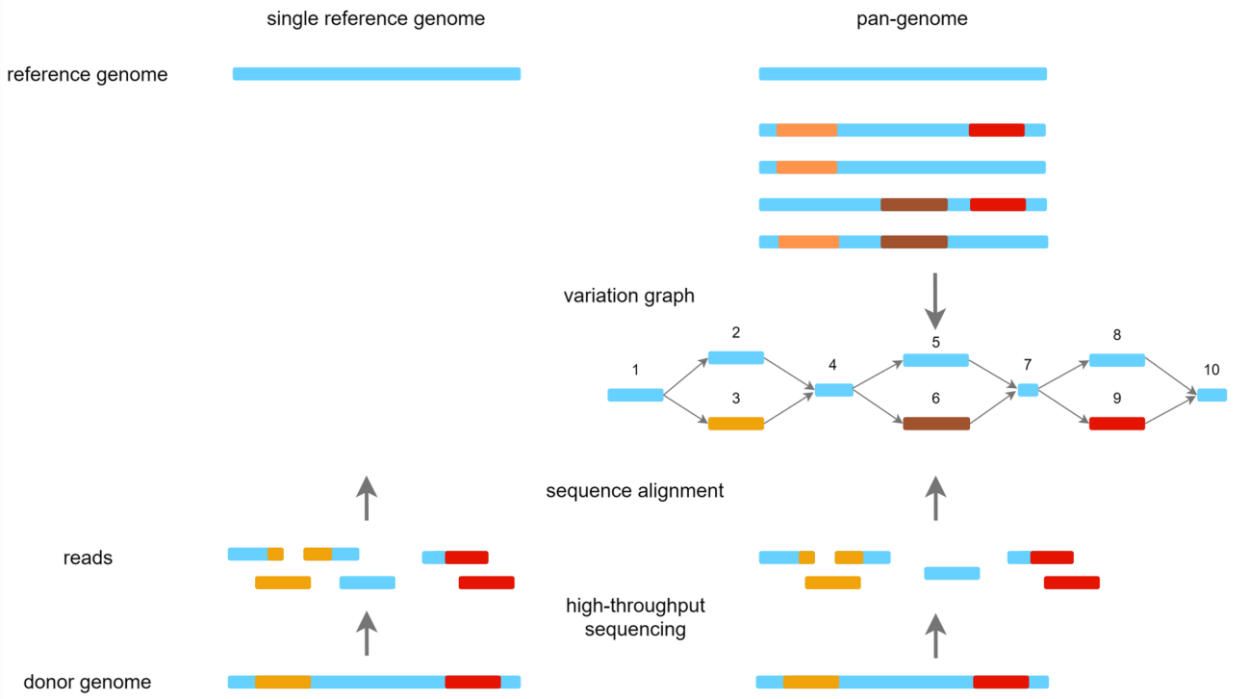


FIGURE 1: Comparison between a single reference genome and a pangenome

Figure 1 illustrates alignment scenarios based on different reference genomes. A pangenome compresses the linear sequences of a population into a variation graph. In this representation, nodes correspond to specific nucleotide sequences; for instance, nodes 1, 4, 7, and 10 represent fragments shared across all genomes, while variant fragments form "bubble" structures within the graph. A path constructed from a series of adjacent nodes represents a possible biological sequence. Compared to the traditional linear reference genome, this approach not only maximizes sequence similarity to reduce redundant storage but also provides a more intuitive and effective way to represent genetic variation within a population.

2.2 Methods:

2.2.1 Elastic-Degenerate Strings:

Having established the concept of the pangenome, it is clear that while graph representations can capture a broader range of genetic variation, they also introduce additional complexity. Functions that are relatively trivial on linear genomes often become more challenging on genome graphs. To represent SNP variations concisely on a graph, the Elastic-Degenerate Strings (EDS) method[22] is adopted. The following provides detailed definitions and illustrates how SNP variations can be expressed.

Definition 1 Elastic-Degenerate Symbol (ξ):

Given the character set $\sum = \{A, C, G, T\}$, where ε denotes the empty character, and $\sum^*$ represents the set of all strings over $\sum$ including ε. An elastic-degenerate symbol ξ is defined as a finite, non-empty set of strings $(\xi \in \sum^* \text{ and } \xi \neq \emptyset)$, which can be expressed as $\xi = [E_1|\ldots|E_m]$, where each E_i is referred to as an alternative string.

Definition 2 Elastic-Degenerate String $(\hat{S})$:

An EDS over $\sum$ is a string consisting of a sequence of elastic-degenerate symbols, i.e., $\hat{S} = \xi_1 \ldots \xi_n$。

Definition 3 Possibility Set $(\Re)$: A string Y matches an elastic-degenerate string $\hat{S}$ if and only if $Y \in \Re$, where $\Re$ is the set of all possible strings generated from $\hat{S}$. In other words, by substituting each elastic-degenerate symbol with one of its alternative strings, a new string can be obtained.

To better illustrate the above definitions, consider a pangenome represented by the elastic-degenerate string $\hat{P} =$ CG[T|G]GCCA[C|T]AT. Here, [T|G] denotes an elastic-degenerate symbol at position 3, which can be substituted by either T or G; similarly, [C|T] represents another substitution at the corresponding position.

2.2.2 Seed-and-Extend Method:

In sequence alignment algorithms, the Seed-and-Extend strategy[23] is widely employed. Its primary purpose is to narrow the alignment search space from the entire reference genome to a subset of candidate regions by means of seed localization. This method generally consists of three steps:

Step 1. Extract fixed-length substrings (k-mers) from the read. These substrings are referred to as seeds.

Step 2. Perform rapid seed lookup using an index. With a pre-built reference genome index, the positions of the read seeds in the genome can be efficiently queried. Candidate positions of poor quality are then filtered out according to specific strategies.

Step 3. Map the seed positions onto the reference genome and extend them in combination with the read length to generate candidate regions, which are then aligned with the read at base-level resolution. The alignment quality is assessed using dynamic programming algorithms, producing the final matching location. The overall workflow of the Seed-and-Extend method is illustrated in Figure 2.

