Informatics in Medicine Unlocked 19 (2020) 100323

Contents lists available at ScienceDirect

Informatics in Medicine Unlocked

journal homepage: http://www.elsevier.com/locate/imu

Stretch Profile: A pruning technique to accelerate DNA sequence search

Nalakkhana Khitmoh [a,*], Sucha Smanchat [a], Sissades Tongsima [b]

[a] Faculty of Information Technology and Digital Innovation, King Mongkut 's University of Technology North Bangkok, Bangkok, Thailand
[b] National Biobank of Thailand National Center for Genetic Engineering and Biotechnology National Science and Technology Development Agency, Pathum Thani, Thailand

ARTICLE INFO

Keywords:
Sequence search
Sequence retrieval
Sequence profiling
Pruning

ABSTRACT

DNA sequence similarity search has been used by scientists to facilitate biological research. Over the years, more sequences are added to databases, making them constantly larger. Existing sequence search techniques usually focus on the improvement of sequence search algorithms to make the search faster, ignoring the possibility of reducing unrelated sequences from the search. This paper presents a pruning technique to accelerate DNA sequence search based on a novel Stretch Profile created from stretches of consecutive base characters: A-Stretch, C-Stretch, G-Stretch, and T-Stretch. The Stretch Profile is pre-generated for each sequence in a sequence database. During the search, the Stretch Profile of the query sequence is generated for comparison. The sequences in the database whose profiles do not match the Stretch Profile of the query sequence are excluded from the search, resulting in the reduction of search space, and consequently, search time.

For evaluation, we compare sequence retrievals from the Greengenes database and processing time when using only BLAST and when using the proposed pruning technique with BLAST. The results show that the proposed pruning technique can reduce the search time by 30.43% up to 63.74% depending on the length of input query, while maintaining a sensitivity of 1.00 when compared to the result of the original BLAST search.

1. Introduction

The development of bioinformatics is a result of advances in both molecular biology and computer science in the past few decades. Bioinformatics has been used to transform the research process from the traditional methods [1] to the combination of data analysis and computer simulation prior to conducting real experiments in wet lab to confirm result [2].

Researchers and scientists rely on tools and knowledge in bioinformatics to manage data resulting from various genome projects. Nucleotide and protein sequencing leads to the development of data storage systems, sequence analysis, and interpretation of genetic data [3]. Mathematics, statistics, and computer science are integrated to facilitate the understanding of gene expression control mechanisms, the relationship between different organisms, and the functions of DNA sequences [4].

DNA sequence search is one of the techniques in bioinformatics that identifies organisms that share similarity based on their biological sequences. Scientists use DNA sequence search to explain similarities between organisms that are useful in biological researches such as to identify the diversity and the functions of bacteria in an area of interest.

Many techniques have been proposed for DNA sequence search such as BLAST [5] and its extensions. However, existing techniques usually focus on the improvement of sequence search algorithms, ignoring the possibility of reducing unrelated sequences from the search. We believe that, by feeding better input, existing sequence search techniques can be accelerated. This paper thus proposes a pruning technique for DNA sequence search by eliminating unrelated sequences from the search process. A novel DNA sequence profiling based on consecutive base characters called Stretch Profile is developed to facilitate the proposed sequence pruning.

The remainder of this paper is organized as follows. Section 2 explains the existing work in DNA sequence search algorithms. Section 3 explains the proposed Stretch Profile and the pruning technique. Section 4 presents the experiment and the evaluation of the proposed work. The result is discussed in section 5 and the paper is then concluded in section 6.

* Corresponding author.
E-mail addresses: s5707011910039@email.kmutnb.ac.th (N. Khitmoh), sucha.s@it.kmutnb.ac.th (S. Smanchat), sissades@biotec.or.th (S. Tongsima).

https://doi.org/10.1016/j.imu.2020.100323
Received 17 February 2020; Received in revised form 20 March 2020; Accepted 24 March 2020
Available online 28 March 2020

2. Existing works in DNA sequence search

A genetic material or DNA, is a nucleic acid that stores the genetic information of an organism consisting of subunits called nucleotides. There are four types of nucleotides: Adenine (A), Thymine (T), Cytosine (C) and Guanine (G) arranged in a double helix shape [6]. The sequencing of the nucleotides affects the diversity and creates variances in the sequence of the DNA strands that are specific to each organism.

The efficiency of the DNA search becomes critical as the DNA database grow continuously. A sequence search usually takes a long time to find similar matching. Thus, the development of sequence search algorithms is usually based on two objectives: increasing accuracy and reducing search time. Popular sequence search techniques utilize dynamic programming and heuristics [7].

A dynamic programming technique is the process of solving complex problems by dividing them into sub-problems that can be readily solved. In solving a complex problem, it is necessary to solve sub-problems and then combine the answers of those sub-problems into the answers of the complex problem. For example, Needleman-Wunsch's algorithm employs similarity and distance measurements to find the relationships between two sequences, and keeps them as numeric values stored in the form of a matrix [8]. The Smith-Waterman algorithm also uses dynamic programming to perform local alignment to find the pattern of two sequences, to align the proteins or nucleotides using the matrix scoring system [9]. Although dynamic programming is able to give an accurate result, it is slow and requires high computational power.

