

# A Minimum Cost Process in Searching for a Set of Similar DNA Sequences

M YAZID M SAMAN<sup>a</sup>, M NORDIN A RAHMAN<sup>b</sup>, AZIZ AHMAD<sup>c</sup> AND A OSMAN M TAP<sup>d</sup>

<sup>a</sup>Computer Science Department, Kolej Universiti Sains dan Teknologi Malaysia  
21030 K Terengganu, MALAYSIA

<sup>c</sup>Biology Science Department, Kolej Universiti Sains dan Teknologi Malaysia  
21030 K Terengganu, MALAYSIA

<sup>d</sup>Mathematics Department, Kolej Universiti Sains dan Teknologi Malaysia  
21030 K Terengganu, MALAYSIA

<sup>b</sup>Information Technology Center, University of Darul Iman, KUSZA Campus  
21300 K Terengganu, MALAYSIA

*Abstract:* - DNA sequence alignment for similarity search is a vital topic in bioinformatics algorithm development. Computational searching for a set of DNA sequences,  $S$ , that similar to a query sequence,  $q$ , in a large scale of DNA databases is very complicated and requires high processors performance as well as large memory spaces. Frequently, quadratic running time complexity dynamic programming algorithms used to produce a local optimal sequence alignment. However, this algorithm is time consuming in dealing with a long DNA sequences. By means of local alignment, this paper presents a framework to search a set of similar sequences in a large scale of DNA databases with reliable output and minimum cost. The Knuth-Morris-Pratt algorithm (KMP) is adapted and acts as a *filtering* mechanism before exhaustive dynamic programming is applied. The KMP algorithm is used to scan the generated patterns from query sequence to the sequences in databases. This filtering process generates scores which are used for ranking purposes. The Smith-Waterman algorithm then is applied to each sequences starting from the top of the constructed ranking. The paper also discusses the optimal patterns length that highly appropriate for the database scanning process. The experiment results show that the filtering mechanism proposes discard irrelevant sequences. Therefore, the time for searching and retrieving the set of similar sequences from databases to the query is minimized.

*Keywords:* - *sequence alignment, exact string matching, dynamic programming, Knuth-Morris-Pratt algorithm, Smith-Waterman algorithm*

## 1 Introduction

Comparison of genome sequences (DNA, mRNA and protein) are among the most important task done by researchers in the area computational biology. Comparison, pattern recognition, detecting similarity and phylogenetic trees constructing in genome sequences are the most popular tasks defined by [5]. All of these task aspects are done by getting the result from sequence alignment processes. The process of sequence alignment allows the insertion, deletion and replacements of symbols that representing the nucleotides or amino acids sequences. From the biological point of view sequence comparison is motivated by the fact that all living organisms are related by evolution. That implies that the genes of

species that are closer to each other should show signs of similarities at the DNA level. Moreover, those similarities also extend to gene function.

The DNA alphabet consists of the four nucleotides a, c, g and t (standing for adenine, cytosine, guanine, and thymine, respectively) used to encode DNA, and could be signed as  $\Sigma = \{a, c, g, t\}$ . Normally, when a new DNA or protein sequence is determined, it would be compared to all known sequences in the annotated databases such as GenBank, SwissProt and EMBL. Blast [1], FASTA [15] and PatternHunter [11][12] tools are three rapid heuristic algorithms are regularly used for searching protein and DNA sequence databases. The idea in these tools is to find subsequences which share some patterns called as filtration

techniques. The weakness of these techniques give approximate results and there is a possibility of missing an alignment or giving inaccurate output.

Relatively, it is reasonable to align two sequences by using classical Smith-Waterman dynamic programming algorithms [17] when the sequences length is not very long. However by using this approach, searching and comparing a query sequence with the databases with large size of sequences is complicated and requires for more time and spaces complexity. Therefore, the need of mechanism to discard the unrelated or irrelevant sequences compared to a query is highly demanded.

In this paper, we present a new search method for DNA sequence matching in a large size of DNA sequence databases. The Knuth-Morris-Pratt algorithm (KMP) [9] is used to scan the generated patterns from query DNA to every sequence database and a matching score would be assigned to each result. In greedily, a set of highest targeted sequences matching score result is aligned using exhaustive Smith-Waterman dynamic programming algorithm.

