Biochemical and Biophysical Research Communications 502 (2018) 313–317

Contents lists available at ScienceDirect

Biochemical and Biophysical Research Communications

journal homepage: www.elsevier.com/locate/ybbrc

Pairwise alignment for very long nucleic acid sequences

Jie Sun, Ke Chen*, Zhixiang Hao

School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin, 300387, China

ARTICLE INFO

Article history:
Received 14 May 2018
Accepted 18 May 2018
Available online 29 May 2018

Keywords:
Sequence alignment
Smith–Waterman algorithm
Dynamic programming
Memory deduction
Very long sequence

ABSTRACT

Sequence alignment is one of the fundamental problems in computational biology and has numerous applications. The Smith–Waterman algorithm generates optimal local alignment for pairwise alignment task and has become a standard algorithm in its field. However, the current version of the Smith–Waterman algorithm demands a significant amount of memory and is not suitable for alignment of very long sequences. On the hand, the recent DNA sequencing technologies have produced vast amounts of biological sequences. Some nucleic acid sequences are very long and cannot employ the Smith–Waterman algorithm. To this end, this study proposes the PAAVLS algorithm that follows the dynamic programming technique employed by the Smith–Waterman algorithm and largely reduces the demand of memory. The proposed PAAVLS algorithm can be employed for alignment of very long sequences, i.e., sequences contain more than 100,000,000 nucleotides, on a personal computer. Additionally, the running time of the proposed PAAVLS algorithm is comparable with the running time of the standard Smith–Waterman algorithm.

1. Introduction

To date, vast amounts of biological sequence data have been produced by recent DNA sequencing technologies [1]. However, the computational algorithms/programs for the analysis of sequence data are not sufficiently efficient when compared with the magnitude of the sequence data. Currently, sequence alignment is still the cornerstone in biological sequence analysis. In the area of sequence alignment, there are three fundamental problems. First, for a pair of sequences, how to measure the similarity between the two sequences and how to calculate the optimal alignment solution. Second, for a given sequence, how to identify all similar sequences in a sequence database. Third, for a number of sequences (more than 2), how to calculate the optimal multiple sequence alignment solution. For the first problem, a number of dynamic programming algorithms are developed [2]. For instance, the Needleman– Wunsch algorithm generates global and optimal alignment solution for a pair of sequences [3]. On the other hand, the Smith–Waterman algorithm identifies the most similar local regions between two sequences [4]. Both algorithms are implemented in the EMBOSS sequence alignment package [5,6]. For the second problem, the BLAST and FASTA programs are developed and they have very good performance in identification of similar sequences in a sequence database [7,8]. Since these programs perform database search and consume a significant amount of time, they concern more about computational efficiency than alignment accuracy. These algorithms usually employ some heuristic techniques, i.e., the *k*-tuple methods, to reduce the time complexity. The deficiency of these algorithms is that they do not guarantee to produce an optimal alignment solution. For the third problem, the dynamic programming method employed by pairwise sequence alignment cannot be directly extended to multiple sequence alignment, since it involves the calculation of an *n*-dimensional matrix, resulting in an exponential time and space complexity. To this end, a number of progressive alignment methods by incorporating information from all pairwise alignment results into the objective function are proposed [9–14].

Though Smith–Waterman algorithm generates optimal solution for pairwise alignment tasks, it demands a significant amount of memory and cannot be used for alignment of very long sequences. The concept of scoring matrix is the core ideal of dynamic programming-based algorithms, including the Smith–Waterman algorithm. This matrix is indispensable in the trace back step in recursively identifying the best local alignment. The size of the matrix is $m \times n$, where m and n are the lengths of the two aligned sequences. It is known that some long non-coding RNAs (lncRNAs) can be very long and contain more than 100,000 nucleotides. When a pairwise alignment is performed between two long RNA

* Corresponding author.
E-mail address: chenke@tjpu.edu.cn (K. Chen).

https://doi.org/10.1016/j.bbrc.2018.05.134
0006-291X/

Table 1
The major steps of Smith–Waterman Algorithm.