An alternative approach to dynamic programming is heuristics, which is faster but may not guarantee optimal results. One of the most popular DNA sequence searches using heuristics is the BLAST (basic local alignment search tool) algorithm [5]. BLAST fragments the query sequence into words of size 11 for DNA and 3 for protein. For an input word, neighboring words are generated. Then, the scoring function is used to measure the similarity of each neighboring word in comparison to the input word. Those words with the scores higher than a threshold are collected. BLAST then scans the database for the matching of the collected words. Once a match is found, the word is repeatedly extended on both sides until the score falls below the threshold. The similarity score is then calculated; a higher similarity score means that the sequence in question is more similar to the query sequence [10].

BLAT (BLAST-like alignment tool) [11] is a tool that uses sequence search for sequence alignment. Indexes of sequences are constructed and kept in memory, allowing for faster processing. The sequences of nucleotides or sequences of proteins must be perfectly matched or nearly identical to be retrieved. Thus, BLAT is less flexible than BLAST.

The search techniques proposed in later years incorporate indexing to improve search time. DSIM (Distance-based Sequence Indexing Method) [12] was developed to reduce the memory space through indexing and compression based on a video compression technique. Sequences in the database are indexed and compressed. The query sequence is first searched through the index based on a similarity distance between the sequences in the database and the query sequence. The index is updated when new sequences are added. This method reduces the processing time to search the entire sequence.

In addition to index compression techniques to reduce memory space, the Hamming distance has been adopted in Hfwd^2 [13] to calculate the similarity between sequences. In the first step, the DNA sequence is divided into windows of equal length. Subsequences of a query sequence are searched in the database, which is indexed using R-tree, based on Hamming distance. Thus, the similarity comparison of DNA sequences is faster and processes less data [14].

Many other indexing techniques have been proposed under different names. For example, suffix array [15] and suffix tree [16,17] find subsequences called seeds in a DNA sequence database that are similar to query sequence, and keep them as index for further search to reduce database access. The qCluster technique [18] constructs clusters of similar subsequences call q-Gram to serve as index for the search.

From the related work discussed in this paper, most of the existing techniques usually focus on improving the search algorithm. Although indexing has been used to improve the efficiency of sequence search algorithms, it may require large storage memory [16]. The proposed Stretch Profile in this paper, on the other hand, focuses on manipulating the search input by profiling DNA sequences based on consecutive base characters.

3. The proposed pruning technique for DNA sequence search

The main concept of the proposed pruning technique is to reduce the number of sequences to be searched by pruning or eliminating the sequences that should not satisfy the similar matching, improving the retrieval speed while maintaining accuracy.

3.1. Stretch Profile

For the pruning, we introduce a novel DNA sequence profiling to generate profiles of the input query sequence and the sequences in the database. We take to advantage that a DNA sequence is a combination of only four base characters (A, C, G, and T) to create a sequence profile. The proposed Stretch Profile is based on the observation that there are consecutive occurrences of each base character in a sequence. If the input query sequence has a certain length or stretch of consecutive base characters, then only the sequences in the database with equal or longer length of consecutive base characters should be searched.

To create the proposed Stretch Profile, we use the longest stretch of consecutive occurrence of each base character. Thus, the Stretch Profile of each sequence is defined by four numbers, namely, A-Stretch, C-Stretch, G-Stretch, and T-Stretch. For example, if a sequence contains the longest consecutive of "GGGGGGGG", the G-Stretch of this sequence is 8.

However, when searching for similar matching, it is necessary to accommodate possible mutation in the sequences. In other words, the search needs to include the sequences that are slightly different from the input query sequence. Thus, the longest stretch of consecutive occurrences must also consider a similarity rate. In this work, we consider three similarity rates of >60%, >70%, and >80%. For example, a subsequence "CCCCTCACCGCC" would satisfy the similarity rate of >70% of "C", thus the C-Stretch would be 12.

To find the longest stretch of consecutive occurrences with a similarity rate, we develop the Growing Stretch Window algorithm as shown in Fig. 1. The left pointer starts at the first base character of the sequence. The right pointer starts at the second character from the left pointer to form the initial window with the size of 3 characters (lines 4–5). If the current window satisfies the similarity rate (line 9) then the largest window size is saved (lines 10–11). Then the right pointer moves to the next character growing the window size (line 13).

With the initial window size of 3 characters, it is very likely that the initial similarity rate would not be satisfied in the case of one odd character. For example, "CCA" would have the similarity rate of 66% for C-Stretch, which is always below the >70% similarity setting. Thus, we introduce a fail chance (line 3). This fail chance allows the window to grow (without updating the stretch size) with the hope that the next character would make the window satisfy the similarity rate (lines 14–16). This fail chance is restored if the window once again satisfies the similarity rate (line 12).