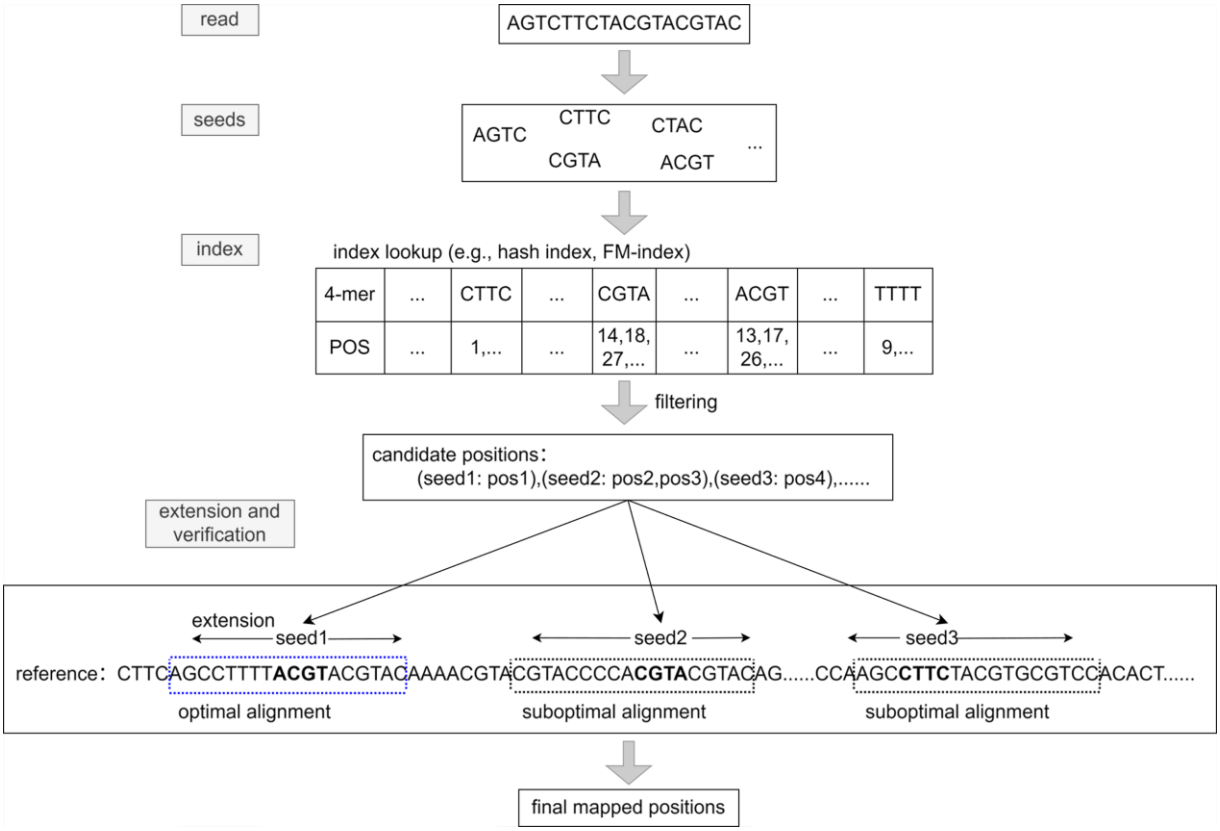


FIGURE 2: Seed-and-Extend method

As shown in Figure 2, the seeds generated from a read can result in numerous redundant and erroneous candidate positions, which must be filtered out. Therefore, while maintaining alignment sensitivity, it is essential to adopt effective filtering strategies for low-quality seeds in order to reduce the computational burden during the verification stage and to obtain the final alignment positions efficiently.

2.2.3 Longest Common Substring:

The Longest Common Substring (LCSstr)[24] refers to the longest identical substring that appears contiguously in both strings. In contrast, the Longest Common Subsequence (LCS)[25] is the longest sequence that appears in two or more sequences, with elements occurring in the same relative order in all sequences but not necessarily contiguously. Their applications differ: LCSstr

is suitable for scenarios requiring strictly contiguous matches (e.g., genomic sequence alignment), whereas LCS is applicable to tasks where order preservation is required but interruptions are allowed (e.g., text difference comparison).

For two given strings S_1 and S_2, the LCSstr can be computed using a dynamic programming algorithm. A two-dimensional array $c[i,j]$ is defined, where $c[i,j]$ denotes the length of the LCSstr ending at $S_1[i]$ and $S_2[j]$. The values of $c[i,j]$ can be computed using recurrence relation (1).

$$c[i,j] = \begin{cases} 0, & i = 0 \ or \ j = 0 \\ c(i-1, j-1) + 1, & if \ S_1[i] = S_2[j] \\ 0, & else \end{cases} \quad (1)$$

By filling the entire array according to this recurrence, the maximum value in the array represents the length of the LCSstr between S_1 and S_2. For example, given the read $r = \mathrm{GTGCCATT}$ and the pangenome $\hat{P} = \mathrm{CG[T|G]GCCA[C|T]AT}$, the longest common substring is "GCCA". To enable subsequent seed filtering, it is necessary to record the starting and ending indices of all maximum LCSstr occurrences in both the read r and the pangenome $\hat{P}$, denoted as $[r_i, r_j]$ and $[\hat{P}_i, \hat{P}_j]$, respectively. In this case, $[r_i, r_j] = [2,5]$ and $[\hat{P}_i, \hat{P}_j] = [7,10]$.

III. ALIGNMENT ALGORITHM DESIGN

The sequence alignment algorithm proposed in this study is primarily based on the Seed-and-Extend paradigm, and the entire alignment process can be divided into three major stages: (1) Index construction; (2) Seed selection; (3) Extension and verification. The specific workflow is illustrated in Figure 3.