This paper is organized as follows. Section 1.1 discusses briefly the general concepts of exact string matching algorithms especially KMP algorithm. Section 1.2 emphasizes on dynamic programming algorithms idea and its application in DNA sequence alignment. Idea on seeking minimum cost in retrieving a set of similar DNA sequences from databases to a query DNA sequence is sketched in Section 2. Some experiments result is given in Section 3. Section 4 concludes the proposed framework and future research subjects are stated.

### 1.1 KMP Exact Pattern Matching Algorithms

Pattern matching is a vital component of many domain problems, including text editing, information retrieval, signal processing and recently in bioinformatics applications. Generally, exact pattern matching problem on string consists of finding all occurrences (or the first occurrence) of a pattern,  $p$  of length  $m$ , in a text,  $t$  of length  $n$ , where the  $p$  and the  $t$  are strings over some alphabet,  $\Sigma$ . To date, there are many algorithms for exact pattern matching developed such as Brute-Force (BF), Knuth-Morris-Pratt (KMP) and Boyer-Moore (BM) [10]. The reliability of these exact string matching algorithms constantly depends on the ability to

detect the presence of match characters and the ability to discard of any mismatch characters.

The  $O(m \times n)$  running time BF pattern matching algorithm (or naive algorithm) consists of two nested loops. The external loop used to index all possible starting indices of the pattern in the text. Whilst, the inner loop performs the indexing for each character of the pattern, and then comparing it to it's potentially relating character in the text.

The BM pattern matching algorithm is developed to improve the running time of BF algorithm. BM algorithms have worst-case-linear-search behavior, improving the quadratic BF algorithm. The BM algorithm positions the pattern over the leftmost characters in the text and attempts to match it from right to left. If no mismatch occurs, the pattern has been found. Otherwise, the algorithm computes a shift (character jump); that is, an amount by which the pattern is moved to the right before a new matching attempt is undertaken.

The KMP algorithm is the first algorithm for which the constant factor in the linear term, in the worst case, does not depend on the length of the pattern [9]. The main initiative of KMP algorithm is to preprocess the pattern in time  $O(m)$  with the expected number of comparisons,  $\bar{C}_n$ , performed is bounded by  $n + O(1) \leq \bar{C}_n \leq 2n + O(1)$ . The worst case running time of KMP algorithm is  $O(m+n)$ . Generally, the algorithm may be used for filtering of potential matches or for searching retrieval terms that will be highlighted in the output. The central inspiration of this algorithm is to avoid the repeating comparisons with the known characters. Therefore, within the processing, each time a mismatch is detected, the "false start" consists of characters that already examined [9][16]. In addition, it is always possible to arrange the algorithm so that the pointer in the text is never decremented. To accomplish this, the pattern is preprocessed to obtain a table that gives the next position in the pattern to be processed after a mismatch [16].

### 1.2 Dynamic Programming in Local Sequence Alignment

The alignment process of DNA sequence data has a long history which has been studied by many biologist and computer scientist researchers. An alignment process produces a maximal level identity (similarity) between two or more

sequences. Two or more sequences are *homologous* if they share a common ancestor, which is not always easy to determine [5].

Frequently, DNA sequence alignment (also can be extended to multiple sequence alignment) is solved by a using dynamic programming algorithm running in quadratic time complexity,  $O(n \times m)$ , where  $n$  is length of first sequence and  $m$  is length of second sequence. The inspiration of this algorithm is to build up the result by using previous results for smaller subsequences. The dynamic programming algorithm applies a two dimensional array called *similarity matrix*, denoted as  $F(i, j)$ . Those cells in similarity matrix used to store the calculated value, corresponding to partial results, in a data structures and reuses them as the new value calculated. This method is computational expansive should the alignment process is apply to the whole genome sequences databases.

DNA sequence similarity could be evaluated globally or locally depending on the needs of biologist requirements. Global similarity requires reflection of the total lengths of the DNA sequences under study while local similarity might consider only portions of DNA sequences that have the best matches. The first algorithm for global similarity search is Needleman-Wunsch algorithm [13] and the most commonly used today is local similarity search Smith-Waterman algorithm [17]. Both of these algorithms apply the techniques of dynamic programming.