With the fail chance, the window stops growing when it cannot satisfy the similarity rate in two consecutive characters. When this happens, the window is reset by shifting the left pointer to the next character and setting the right pointer to the second character from the left pointer (lines 17–19). The fail chance is also restored for the new window (line 20). The whole process is repeated until the left pointer reaches the end of sequence. The longest stretch of the character in question is then returned. An example of finding an A-Stretch using the Growing Stretch Window algorithm with the similarity rate of >70% is

```
Algorithm: Growing Stretch Window
Input:  sequence – a DNA sequence
        character – the base character to find stretch
        similarity – the predefined similarity rate
Output: stretch of character
 1  EOS ← length(sequence)
 2  stretch ← 0
 3  chance ← true
 4  left ← 0
 5  right ← left + 2
 6  while (true) do
 7    occurrences ← count(sequence[left:right]), character)
 8    size ← length(sequence[left:right])
 9    if (occurrences / size > similarity)
10      if (stretch < size)      #update stretch size
11        stretch ← size
12      chance ← true    #restore chance
13      right ← right + 1        #grow window
14    else if (chance)  #if fail chance available
15      chance ← false
16      right ← right + 1
17    else                       #shift window and restore chance
18      left ← left + 1
19      right ← left + 2
20      chance ← true
21    if (left ≥ (EOS - 2)) or (right ≥ EOS)
22      break
23  return stretch
```

Fig. 1. Growing stretch window.

depicted in Fig. 2.

The Growing Stretch Window algorithm may not guarantee that the obtained stretch of consecutive occurrences is actually the longest in the sequence because it cannot look far ahead in the sequence. For example, a subsequence "GGTCGGGGGG" should satisfy the similarity rate of >70% for G-Stretch. However, the algorithm would encounter "C" at the fourth characters (with the preceding "T" consuming the fail chance). Thus, the window size is reset because the similarity rate at that point is below 70% before the right pointer reaches the further "G"s ahead. In this case, the G-Stretch would be 7 instead of 10. Despite this flaw, the Growing Stretch Window algorithm is simple and fast within the $O(n^2)$ complexity, where n is the size of the sequence. Our preliminary test shows that this algorithm is sufficient for generating the Stretch Profile.

We have considered the other options to find the longest stretch by repeatedly scanning the whole sequence with fixed-size windows. However, it is impossible to find the correct window size unless every window size from one to the sequence size is tried. This would significantly increase the processing time, which is not favorable.

Using the longest stretch of consecutive occurrences allows the Stretch Profile to be created faster than a more costly approach of mining for specific patterns in each sequence that is usually used in existing indexing approaches. In addition, the Stretch Profile is generic so that it can be applied to any DNA sequence. The only drawback is that the Stretch Profile must first be created for every sequence in the database. However, this overhead occurs only once for each sequence in the database with each similarity rate.

3.2. Sequence pruning

Sequence pruning is the process that prunes sequences in the database by comparing the profiles of the sequences in the database and the profile of the input query sequence for similar matching.

Once the Stretch Profile is created for every sequence in the database, the sequences are then sorted in non-descending order for comparison during the search. Because the Stretch Profile is composed of four numbers for the four stretches, the sequences are sorted first based on A-Stretch followed by T-Stretch, C-Stretch, and then G-Stretch. Note that the stretches can be in any order; the efficiency would depend on the values of the four stretches across the whole database. However, with binary search employed for comparison, the effect of different stretch orders would be kept minimal.

When scientists need to search the database for the sequences that are similar to a query sequence, the Stretch Profile of the query sequence is first created using the Growing Stretch Window algorithm. The Stretch Profile of the query sequence is then compared to the sorted profiles in the database using binary search. The sequences in the database with the Stretch Profile with all A-Stretch, T-Stretch, C-Stretch, and G-Stretch greater than or equal to those of the query sequence will be marked for the search process, while other sequences are pruned. With binary search, the complexity for comparing Stretch Profiles is kept minimal within $O(log_2 n)$, where n is the number of sequences in the database. The output of the pruning is then fed to existing search algorithms such as BLAST. The process of the proposed pruning technique is shown in Fig. 3.

With such pruning, fewer sequences will become input for the search process, thus reducing the time required to complete the search. It is possible in the worst case that no sequence is pruned because the stretches of the query sequence are very small. In such a case, the search performance will be reduced to the normal search process without pruning.

4. Evaluation and results

We evaluate the proposed Stretch Profile and the pruning technique in three aspects: sensitivity, reduction of input sequences, and search time. BLAST is employed as the core search algorithm for all evaluations. We also use the output of the original BLAST search as the baseline for all the experiments.

Sensitivity indicates the performance of the proposed technique at finding the sequences that are retrieved by BLAST. For a good result, the sensitivity should be high. Note that it is not possible to measure the specificity and the precision of the proposed pruning technique. With