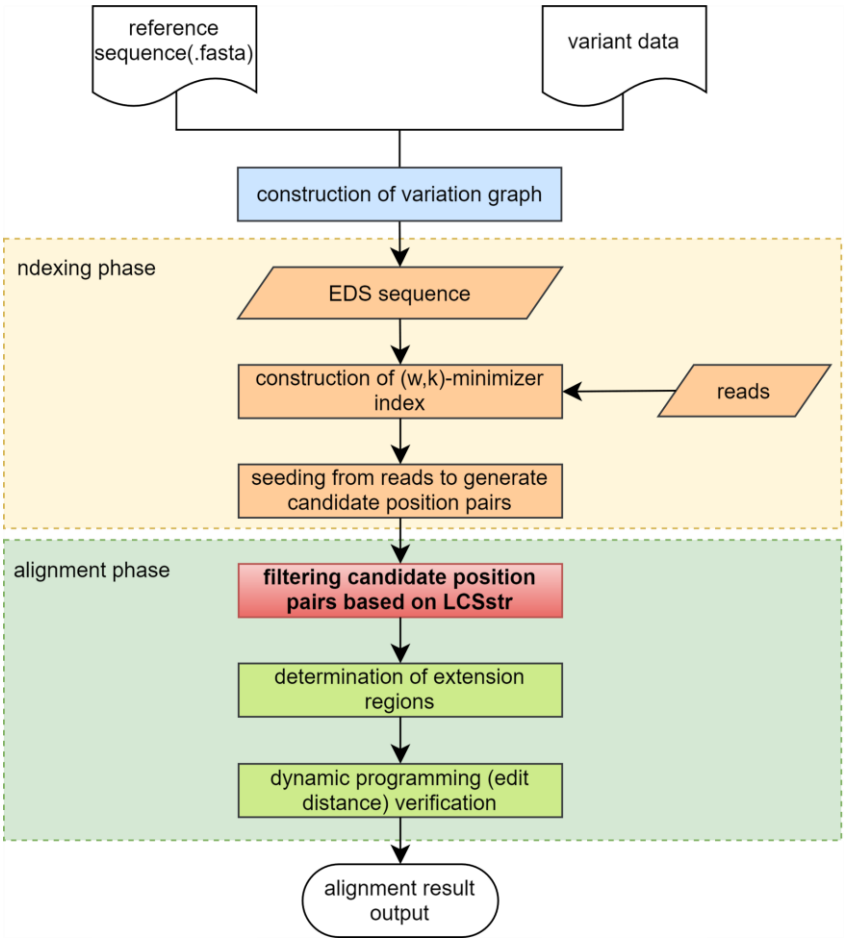


FIGURE 3: Workflow of the sequence alignment algorithm

In the indexing stage, minimizer indexes are constructed for both the pangenome $\hat{P}$ and the read r, and stored in hash tables. In the seed selection stage, the positions of seeds from the read are queried against the pangenome, forming a series of position pairs. These numerous candidate positions are then filtered using the LCSstr ranges to obtain the candidate seeds. In the

alignment stage, the candidate regions are determined by combining seed positions with read length extension, followed by base-level verification using dynamic programming to produce the final alignment results. The sections marked in red in Figure 3 highlight the main improvements proposed in this paper. The following subsections provide a detailed description of these three stages.

3.1 Index Construction:

To further reduce memory overhead during the sequence alignment process, this study employs the (w, k)-minimizer method[8]. The core idea is to select, from every consecutive set of w k-mers (each of length k), the lexicographically smallest k-mer as the seed, which is referred to as the minimizer. The sequence formed by these consecutive w k-mers is called a window, and its length is defined as $win = k + w - 1$. In this paper, lexicographic order is adopted as the sorting rule. Figure 4 illustrates the (4,3)-minimizer computation for the example read $r = \mathrm{GTGCCATT}$. Each row represents a window of length 6, with the bolded 3-mer indicating the minimizer for that window.

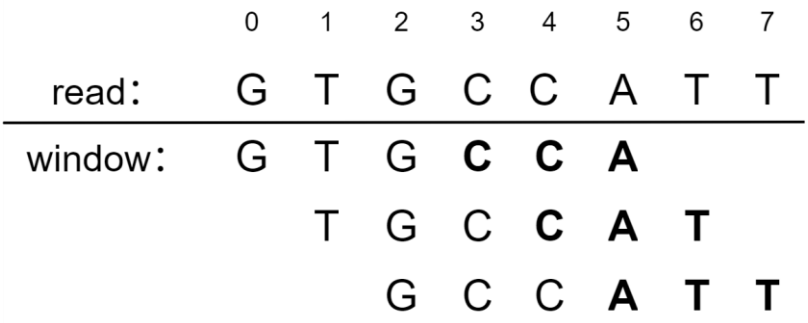
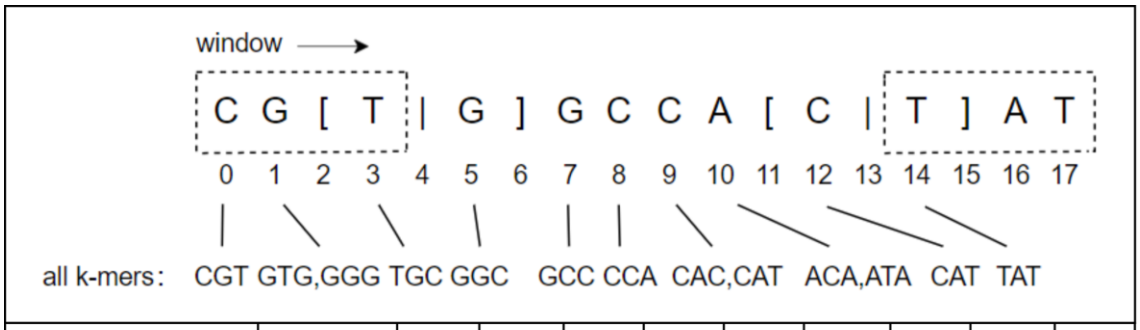


FIGURE 4: Index table construction for the read

When constructing the minimizer index table for the pangenome $\hat{P}$ special syntactic symbols ("[", "|", "]") must be considered. Specifically, each alternative option within an elastic-degenerate symbol needs to be evaluated, while k-mers overlapping with syntactic symbols are excluded. If consecutive windows produce the same minimizer, only the positional information of that minimizer is added to the index table. Figure 5 shows the process of generating the index table for the pangenome $\hat{P} = \mathrm{CG[T|G]GCCA[C|T]AT}$.