The dynamic programming in Smith-Waterman algorithm checks every possible alignment between two given sequences. The two sequences define a matrix in which every cell corresponds to the alignment of two specific positions in compared sequences. The first step, scores in the first row,  $F(0, 0), F(0, 1) \dots F(0, n)$ , and column,  $F(0, 0), F(1, 0) \dots F(m, 0)$  are initialized to zero. Entries  $F(i, j)$  in all other cells of the matrix are generated as the score of the best alignment in the position matching  $x_i$  and  $y_i$ , and calculated using the following recurrence:

$$F(i, j) = \max \begin{cases} 0 \\ F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

where,  $s(x_i, y_i)$  is similarity score for  $x_i$  and  $y_i$  and  $d$  is constant gap penalty. Similarity score between  $x_i$  and  $y_i$  could be assigned as 1 for  $x_i = y_i$  and 0 for  $x_i \neq y_i$ . Smith-Waterman algorithm uses a constant gap penalty where each gap would be given the same penalty regardless of its length. Upon the filling of  $F(i, j)$  process is completed, an optimal alignment with gaps for two sequences is constructed. Trace-back is applied which starts from the maximum of  $F(i, j)$  in the whole matrix to the first  $F(i, j) = 0$  is found. If we have a query DNA,  $q$ , and a targeted DNA,  $t$ : during the trace-back processing, a number of gaps (“-”) have been inserted into  $q$  and  $t$ , so that the resulting sequences  $q'$  and  $t'$  which have the same length,  $L$ .

## 2 The Framework for Minimum Cost Process in Searching Similar DNA Sequences

The proposed framework is to solve the following problem statement: “Given a query DNA  $q$  of length  $m$  and  $d$  databases of candidate DNA sequences  $T$ , by means of a local alignment process search a set of sequences in  $d$  databases which highly similar to DNA  $q$ ”.

In our framework, there are four main phases involve, and denoted as {generate the patterns, scanning the patterns, ranking, optimal local alignment} (Fig. 1).

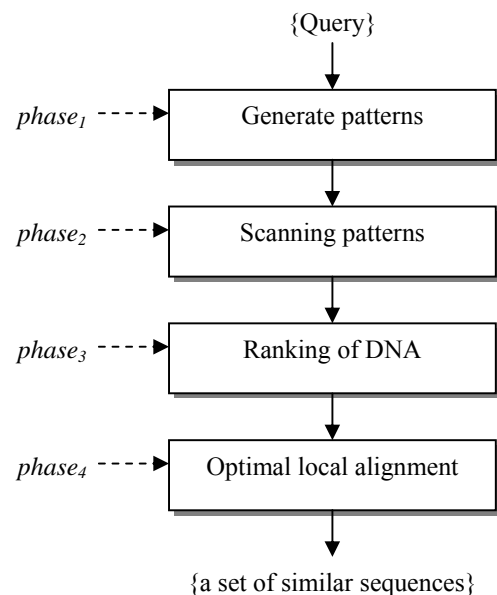


Fig. 1 The process flow

In *phase<sub>1</sub>*, by using *ant algorithm* a set of random uniquely patterns with length  $\lambda$  are generated from  $q$ ,  $P = \{\rho_1, \rho_2, \dots, \rho_\delta\}$ . Let  $T = \{t_1, t_2, \dots, t_n\}$  be a collection of DNA sequences in a database, the *phase<sub>2</sub>*, filtering process is started with scanning the  $\delta$  patterns in  $P$  to each sequences in  $T$ . This process is a multi-pattern matching problem consists in finding all occurrence of the patterns with length  $\lambda$  in  $P$  in all DNA sequences with various length in  $T$ . A “match” occurs when a pattern from  $q$  is match character by character in  $t_i \in T$  starting at some index  $j$ . Therefore that match can be denoted as:  $t_i[j] = \rho_i[0]$ ,  $t_i[j+1] = \rho_i[1]$ , ...,  $t_i[j+\lambda-1] = \rho_i[\lambda-1]$ . For each  $t_i \in T$ , patterns matching score is calculated which is represented

$$\Phi = \sum_{i=1}^k ss, k \leq \delta, \text{ where } ss \text{ is matching}$$

score for a pattern. The detail regarding to scoring method is discussed in Section 2.1.

The main purpose for filtering mechanism by using exact pattern matching is to quickly discard DNA sequences from databases (hopefully the large number of sequences) that do not much similar to query sequence. It is important to notice that this filtering mechanism is unable to discover the similarity degree between the two DNA sequences. Hence, a dynamic programming Smith-Waterman algorithm is still extremely needed.

The third phase is straight forward. A simple quick sort algorithm [19] with time complexity  $O(n \log n)$  is used to rank the DNA sequences in databases based on their matching score,  $\Phi$ . Finally, by using Smith-Waterman algorithm, an optimal local sequence alignment is employed and the similarity score (alignment score and percent identity) is calculated for each  $t_i \in T$ . Fig. 2 envisages a general algorithm for seeking minimum cost for retrieving a set of similar DNA sequences from databases to query DNA sequence.