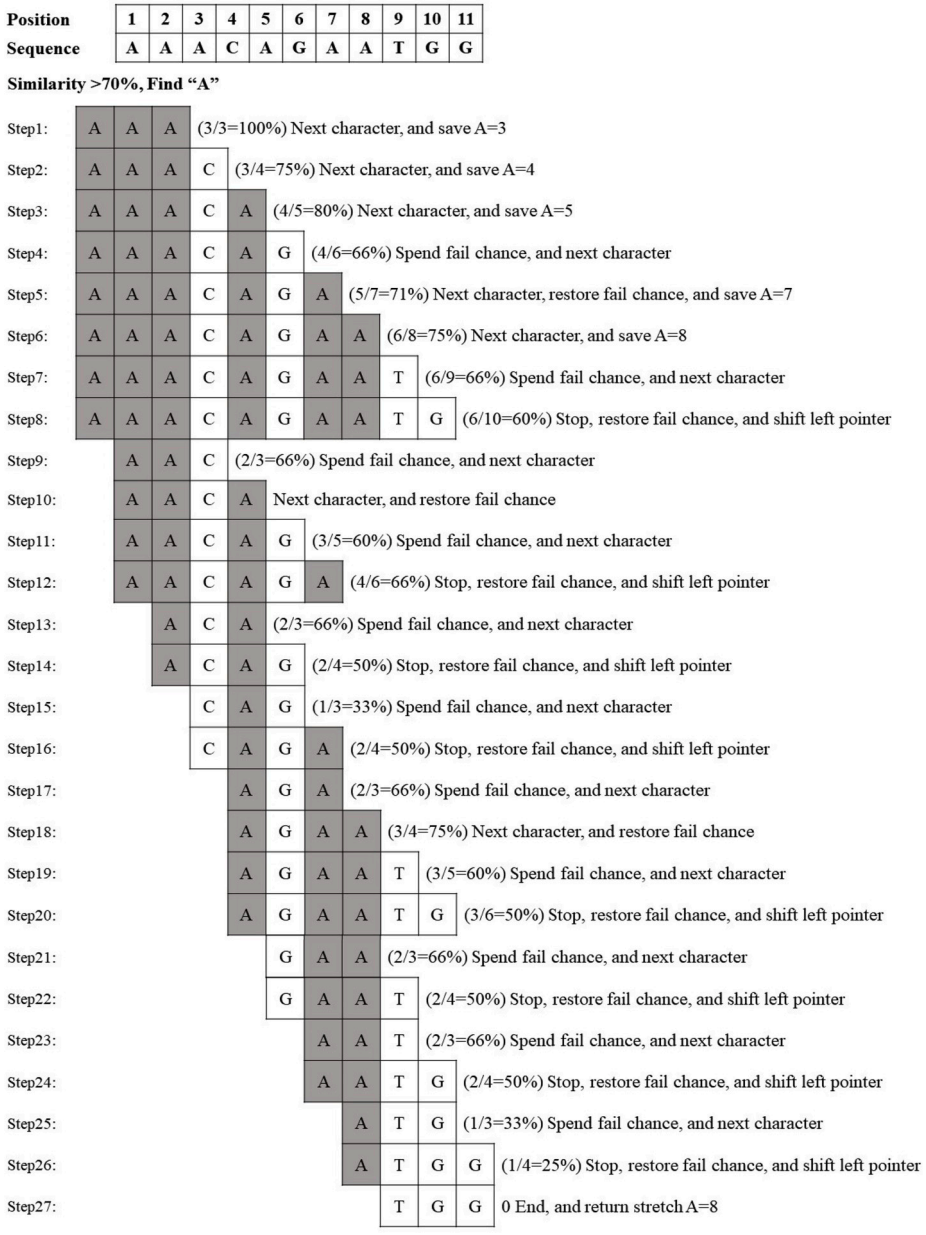

Fig. 2. Example of generating Stretch Profile.

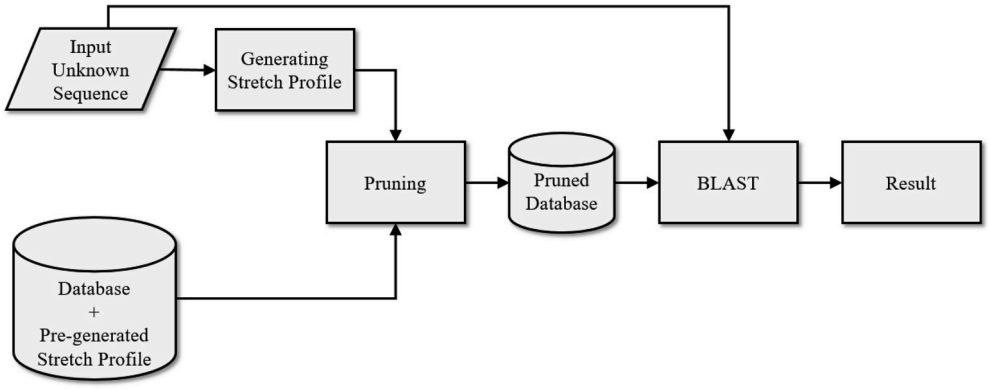

Fig. 3. The process of the proposed pruning technique.

the original BLAST output as the baseline, the false positive and the true negative of the evaluation are not meaningful.

4.1. Dataset and experiment setting

For the experiment, we use the Greengenes database (http://greengenes.lbl.gov) under the management of the University of Colorado and the University of Queensland, which contains 16S rRNA genes of bacteria. The database contains 1,075,170 sequences. The lengths of the shortest and the longest sequences are 1253 and 2368 characters, respectively, with the average length of 1396 characters.

We also obtained an unknown bacteria sequence dataset from the Genome Institute, National Center for Genetic Engineering and Biotechnology, Thailand. The dataset is stored in FASTA format containing 410,551 sequences. The lengths of the shortest and the longest sequences are 365 and 456 characters, respectively, with the average length of 412 characters. For simplicity, we further refer to this dataset as the query dataset.

All the experiments are carried out on a desktop computer with Intel Core i7 3.4 GHz with 32 GB RAM. The proposed technique is implemented in Python3. The BLAST software [19] is employed for sequence search using 4 processor cores.