(4,3)-minimizer Index:	Minimizer	CGG	GGG	GGC	GCC	**CCA**	CAC	ACA	**CAT**	TAT
	Positions	0	1	5	7	8	9	10	12	14

FIGURE 5: Minimizer index table construction for the pangenome

As can be seen, in real reference genomes, due to long sequence lengths and abundant repetitive regions, the sliding window technique can effectively reduce memory requirements. This not only preserves alignment sensitivity but also provides a more efficient computational foundation for downstream sequence analysis and alignment tasks. However, it is important to note that the construction of minimizer index tables involves a trade-off between sensitivity and speed. Therefore, appropriate values of w and k must be carefully chosen. Algorithm 1 presents the pseudocode for computing the minimizer set of a pangenome using the sliding window approach.

Algorithm 1： Minimizer Index Construction Algorithm for the Pangenome

Input： pangenome $\hat{P}$， window length w， substring length k

Output： index table $index$

initialization： $L = len(\hat{P}); R_s \leftarrow []; R_i \leftarrow [0] * L; i \leftarrow 0; index \leftarrow \{\}; start \leftarrow 0$

while $i < L$ **do** // construct the compressed sequence R_s and the index mapping R_i

 if $P[i] \neq$ "[" **then**

 $R\_i[i] \leftarrow len(R_s); R\_s.append(P[i]); i \leftarrow i + 1$

 else

 treat the ED symbol as an atomic unit and update the index into R_i

 append the lexicographically smallest character within the ED symbol to R_s

while $start + w <= L$ **do**

 $end \leftarrow start + w - 1$

$A \leftarrow []$

for $j \leftarrow (start, end + 1)$

 if $P[j] \in \{'[', '|', ']'\}$ **continue**

 $idx \leftarrow R\_i[j]$

 if $idx + k > len(R\_s)$ **continue**

 $r \leftarrow P[j] + R\_s[idx + 1 : idx + k]$

 $A \leftarrow A \cup \{(r, j)\}$

if $A = \emptyset$ **continue**

$r' \leftarrow minLex(A)$

for $(r', p) \in A$

 $index[r'].appendUnique(p)$

return $index$

3.2 Seed Selection:

Considering that during biological variation and evolution, certain identical gene sequences are retained, these regions are referred to as conserved regions[26] . Conserved regions generally play important physiological or genetic roles. The Longest Common Substring (LCSstr) can reflect a degree of conservation, making it an important reference in alignment. Therefore, this study employs the LCSstr interval to filter seeds.

By querying the positions of seeds from the read in the reference genome index table, a series of position pairs $(P_r, P_{\hat{P}})$ can be obtained. These position pairs $(P_{r_n}, P_{\hat{P}_n})$ are then filtered using the LCSstr interval. If a position pair satisfies Formula (2), it is retained as a candidate position pair (i^*, j^*) for subsequent extension and verification. In this formula, r_i and r_j denote the start and end indices of the LCSstr within the read r, while $\hat{P}_i$ and $\hat{P}_j$ denote the start and end indices of the LCSstr within the pangenome $\hat{P}$.

$$r_i \leq P_{r_n} \leq r_j \;\&\; \hat{P}_i \leq P_{\hat{P}_n} \leq \hat{P}_j \tag{2}$$

If no candidate positions remain after LCSstr-based filtering, the relationships among seeds are further considered by applying the colinear chaining method[12] to filter the initially formed position pairs. The colinearity rule specifies that two position pairs (x, y) and (x', y') are colinear if and only if $x < x'$ and $y < y'$, denoted as $(x, y) < (x', y')$. The score of a position pair (x, y) is defined as the length of the maximum colinear chain ending at (x, y). Finally, the chain with the highest score is used for base-level alignment. If the resulting alignment is unsatisfactory, the position pair from the chain with the second-highest score is selected for alignment.

In the example above, querying the seeds from Figure 4 in the pangenome index table yields candidate seeds, as indicated by the bolded minimizers in Figure 5. By combining their positions in r and $\hat{P}$, the position pairs (3,8) and (4,12) are obtained. Finally, by computing the interval of the longest common substring between the read and the pangenome, the final candidate position retained for alignment is (3,8). Algorithm 2 provides the pseudocode for the seed filtering algorithm based on the LCSstr interval.

Algorithm 2： Seed Filtering Algorithm Based on Longest Common Substring Intervals

Input： pangenome $\hat{P}$， its index M_idx， read set R， and its index R_idx， seed score s

Output： candidate seed set $filt$

initialization： $pos \leftarrow []; filt \leftarrow []; s \leftarrow []$
for $r \in R$ **do**
 for (k, p_r) in $R\_idx[r]$ **do**
 if $k \in M\_idx$ **then**
 for p_p in $M\_idx[k]$ **do**
 $pos \leftarrow pos \cup \{(p\_r, p\_p)\}$
for $(p\_r, p\_p) \in pos$ **do**
 for (r_s, r_e, p_s, p_e) in LCSstr(r, P) **do**
 if $r\_s \leq p\_r \leq r\_e$ & $p\_s \leq p\_p \leq p\_e$ **then**
 $filt \leftarrow filt \cup \{(p_r, p_p)\}$
if $filt = \emptyset$ **then**
 for $(x, y) \in pos$ **do**
 $s(x, y) \leftarrow |\{(x', y') \in pos | x' < x \text{ and } y' < y\}|$
 $filt \leftarrow filt$ store in descending order of s
return $filt$

3.3 Extension and Verification:

In the verification stage, the filtered candidate positions are mapped to the reference genome, and appropriate extension strategies are applied to determine candidate regions. Finally, the edit distance between the read and these candidate regions is computed. This process effectively narrows down the alignment scope: instead of performing a time-consuming alignment of the short read against the entire long reference genome, the comparison is confined to localized regions of the reference sequence, thereby significantly improving alignment efficiency.

Specifically, the forward alignment fragment of the read can be represented as $r[i^*, m]$, where i^* denotes the starting position of the candidate seed in the read, and m is the read length. Correspondingly, on the pangenome $\hat{P}$, the forward alignment fragment is extended based on the coordinate j^* of the candidate seed combined with the read length. The final alignment is then computed using a variant of the edit distance algorithm[14], provided in Algorithm 3.