### 2.1 Scoring Methods

The solid approach in which distance or similarity between two strings (DNA sequences) is expressed by means of a scoring function or matrices for matches, mismatches and gaps. The score for each pattern found in an appropriate sequence and the score for the optimal local alignment of two sequences with length  $L$  can be calculated by using equations (1) and (2) respectively:

$$ss(\rho_i, t_i) = s(\rho_i[0], t_i[j]) + s(\rho_i[1], t_i[j+1]) + \dots + s(\rho_i[\lambda-1], t_i[j+\lambda-1]) \quad (1)$$

$$\text{Alignment\_Score}(q', t') = \sum_{i=1}^L s(x_i, y_i) \quad (2)$$

The percent identity formula for the aligned of sequences  $q'$  and  $t'$  is given in the following equation,  $\%ID = \left(\frac{R}{L}\right) \times 100$ , where  $R$  is number of matching residues after alignment process.

**Input** : 1) Query sequence,  $q$   
 2) Targeted sequences,  $T = \{t_1, t_2, \dots, t_n\}$

**Output** : A set of similar sequences to  $q$

- Input,  $q$
- Generate  $\delta$  exclusive random patterns from  $q$  with length  $\lambda$ ,  $P = \{\rho_1, \rho_2, \dots, \rho_\delta\}$
- For each sequence in  $T$  // filtering process
  - With KMP algorithm,  $\delta$  patterns are scan for exact matching.
  - Calculate exact matching score for each  $t_i \in T$ ,
$$\Phi = \sum_{i=1}^k ss, k \leq \delta$$
- For each  $t_i \in T$  that have the highest matching score
  - Apply Smith-Waterman algorithm for pair-wise alignment ( $q, t_i$ )
  - Calculate the homology (alignment score, identity)

Fig. 2 A general algorithms of the framework

### 3 Experiments

Based on this framework, a tool has been developed using Java programming language. In order to manipulate the operations of reading and converting the GenBank DNA sequences format several BioJava classes have been embedded into the program. The hardware used for the experiments is based on Intel Pentium IV architecture which has 1.8 GHz processors with 640MB of RAM and running under Windows XP operating systems. Several classes from BioJava [18] libraries are reused in order to establish communication between tool application and genome databases. We have downloaded five GenBank DNA sequence databases which contain approximately 320,000

sequences. From each database, 10,000 DNA sequences have been selected (50,000 sequences) to be used in the experiments. The performance and reliability of the developed tool was evaluated using various query sequence lengths. These query sequences have been chosen randomly from the 50,000 sequences selected before.

The experiment results from five selected query are shown in Table 1. The tabulated results are based on the process of retrieving 10% sequences from the 50,000 DNA sequences in databases which are similar to the query sequence. The length of each generated pattern ( $\lambda$ ) is 6. The results show that filtering mechanism could reducing the time for retrieving a set of similar sequences from databases to the query sequence. The filtering process (KMP algorithms) successfully generate a group of DNA sequences from database that have a highly potential similar to the query sequence. For instance in  $q_3$ , with the first 5602 (602 sequences are unrelated to query) sequences produced after filtering process, the tool has successfully retrieves 5000 (10% from 50,000) sequences which are similar to query. Therefore, 44,398 sequences have been skipped from exhaustive local optimal alignment process (Smith-Waterman algorithms).

Meanwhile, for a query with length is quite small (e.g.  $q_1$  and  $q_2$  in Table 1) there are a huge number of irrelevant sequences have been selected during filtering process. This problem could be solved by increase the size of patterns length. From the experiments with the pattern length,  $\lambda = 8$ , for  $q_1$ : number of discarded sequences increase to 41653 (83.31%); and for  $q_2$ : number of discarded sequences increase to 43208 (86.42%). Nevertheless, the optimal length of the pattern which is could generate the best result in filtering process is 6.

Table 1 Experiment results

Query #	Query length	No. of S-W algorithm performed	% discarded from S-W algorithm
$q_1$	99	11955 (6955)	76.09
$q_2$	660	6854 (1854)	86.29
$q_3$	1528	5602 (602)	88.88
$q_4$	2116	5455 (455)	89.09
$q_5$	2593	5331 (331)	89.34

## 4 Conclusions

Scanning and searching of DNA sequence databases is a regular and often repeated task in molecular biology. The need for speeding up with optimal results for this action comes from the exponential growth of the bio-sequences databases.