The evaluation is divided into two parts. The first part evaluates the effect of the similarity rate on the search. The second part evaluates the effect of the length of the query sequence. The evaluation process of the three parts is depicted in Fig. 4.

4.2. Evaluation of similarity rate

This section evaluates the effect of different similarity rates of the Stretch Profile on the search. The similarity rates considered are >60%, >70%, and >80% as mentioned earlier. The evaluation consists of five processes including selecting input query sequences, generating Stretch Profile, pruning the sequences, sequence search, and evaluating the results.

For the input query, 100 sequences are randomly selected from the query dataset to search for similar matching in the Greengenes database. The Growing Stretch Window algorithm is then applied to the 100 input sequences and the entire Greengenes database to generate the Stretch Profiles. Each sequence (both in the input and in the Greengenes database) is associated with three Stretch Profiles (one for each similarity rate).

Table 1 list the overhead times required to generate the Stretch Profiles for all the sequences in the Greengenes database at the three similarity rates. As mentioned earlier, this overhead occurs only once.

Table 1
Overhead time for generating Stretch Profiles of Greengenes database.

Similarity Rate	Time (s)	Average time per sequence (s)
>60%	40,138.61 (over 11 h)	0.0373
>70%	34,571.42 (over 9 h)	0.0322
>80%	35,094.01 (over 9 h)	0.0326

The improvement of search time would cover this overhead in the long term.

During the pruning, the profiles of the input sequences are compared to the profiles of the sequences in the Greengenes database with the same similarity rate. Binary search is employed in the comparison. After pruning, only the sequences that are marked for search will be considered as the input for sequence search. For simplicity, we refer to these marked sequences as the pruned database as shown in Fig. 4.

Two searches are then performed for comparison: original search and the proposed pruned search, both employing BLAST software as the search algorithm. The output of each search is a text file containing the identity numbers, the names, and the percentage of similarity to the input sequence. For the evaluation, we are only interested in the names of the retrieved output sequences. We consider a search correct when the top five bacteria names in the output of the pruned search are similar to those found in the output of the original BLAST search. The results of this experiment is shown in Table 2. The second and third columns show the mode values of the stretches in the sequences (not the mode of the profiles).

From the experiment result in Table 2, at a similarity rate of >60%, the proposed pruned search reduces the average search time by almost half (1.91 s per sequence against 3.31 s per sequence, approximately 42.30%). This is due to the reduction of the number of sequences to be searched (599,017 remaining sequences against 1,075,170 total). However, the sensitivity, although high, indicates a deviation from the result of the original BLAST search. The similarity rate of >60% allows the stretches of the input sequence profile to be longer, with the modes of (8, 12, 9, 8), when compared to those of the other similarity rates. The longer stretches result in more sequences being pruned from the database. However, it also prunes the sequences that should be in the search resulting in the reduced sensitivity.

At the similarity rate of >80%, the proposed pruning technique provides the search result similar to those of the original BLAST search with the sensitivity of 1.00. The average search time improves by 51.96% (1.59 s per sequence against 3.31 s per sequence) due to the shorter stretches, with the modes of (5, 8, 5, 4), and thus the smaller reduction of the number of sequences to be searched (459,327 remaining sequences against 1,075,170 sequences total).

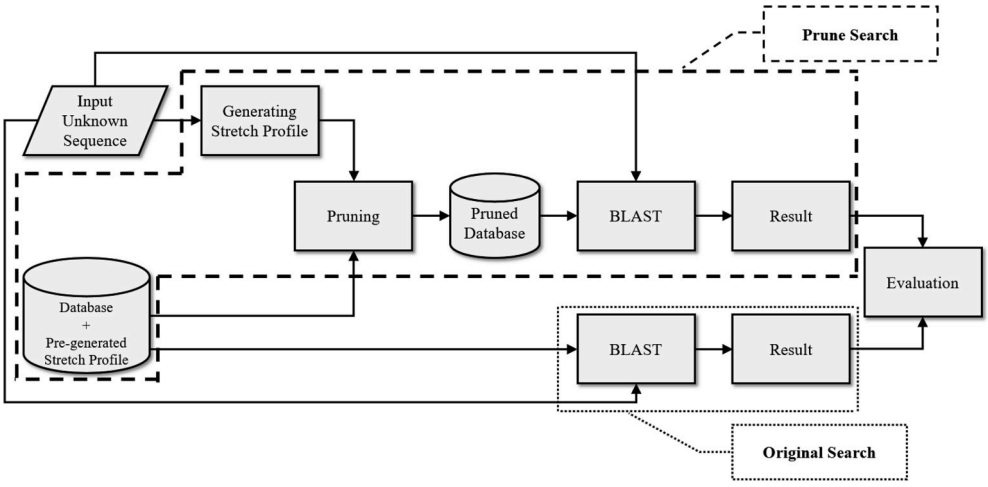

Fig. 4. Evaluation process.

Table 2
Result of similarity rate experiment.