Algorithm 3： Dynamic Programming-based Edit Distance Variant Algorithm

Case1： $\hat{P}[j] \in \sum$

$$D_{i,j} = \begin{cases} i, if\ j = 0 \\ j, if\ i = 0 \\ min \begin{cases} D_{i-1,j-1}\ if\ r[i] = P[j] \\ D_{i-1,j-1} + 1\ if\ r[i] \neq P[j] \\ D_{i,j-1} + 1 \\ D_{i-1,j} + 1 \end{cases} \end{cases}$$

Case2： $\hat{P}[j] =$ "["

$$D_{i,j} = D_{i,j-1}$$

Case3： $\hat{P}[j] =$ "|"

$$D_{i,j} = D_{i,j'}, if\ P[j'] = '['$$

Case4： $\hat{P}[j] =$ "]"

$$D_{i,j} = min(D_{i,j''})\ if\ j' \leq j'' < j\ and\ P[j''+1] \in \{|,\}\}$$

Starting from the selected seed position (i^*, j^*), forward alignment is performed. Let $D_{i,j}$ represent the edit distance from the sequence fragment $r[i^*, i]$ to the fragment $\hat{P}[j^*, j]$. Algorithm 3 describes the method for calculating the edit distance in the presence of elastic degenerate symbols, which is divided into four cases. The first case follows the standard definition of edit distance, while the second, third, and fourth cases provide explicit rules for handling the special syntax symbols "[", "|", and "]". Specifically:

If $\hat{P}[j] =$ "[" , then the value of column j is set equal to the value of column $j-1$.

If $\hat{P}[j] =$ "|" , then the value of column j corresponds to the value of a specific column j', where $P[j'] =$ "[", meaning that under the constraints of the syntax definition, the value from column j' is adopted.

If $\hat{P}[j] =$ "]", then the value of column j is the minimum among the values of certain columns j'', where the constraint requires $P[j''+1] \in$ {"|","}"}, indicating that the minimum value is selected based on the column corresponding to the syntax symbol.

This variant of edit distance calculation enables more accurate and effective handling of alignment tasks involving syntax symbols, thus improving precision in pangenome sequence alignment.

In the example above, the forward alignment fragment of r is $r[3,8] =$ CCATT, and the corresponding forward alignment fragment of $\hat{P}$ is $\hat{P}[8,17] =$ CCA[C|T]A. Using Algorithm 3 to compute the edit distance between these two fragments yields the alignment result shown in Figure 6.

	$\hat{P}$	C	C	A	[	C	|	T	]	A
r	0	1	2	3	3	4	3	4	4	5
C	1	0	1	2	2	3	2	3	3	4
C	2	1	0	1	1	2	1	2	2	3
A	3	2	1	0	0	1	0	1	1	2
T	4	3	2	1	1	1	1	0	0	1
T	5	4	3	2	2	2	2	1	1	1

alignment result

C	C	A	[	C	|	T	]	A
|	|	|				|		
C	C	A	-	-	-	T	-	T

FIGURE 6: Edit Distance computation and traceback path

From the lower-right corner of the dynamic programming matrix in Figure 6, it can be observed that the edit distance between the read and the candidate region in the pangenome is 1. The red-highlighted values represent the optimal alignment path, from which the best alignment result of the example is obtained through traceback. Similarly, the reverse alignment between r and

$\hat{S}$ can also be computed, where reverse alignment is treated as the forward alignment of the reversed string. Finally, combining the forward and reverse alignment results yields the complete alignment outcome.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experimental Data and Evaluation Metrics:

The experimental data used in this study were obtained from the China National Center for Bioinformation (CNCB). From this database, the SARS-CoV-2 reference sequence NC_045512.2 (length: 29,903 bp) and its corresponding variant data were downloaded. Considering the generation of the longest common substring and the presence of special syntax markers in elastic-degenerate symbols, a limited number of variants were introduced to the reference sequence to minimize potential interference. After preprocessing steps such as deduplication and removal of overlapping variants, a total of 1,383 SNPs were retained and incorporated into the reference sequence using elastic-degenerate symbols, generating a pangenome for alignment.

For the read datasets, experiments were conducted on both simulated and real datasets. To evaluate the performance of the proposed method in precise alignment, a series of comparative experiments were performed with leading sequence alignment algorithms. Two mainstream BWT-based aligners, BWA and Bowtie2, were selected. BWA includes three alignment algorithms; the two most performant, BWA-SW and BWA-MEM, were used in the experiments. Additionally, the popular hash-based aligner Minimap2 was included as a control. All tools were run with default parameters.

In experiments with the simulated dataset, the recall rate and precision rate were calculated by comparing the alignment positions reported by the algorithms with the true positions of reads**Error! Reference source not found.**. The recall rate reflects the ability of the alignment tool to correctly locate reads on the reference sequence, while the precision rate measures whether reads are aligned to their original positions on the reference sequence. The formulas are as follows**Error! Reference source not found.** :

$$recall\ rate = \frac{TM}{N} * 100\% \tag{3}$$

$$precision\ rate = \frac{TM}{M} * 100\% \tag{4}$$

where N is the total number of reads in the dataset, M is the number of reads that can be aligned to the reference sequence, and TM is the number of reads correctly aligned to their original positions on the reference. A read is considered correctly aligned if the difference between its reported position and true position falls within a predefined threshold.

For experiments with the real dataset, since the exact positions of sequenced reads on the reference genome are unknown, precision cannot be directly assessed. Instead, sensitivity is used as the evaluation metric. Sensitivity measures the proportion of correctly identified bases by the alignment algorithm and is calculated as follows**Error! Reference source not found.**:

$$sensitivity = \frac{M}{N} * 100\% \tag{5}$$

4.2 Results of Simulated Dataset Experiments:

For the simulated datasets, considering that the constructed pangenome contains elastic-degenerate symbols, the following procedure was adopted: first, a random starting position was uniformly selected in the pangenome sequence, and a segment of a specified length was extracted from this position, with its location in the reference sequence recorded. For known variant sites, one alternative allele was randomly selected for substitution. Errors were then introduced into the extracted subsequence, including base substitutions, insertions, or deletions, with an error probability of 0.1% per base.

In terms of experimental setup, seven short-read simulated datasets were generated with read lengths ranging from 50 bp to 150 bp, specifically 45 bp, 65 bp, 85 bp, 105 bp, 125 bp, 145 bp, and 165 bp, with each dataset containing 100,000 reads. Additionally, to evaluate the applicability of the proposed method for long reads, seven long-read simulated datasets were constructed with read lengths of 200 bp, 250 bp, 300 bp, 350 bp, 400 bp, 450 bp, and 500 bp, also containing 100,000 reads each.