The paper demonstrates the employ of exact string matching KMP algorithm which has time complexity  $O(m+n)$  as a filtration mechanism before an optimal alignment (Smith-Waterman algorithm) between sequences is implemented. Based on the propose framework, a tool has been developed using object oriented programming language Java. Five DNA databases from GenBank have been downloaded and experiments are executed. Experiments prove that KMP algorithm could discard a huge number of irrelevant targeted sequences. Consequently, the number of Smith-Waterman process could be reduced and a set of similar DNA sequences to a query is generated more quickly. Finally, we plan to improve our framework and the following matters are to be considered in our future research:

- The use of parallel computers to speed the processing of those phases.
- To increase the sensitivity of filtering mechanism, we are concern to focus on the problem of pattern matching that allows errors (approximate string matching).
- Trying to use another approach in filtering processing such as Aho-Corasick automaton machine and its variants.

### References:

- [1] Altschul, S. F., Gish, W., Miller, W., Myers, E. & Lipman, D. J. Basic Local Alignment Search Tool. *Journal Molecular Biology*, Vol. 215, 1990, pp. 403 – 410.
- [2] Asim, Y. & Jack, J. D. Biological Sequence Alignment on the Computational Grid Using the GrADS Framework. *Future Generation Computer Systems*, Vol. 21, 2005, pp. 980 – 986.
- [3] Chen, C. & Schmidt, B. An Adaptive Grid Implementation of DNA Sequence Alignment. *Future Generation Computer Systems*, Vol. 21, 2005, pp. 988 – 1003.
- [4] Choi, K. P., Zeng, F. & Zhang, L. Good spaced seeds for homology search.

- Bioinformatics*, Vol. 20, No. 7, 2004, pp. 1053 – 1059.
- [5] Cohen, J. Computer Science and Bioinformatics. *COMMUNICATION OF THE ACM*, Vol. 48, No. 3, 2005, pp. 72 – 78.
- [6] Dayhoff, M. O. & Schwartz, R. M. Atlas of Protein Sequence and Structure. *National Biomedical Research Foundation*, Vol. 3, 1978, pp. 353 – 358.
- [7] Delcher, A. L., Phillippy, A., Carlton, J. & Salzberg, L. Fast Algorithms for Large-Scale Genome Alignment and Comparison. *Nucleic Acids Research*, Vol. 30, No. 11, 2002, pp. 2478 – 2483.
- [8] Kucherov, G., Noe, L. & Ponty, Y. Estimating seed Sensitivity on Homogenous Alignments. *Proceedings of IEEE 4<sup>th</sup> Symposium on Bioinformatics and Bioengineering*, 2004, pp. 387 – 394.
- [9] Knuth, D. E., Morris, J. H. & Pratt, V. R. Fast Pattern Matching in Strings. *SIAM Journal Computer*, Vol. 6, No. 2, 1977, pp. 323 – 350.
- [10] Lecroq, T. Experimental Results on String Matching Algorithms, *Software-Practice and Experience*, Vol. 25(7), 1995, pp. 727 – 765.
- [11] Li, M. & Ma, B. PatternHunter II: Highly Sensitive and Fast Homology Search, *Genome Informatics*, 14, 2003, 164 – 175.
- [12] Ma, B., Tromp, J. & Li, M. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3), 2002, 440 – 445.
- [13] Needleman, S. B. & Wunsch, C. D. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Sequences. *Journal of Molecular Biology*, Vol. 48, 1970, pp. 443 – 453.
- [14] Noe, L & Kucherov, G. YASS: enhancing the sensitivity of DNA similarity search. *Nucleic Acids Research – Web server issue*, Vol. 33, 2005, 540 – 543.
- [15] Pearson, W. R. & Lipman, D. J. Improved Tools for Biological Sequence Comparisons. *Proceedings of The National Academy of Sciences of The USA*, Vol. 85, 1988, pp. 2444 – 2448.
- [16] Reingold E. M., Urban, K. J. & Gries, D. K-M-P String Matching Revisited. *Information Processing Letters*, Vol. 64, 1997, pp. 217 – 223.
- [17] Smith, T. F. & Waterman, M. S. Identification of Common Molecular Subsequences, *Journal of Molecular Biology*, Vol. 147, 1981, pp. 195 – 197.
- [18] <http://en.wikipedia.org/biojava>
- [19] <http://en.wikipedia.org/wiki/Quicksort>