Similarity rate	Mode of stretches of query sequences (A,T,C,G)	Mode of stretches of Greengenes database (A,T,C,G)	Sensitivity	Average number of sequences after pruning	Average BLAST search time (s/seq)		
					Original	Pruned	Time reduction
>60%	(8, 12, 9, 8)	(11, 12, 11, 9)	0.97	599,017	3.31	1.91	42.30%
>70%	(6, 11, 7, 4)	(7, 8, 7, 7)	1.00	297,017	3.31	0.98	70.39%
>80%	(5, 8, 5, 4)	(6, 7, 7, 6)	1.00	459,327	3.31	1.59	51.96%

The similarity rate of >70% also provides the similar search result to those of the original BLAST search with the sensitivity of 1.00. In addition, the improvement of the average search time is the highest among the three settings (0.98 against 3.31, approximately 70.39%) due to the largest reduction of the number of sequences to be searched (297,017 remaining sequences against 1,075,170 total).

4.3. Evaluation of query sequence length

The experiment in this section aims to evaluate the effect of the different lengths of the input query sequences on the proposed pruning technique. The evaluation process is similar to the evaluation of the similarity rates with the addition of different input query sizes. Two similarity rates, >70% and >80%, are used in this experiment as they yielded a sensitivity of 1.00 in the previous experiment.

Seven different sequence lengths of 500, 400, 300, 200, 150, 100, and 50 characters are evaluated to reflect the length of the query dataset in this experiment (ranging from 365 to 456 characters) and our actual use cases. To construct the input query sequences of these lengths, we first randomly select 10 sequences of different bacteria from the Greengenes database. Each of the 10 sequence is randomly cut into the 10 subsequences of each lengths. Thus, each sequence will generate 70 subsequences, totaling 700 subsequences for all 10 bacteria sequences.

The Growing Stretch Window is then applied to generate the Stretch Profiles of the 700 subsequences. The Stretch Profiles of the Greengenes database have already been created from the previous experiment and can be reused.

The pruning process is then applied resulting in the pruned database, similar to the previous experiment. The sequence search using BLAST is then performed on both the pruned database and the entire Greengenes database for evaluation.

Similar to the previous experiment, we are only interested in the names of the retrieved output sequences. We consider a search correct when the top five bacteria names in the output of the pruned search are similar to those found in the output of the original BLAST search. The results of this experiment are shown in Table 3 and Table 4 for the similarity rates of >70% and >80%, respectively. The second column shows the mode values of the stretches in the sequences (not the mode of the profiles).

For the similarity rate of >70%, the sensitivity is 1.00 for every length of input subsequence except for the query of 500 characters. The percentage of time reduction is lower for shorter lengths as depicted in Fig. 5. This is due to the reduction of the sequences to be searched. A longer query length contributes to longer stretches (as seen from the mode values of the stretches), and thus more sequences are pruned from the search. However, longer stretches may also prune the correct sequences from the search as observed in the query length of 500 characters with the sensitivity of 0.94; this is the same effect observed in the case of a lower similarity rate discussed in the previous section. Considering the sensitivity of 1.00, the proposed pruning technique can reduce the search time from 19.58% up to 66.83%. In other words, the speed is approximately three times faster in the best case.

For the similarity rate of >80%, the percentage of time reduction is also lower for shorter lengths as depicted in Fig. 6. However, the proposed pruning technique can maintain the sensitivity of 1.00 in all lengths and can reduce the search time from 30.43% up to 63.74%. In other words, the speed is also approximately three times faster in the best case.

It is noticeable that the time reduction is not in line with the trend when the input length is 150 characters with the similarity rates of both >70% and >80%. We attribute this to the random selection when constructing the input sequences for the experiment.

This experiment shows that, with the similarity rate of >80%, the proposed pruning technique based on the Stretch Profile can be used to improve the processing time for DNA sequence search while maintaining 1.00 sensitivity when compare to the original BLAST search.

5. Discussion

While existing techniques usually try to improve the DNA sequence search algorithm [11–13], the proposed pruning technique instead focuses on the sequences to be searched by eliminating the sequences that should not satisfy the similar matching so that the search process is accelerated. Unlike existing indexing approaches that need to find similar patterns to make the search faster [20] and may require large storage memory [14,16], our Stretch Profile is simple and generic by merely scanning for consecutive occurrences of the four base characters. The complexity of the Growing Stretch Window algorithm for generating the Stretch Profile of a sequence is $O(n^2)$ where n is the length of the sequence. In addition, being composed of only four numbers, the storage memory required for the Stretch Profile is very small and negligible. As the proposed pruning technique utilizes existing search algorithms such as BLAST, the complexity of the search itself is similar to the search algorithm employed. However, the search space is reduced by the pruning, which has the complexity of $O(log_2 n)$ where n is the number of sequences in the database as the binary search is employed.

Table 3
Result of query length experiment at similarity rate >70%.

Length of query	Mode of stretches of query sequences (A,T,C, G)	Sensitivity	Average number of sequences after pruning (1,075,170 total)	Average BLAST search time (s/seq)		
				Original	Pruned	Time reduction
500	(6, 11, 7, 7)	0.94	224,735	3.81	1.14	70.08%
400	(6, 11, 7, 4)	1.00	297,869	2.98	0.99	66.83%
300	(6, 11, 7, 4)	1.00	422,931	2.39	1.05	56.16%
200	(6, 7, 7, 4)	1.00	579,640	1.80	1.07	40.47%
150	(4, 7, 7, 4)	1.00	831,894	1.89	1.52	19.58%
100	(4, 6, 4, 4)	1.00	764,748	1.54	1.07	30.51%
50	(4, 4, 4, 4)	1.00	885,713	1.22	0.86	29.51%

Table 4
Result of query length experiment at similarity rate >80%.