In all experiments, (20,5)-minimizers were used to construct the index tables, and algorithm performance was tested across datasets of different scales. Figure 7 presents the alignment results of different algorithms on the simulated short-read datasets.

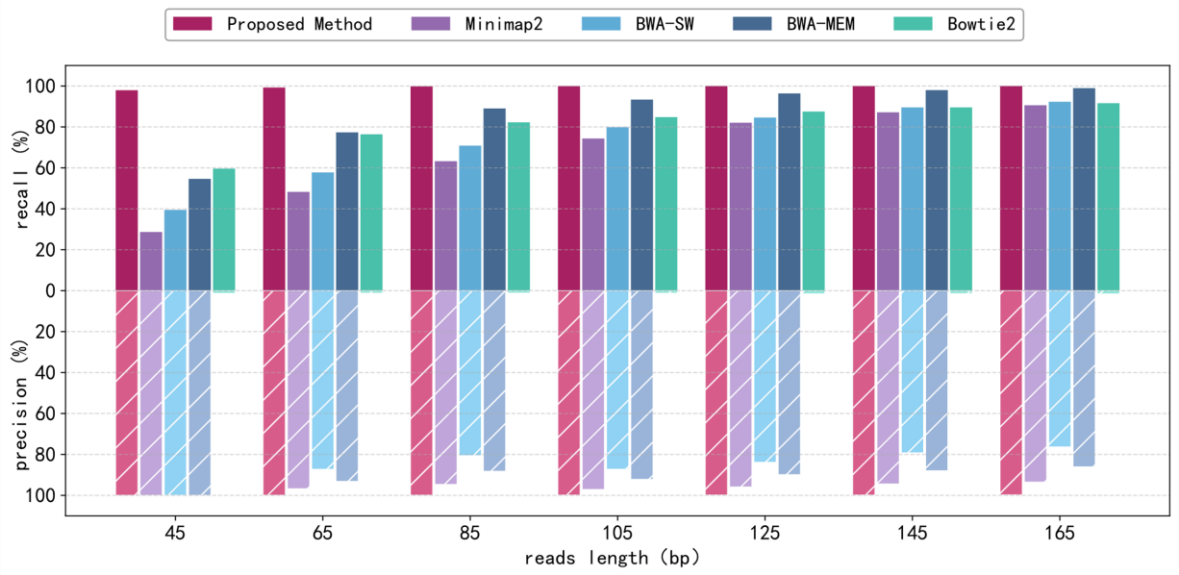


FIGURE 7: Recall and precision for short-read datasets aligned to the pangenome

From Figure 7, it can be observed that for short-read alignment to the pangenome, the proposed method outperforms mainstream alignment tools in both recall (97.94%–99.99%) and precision (99.72%–99.93%). In comparison, Minimap2 achieved a recall of 28.68%–90.60% and precision of 93.47%–99.94%; BWA-SW had a recall of 39.46%–92.30% and precision of 76.27%–99.91%; BWA-MEM reached a recall of 54.62%–98.95% and precision of 86.01%–99.93%; Bowtie2 showed a recall of 59.63%–91.62% but relatively lower precision. This reduced precision is likely due to its alignment strategy, which may produce nonspecific mappings when short reads contain insufficient information, thereby affecting accuracy.

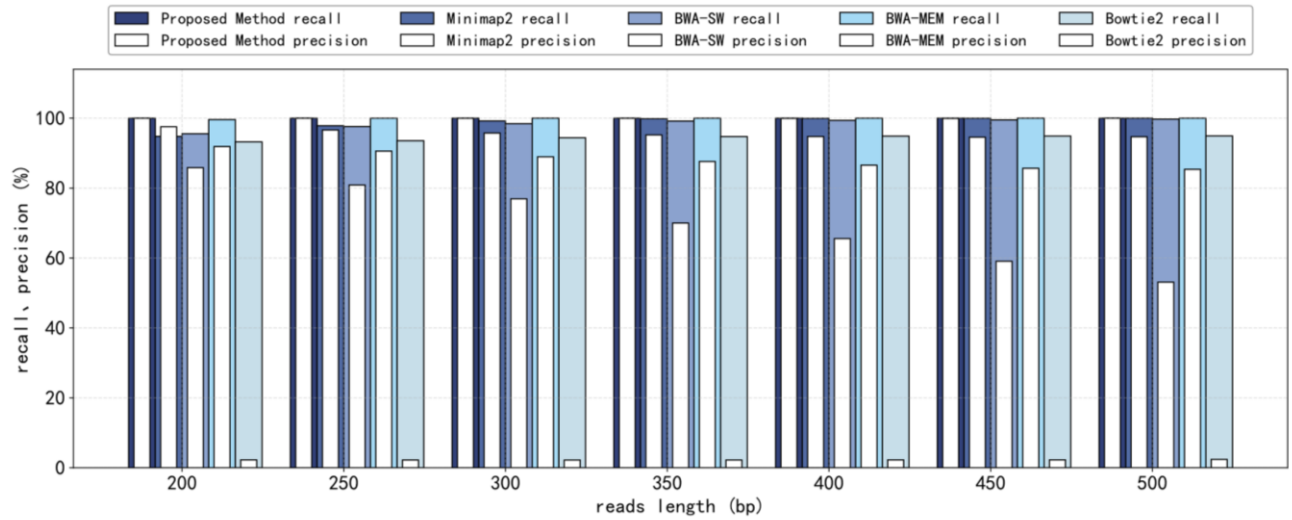


FIGURE 8: Recall and precision for long-read datasets aligned to the pangenome

Furthermore, as shown in Figure 8, for long-read alignment, the proposed method exhibits even more prominent advantages. Specifically, in long-read experiments, the method achieved a recall of 99.99%–100% and a precision of 99.94%–99.99%, indicating its ability to accurately locate reads on the reference sequence. In comparison, Minimap2 also performed stably in long-read scenarios, maintaining high recall and precision. The BWA algorithms remained competitive in recall but had slightly lower precision overall, whereas Bowtie2 performed poorly in both recall and precision. This is likely due to its design targeting short reads and reliance on an FM-index-based local alignment strategy, which may result in higher mismatch rates for long reads and consequently reduce overall precision.