Algorithm 1: Smith–Waterman Algorithm
(1) Initialization: initialize the first row and first column of $H^{n\times m}$ to 0, that is $H_{i0} = H_{0j} = 0$, for $0 \le i \le n, 0 \le j \le m$.
(2) Calculation of the scoring matrix $H^{n\times m}$. For $i = 1$ to n For $j = 1$ to m $H_{ij} = \max\{0, H_{i-1,j-1} + \text{similarity}(a_i, b_j), H_{i-1,j} - \text{penalty}, H_{i,j-1} - \text{penalty}\}$ End End
(3) Trace back and find the best local alignment. Firstly, identify the maximal value in $H^{n\times m}$ (denote the maximal value as H_{xy}). Secondly, based on the iterative function in step (2), find the source path for H_{xy}. Lastly, the traced source path is transformed to the best local alignment.

sequences which both contain more than 100,000 nucleotides, the Smith–Waterman algorithm creates a scoring matrix containing more than 100,000*100,000 elements, which may overflow the memory of a computer. If the aligned sequences contain more than a million nucleotides, i.e., the genome of a bacteria, the current form of the Smith–Waterman algorithm cannot function properly.

To facilitate the pairwise alignment of very long sequences, Li and colleagues have developed a single-GPU parallelization of the Smith-Waterman algorithm [15]. The proposed algorithm can process pairwise alignment of very long sequences. However, this algorithm is GPU-based and cannot be employed by CPU-based computers. To this end, we propose an algorithm that follows the dynamic programming technique as the Smith–Waterman algorithm and largely reduces the demand of memory. The proposed algorithm can properly functions on a normal personal computer in performing alignment of very long sequences, i.e., the pairwise alignment between two nucleic acid sequences which contain more than 100,000,000 nucleotides.

2. Materials and methods

Let $A = a_1a_2\ldots\ldots a_n$ and $B = b_1b_2\ldots\ldots b_m$ be the sequences to be aligned, where n and m are the lengths of A and B respectively. Denote $A_i = a_1a_2\ldots\ldots a_i$ and $B_j = b_1b_2\ldots\ldots b_j$ are the subsequences of A and B, where $1 \le i \le n$, $1 \le j \le m$. The scoring matrix $H^{n\times m} = \{H_{ij}\}_{1\le i\le n,\ 1\le j\le m}$ records the highest alignment score between A_i and B_j.

2.1. Smith–Waterman algorithm

The Smith–Waterman algorithm consists of 3 steps: initialization, calculation of the scoring matrix $H^{n\times m}$ and trace back. The details of the algorithm are given in Table 1. As we see, the algorithm demands the allocation of memory for $H^{n\times m}$. When performing an alignment between two very long sequences, the size of $H^{n\times m}$ may overflow the memory of a computer. To this end, this algorithm needs to be modified to facilitate the alignment of very long sequences.

2.2. A pairwise alignment algorithm for very long sequences

In this section, we propose a Pairwise Alignment Algorithm for Very Long Sequences (PAAVLS), see Table 2. The PAAVLS algorithm consists of 3 steps:

Table 2
The major steps of PAAVLS Algorithm.

Algorithm 2: PAAVLS Algorithm
(1) Initialization: initialize the first row and first column of H_{ij} to 0, that is $H_{i0} = H_{0j} = 0$, for $0 \le i \le n, 0 \le j \le m$. Initialize the maximal value of H_{ij} to 0: $H_{max} = 0$.
(2) Calculation of the maximal value of scoring matrix $H^{n\times m}$. For $i = 1$ to n For $j = 1$ to m $H_{ij} = \max\{0, H_{i-1,j-1} + \text{similarity}(a_i, b_j), H_{i-1,j} - \text{penalty}, H_{i,j-1} - \text{penalty}\}$ H_{max}=max$\{H_{max}, H_{ij}\}$ End End Denote $H_{max} = H_{xy}$
(3) Trace back and find the source path for H_{xy} by calling Trace(H_{xy}, x, y).