Length of query	Mode of stretches of query sequences (A,T,C, G)	Sensitivity	Average number of sequences after pruning (1,075,170 total)	Average BLAST search time (s/seq)		
				Original	Pruned	Time reduction
500	(5, 8, 5, 5)	1.00	354,186	4.55	1.65	63.74%
400	(5, 8, 5, 5)	1.00	447,618	4.83	2.14	55.69%
300	(5, 8, 5, 5)	1.00	520,315	4.12	2.14	48.06%
200	(5, 6, 5, 4)	1.00	680,423	2.65	1.62	38.87%
150	(5, 6, 5, 4)	1.00	789,446	1.84	1.28	30.43%
100	(4, 5, 5, 4)	1.00	838,677	2.13	1.28	39.91%
50	(4, 4, 4, 4)	1.00	1,018,292	2.09	1.42	32.06%

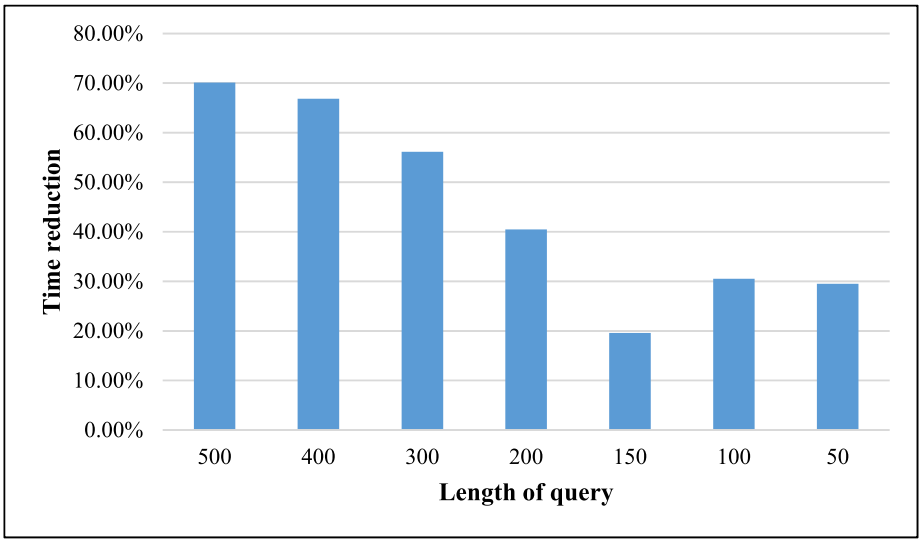

Fig. 5. Comparison of time reduction at similarity rate >70%.

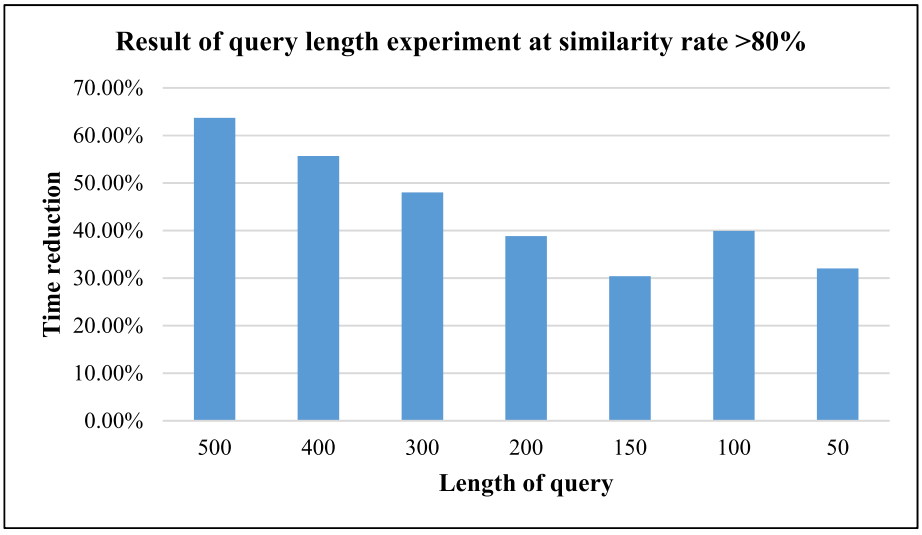

Fig. 6. Comparison of time reduction at similarity rate >80%.

Our technique neither intervenes nor modifies the search algorithm (in this case, BLAST). It merely preprocesses the database and the query sequence before the search. Thus, the proposed pruning technique is applicable to any existing DNA sequence search algorithm. The only disadvantage is that the Stretch Profile must be generated for the sequences in the database and any new sequence being added to the database, which are then sorted. However, this overhead occurs only once and would be compensated by the reduction of search time in the long term.

Two factors affect the performance of the proposed technique: the similarity rate for generating the Stretch Profile and the length of the input query sequence. As per the experiment, the similarity rate should be set to >80% resulting in the sensitivity of 1.00 when compared to the result of BLAST without using our technique. The search time can be

improved by 30.43% up to 63.74% depending on the input length.