In addition to recall and precision, positional error was introduced as a supplementary metric to evaluate the distance between the aligned and true positions. Smaller positional errors indicate higher accuracy in read placement. Figure 9, shown on a semi-logarithmic scale, presents the average positional errors for aligned reads across simulated short-read and long-read datasets.

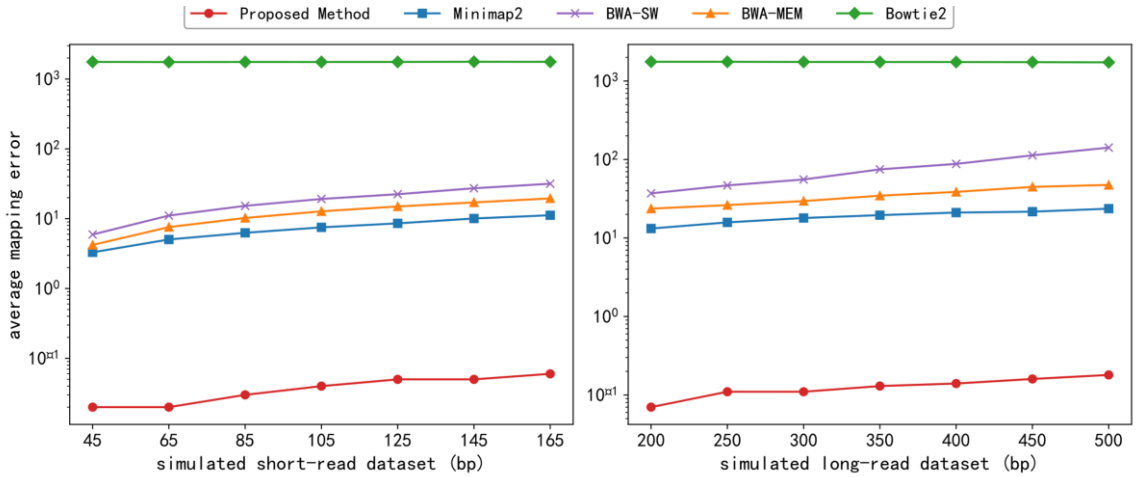


FIGURE 9: Average positional error of different algorithms on simulated datasets

As seen in Figure 9, the proposed method consistently outperforms existing tools in terms of average positional error for both short and long reads, demonstrating higher placement accuracy. Comparative results indicate that Minimap2 and BWA exhibit similar levels of positional error, while Bowtie2 shows the largest error, reflecting a higher proportion of false-positive alignments. These results further validate the superiority of the proposed method in alignment accuracy.

4.3 Results of Real Dataset Experiments:

Since simulated datasets cannot fully replicate the structural variants and sequencing errors present in real sequencing data, alignment rates and accuracy are typically higher for simulated reads than for real reads. To better evaluate the performance of the proposed algorithm in practical applications, comparative experiments were conducted on multiple real sequencing datasets.

For the real sequencing datasets, SARS-CoV-2 sequencing data were downloaded from the Sequence Read Archive (SRA) of the National Center for Biotechnology Information (NCBI). From datasets SRR33976238, SRR33976240, SRR34288732, and SRR31052680, the first 100,000 single-end reads of lengths approximately 51 bp, 76 bp, 102 bp, and 152 bp, respectively, were extracted for the experiments. Additionally, experiments were conducted on four longer-read datasets: the first 100,000 reads of lengths approximately 202 bp and 251 bp from SRR31052679, the first 100,000 reads of length approximately 300 bp from SRR11810739, and 81,000 reads of length approximately 351 bp from SRR11788183.

The four groups of real SARS-CoV-2 sequencing reads were aligned to the constructed pangenome, and the results are shown in Table 1.

TABLE 1
ALIGNMENT SENSITIVITY FOR REAL SARS-COV-2 SEQUENCING DATA

Comparison tools	Dataset			
	51bp	76bp	102bp	152bp
Proposed Method	56.74%	72.63%	**74.16%**	**98.98%**
Miinimap2	30.57%	47.06%	60.22%	76.08%
BWA-SW	41.54%	57.54%	61.67%	86.73%
BWA-MEM	58.35%	**76.34%**	73.28%	97.20%
Bowtie2	**58.43%**	56.21%	61.29%	68.11%

From Table 1, it can be observed that due to the higher complexity of sequencing errors and variants in real data, the alignment sensitivity of all algorithms decreased compared with the simulated datasets. In the 51 bp dataset, the proposed method exhibited slightly lower sensitivity, likely because the short reads themselves contain limited information, and the method primarily relies on the generation of the longest common substring (LCSstr) during seed selection. Variants present in short reads can also impact the construction of LCSstr, thereby affecting the final alignment results. In the 76 bp dataset, BWA-MEM demonstrated higher sensitivity, while the proposed method showed slightly lower sensitivity but still outperformed the

other two algorithms. As read length increased, in the 102 bp and 152 bp datasets, the proposed method exhibited higher sensitivity, indicating that the method has advantages in handling real sequencing read alignment.

To further evaluate algorithm performance, experiments were conducted on the four real long-read datasets, with results presented in Table 2.

TABLE 2
ALIGNMENT SENSITIVITY FOR REAL LONG-READ SARS-COV-2 DATASETS

Comparison tools	Dataset			
	202bp	251bp	300bp	351bp
Proposed Method	**99.22%**	**99.86%**	99.90%	**99.98%**
Miinimap2	91.99%	97.71%	99.60%	98.51%
BWA-SW	91.13%	95.39%	99.50%	98.32%
BWA-MEM	98.86%	99.70%	**99.99%**	99.95%
Bowtie2	89.19%	89.30%	20.10%	58.97%

From Table 2, it can be seen that the proposed method also demonstrates high sensitivity for long-read alignment. For reads of 202 bp, 251 bp, and 351 bp, the algorithm consistently shows strong alignment performance. In the 300 bp dataset, the proposed method has slightly lower sensitivity than BWA-MEM but still significantly outperforms other alignment tools. These results indicate that the proposed method can effectively improve alignment sensitivity for real long reads, demonstrating considerable potential for practical application.

V. CONCLUSION

The algorithm achieves linear representation of SNP variants by introducing elastic-degenerate symbols, and combines minimizer indexing with a seed filtering strategy based on longest common substring intervals. This approach allows large-scale read alignment tasks to balance both sensitivity and accuracy.

Testing on simulated datasets demonstrated that the proposed method achieves high recall and precision across reads of varying lengths, with significantly lower positional errors compared to other alignment tools. This indicates that the method can more accurately map short reads back to the reference genome, thereby improving alignment accuracy. Further evaluation on real SARS-CoV-2 sequencing datasets showed that the proposed method outperforms other tools in terms of sensitivity in most datasets, with its advantage becoming particularly pronounced in long-read alignment tasks. These results validate the effectiveness of the proposed method for viral short sequence alignment and suggest that it provides a viable computational approach for large-scale genome alignment, as well as for pangenome-based alignment and viral variant analysis.

However, this work primarily focuses on aligning sequencing reads to reference genomes. Extracting biological interpretations from these differences requires close integration of biological and bioinformatics analyses to fully leverage the data. Additionally, while the dynamic programming-based seed filtering strategy enables optimal semi-global alignment, computational efficiency remains an area for improvement. Future work will focus on optimizing the algorithm's structure and implementation to enhance its generalizability and runtime performance, thereby broadening its applicability in complex genomic environments.

REFERENCES

[1] Carlos W G, Cruz C S D, Cao B, et al. Novel Wuhan (2019-nCoV) Coronavirus[J]. American Journal of Respiratory & Critical Care Medicine, 2020, 201(4).

[2] Van Dorp L, Acman M, Richard D, et al. Emergence of genomic diversity and recurrent mutations in SARS-CoV-2[J]. Infection, Genetics and Evolution, 2020, 83: 104351.

[3] Wu P, Chen D, Ding W, et al. The trans-omics landscape of COVID-19[J]. Nature Communications, 2021, 12(1): 4543.

[4] Bian Z., Liu R. Y., Ouyang H. T., et al. Investigation of non-structural protein mutations associated with non-selective mutations in SARS-CoV-2[J]. Virologica Sinica,2024,40(05):961-969.

[5] Manber U, Myers G. Suffix arrays: a new method for on-line string searches[J]. siam Journal on Computing, 1993, 22(5): 935-948.

[6] Burrows M, Wheeler D J .A Block-Sorting Lossless Data Compression Algorithm[J].technical report digital src research report, 1994.

[7] Ferragina P, Manzini G. Opportunistic data structures with applications[C]//Proceedings 41st annual symposium on foundations of computer science. IEEE, 2000: 390-398.

[8] Roberts M, Hayes W, Hunt B R, et al. Reducing storage requirements for biological sequence comparison[J]. Bioinformatics, 2004, 20(18): 3363-3369.

[9] Sirén J, Monlong J, Chang X, et al. Pangenomics enables genotyping of known structural variants in 5202 diverse genomes[J]. Science, 2021, 374(6574): abg8871.

[10] Rautiainen M, Marschall T. GraphAligner: rapid and versatile sequence-to-graph alignment[J]. Genome biology, 2020, 21(1): 253.

[11] Jain C, Rhie A, Zhang H, et al. Weighted minimizer sampling improves long read mapping[J]. Bioinformatics, 2020, 36(Supplement_1): i111-i118.

[12] Rajput J, Chandra G, Jain C. Co-linear chaining on pangenome graphs[J]. Algorithms for Molecular Biology, 2024, 19(1): 4.

[13] Li H. Minimap2: pairwise alignment for nucleotide sequences[J]. Bioinformatics, 2018, 34(18): 3094-3100.

[14] Büchler T, Olbrich J, Ohlebusch E. Efficient short read mapping to a pangenome that is represented by a graph of ED strings[J]. Bioinformatics, 2023, 39(5): btad320.

[15] Ding S. N., Wu M., Xu Y. Research on sequencing read alignment algorithms based on regional filtering[J]. Information Technology and Network Security,2018,37(04):45-48+64.

[16] Song S. Y., Cheng H. Y., Xu Y. A third-generation sequencing read alignment method based on low-frequency seeds[J]. Computer Engineering and Science,2019,41(09):1551-1556.

[17] Gao J., Xu Y. Pangenome graph sequence alignment algorithm based on combined minimizer seeds[J/OL]. Computer Engineering,1-10[2024-11-26].

[18] Gao J., Jiao Y., Zhang W. G. Review of high-throughput sequencing read alignment research[J]. Life Science Research, 2014, 18(05): 458-464.

[19] Rathod A B, Gulhane S M, Padalwar S R. A comparative study on distance measuring approches for permutation representations[C]//2016 IEEE international conference on advances in electronics, communication and computer technology (ICAECCT). IEEE, 2016: 251-255.

[20] Lcvenshtcin V I. Binary coors capable or 'correcting deletions, insertions, and reversals[C]//Soviet physics-doklady. 1966, 10(8).

[21] Zheng Z. Y., Niu X. R., Xing K., et al. From assembly to application: challenges and breakthroughs in high-quality chromosome-level pig genomes[J]. Chinese Journal of Animal Husbandry, 2025(8).

[22] Iliopoulos C S, Kundu R, Pissis S P. Efficient pattern matching in elastic-degenerate strings[J]. Information and Computation, 2021, 279: 104616.

[23] Wilbur W J, Lipman D J. Rapid similarity searches of nucleic acid and protein data banks[J]. Proceedings of the National Academy of Sciences, 1983, 80(3): 726-730.

[24] Wang K. Y., Kong S. Q., Fu Y. S., et al. Two bidirectional comparison-based longest common substring algorithms[J]. Computer Research and Development,2013,50(11):2444-2454.

[25] Skiena S S. The algorithm design manual[M]. New York: springer, 1998.

[26] Bejerano G, Pheasant M, Makunin I, et al. Ultraconserved elements in the human genome[J]. Science, 2004, 304(5675): 1321-1325.

[27] Lin H N, Hsu W L. Kart: a divide-and-conquer algorithm for NGS read alignment[J]. Bioinformatics, 2017, 33(15): 2281-2287.