As for the length of the input query sequence, shorter lengths result in less sequences being pruned, leading to a lower reduction of search time. However, our current work is limited by the use of the Greengenes database. The effect of the query length may be specific to different datasets. The proportion of the query length to the length of the sequences in the database may also affect the sensitivity. Nevertheless, manipulating the length of the input query sequences may not be desirable in real use. Thus, our experiments serve to identify that the proposed pruning technique can be used to improve DNA sequence search time. For sensitive applications using other databases that require extremely accurate results, the proposed technique can be used for an initial result that is to be confirmed with an original search if necessary.

6. Conclusion and future work

In this paper, we propose a novel technique to accelerate DNA sequence search using a pruning approach. The pruning is based on the proposed Stretch Profile that is constructed from the length of consecutive occurrences of each base character. By comparison, the sequences in the database with the Stretch Profiles shorter than that of the input query sequence will be pruned from the search thus reducing the search size. The pruned database can then be searched using an existing algorithm such as BLAST. According to the experiments, the proposed technique can improve the BLAST search time by 30.43% up to 63.74% while giving a similar retrieval output as the original BLAST without pruning.

In future work, we plan to utilize the Stretch Profile to divide a sequence database into smaller partitions and distribute them to several computer nodes. This will facilitate a parallel and distributed search, because only the nodes with the matching profiles need to be searched. In addition, different DNA sequence datasets will be tested to further study the performance, the effect of input query length, and the effect of the similarity rate.

Ethical statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

None.

Acknowledgement

The authors would like to express gratitude to Alisa Wilantho for her help in providing the data in the experiments.

References

[1] Pareek CS, Smoczynski R, Tretyn A. Sequencing technologies and genome sequencing (in eng) J Appl Genet 2011;52(4):413–35.
[2] Horner DS, et al. Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing (in eng) Briefings Bioinf Mar 2010;11 (2):181–97.
[3] Lee HC, Lai K, Lorenc MT, Imelfort M, Duran C, Edwards D. Bioinformatics tools and databases for analysis of next-generation sequence data (in eng) Brief Funct Genom Jan 2012;11(1):12–24.
[4] Milicchio F, Rose R, Bian J, Min J, Prosperi M. Visual programming for next-generation sequencing data analytics (in eng) BioData Min 2016;9. 16-16.
[5] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol 1990/10/05/1990;215(3):403–10.
[6] Pal SK, Bandyopadhyay S, Ray SS. Evolutionary computation in bioinformatics: a review. Trans Syst Man Cyber Part C 2006;36(5):601–15.
[7] Altschul SF, Boguski MS, Gish W, Wootton JC. Issues in searching molecular sequence databases. 1994/02/01 Nat Genet 1994;6(2):119–29.
[8] Pearson WR, Lipman DJ. Improved tools for biological sequence comparison. April 1, 1988 Proc Natl Acad Sci Unit States Am 1988;85(8):2444–8.
[9] Smith TF, Waterman MS. Identification of common molecular subsequences. 1981/03/25/ J Mol Biol 1981;147(1):195–7.
[10] Brooks LD, Weir BS, Schaffer HE. The probabilities of similarities in DNA sequence comparisons. 1988/10/01/ Genomics 1988;3(3):207–16.
[11] Kent WJ. BLAT–the BLAST-like alignment tool (in eng) Genome Res 2002;12(4): 656–64.
[12] Xia C, Beng Chin O, Tung AKH, Hwee Hwa P, Kian L. DSIM: a distance-based indexing method for genomic sequences. In: Fifth IEEE symposium on bioinformatics and bioengineering (BIBE'05); 2005. p. 97–104.
[13] Jeong I-S, Park K-W, Kang S-H, Lim H-S. An efficient similarity search based on indexing in large DNA databases. 2010/04/01/ Comput Biol Chem 2010;34(2): 131–6.
[14] Tan Z, Cao X, Ooi BC, Tung AKH. The ed-tree: an index for large DNA sequence databases. In: Presented at the proceedings of the 15th international conference on scientific and statistical database management, Cambridge, MA; 2003. https://doi.org/10.1109/SSDM.2003.1214976. Available.
[15] Suzuki S, Kakuta M, Ishida T, Akiyama Y. GHOSTX: an improved sequence homology search algorithm using a query suffix array and a database suffix array. PLoS One 2014;9(8):e103833.
[16] Hunt E, Atkinson MP, Irving RW. A database index to large biological sequences. In: Presented at the proceedings of the 27th international conference on very large data bases; 2001.
[17] Barsky M, Stege U, Thomo A, Upton C. A new method for indexing genomes using on-disk suffix trees. In: Presented at the Proceedings of the 17th ACM conference on Information and knowledge management, Napa Valley, California, USA; 2008. https://doi.org/10.1145/1458082.1458170. Available.
[18] Cao X, Li SC, Tung AKH. Indexing DNA sequences using q-grams. In: Zhou L, Ooi BC, Meng X, editors. Database systems for advanced applications: 10th international conference, DASFAA 2005, beijing, China, April 17-20, 2005. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg; 2005. p. 4–16.
[19] Schloss DP. Mothur. 7/2/2020. 2008-2019. Available, https://www.mothur.org/.
[20] Califano A, Rigoutsos I. FLASH: a fast look-up algorithm for string homology. In: Proceedings of the first international conference on intelligent systems for molecular biology (ISMB '93); 1993.