# Opti-SW: An improved gene sequence alignment algorithm

**Leixiao Li[1,2,3], Jing Gao[1,*], Yanfeng Liu[2,3]**

*1. College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot 010018, China*

*2. College of Data Science and Application, Inner Mongolia University of Technology, Hohhot 010080, China*

*3. Inner Mongolia Autonomous Region Engineering & Technology Research Center of Big Data Based Software Service, Hohhot 010080, China*

*gaojing@imau.edu.cn*

ABSTRACT. *This paper aims to improve the speed and complexity of Smith-Waterman (SW) algorithm. For this purpose, the SW algorithm was improved by reducing the complexity and task load of the computation of the scoring matrix without sacrificing the alignment accuracy. Then, the optimized algorithm, denoted as the Opti-SW, was verified through experiment. The results show that the Opti-SW boasts low time complexity, fast computing speed and light computing load. The research findings shed new light on the database search for gene sequences.*

RÉSUMÉ. *Cet article vise à améliorer la vitesse et la complexité de l'algorithme Smith-Waterman (SW). À cet effet, l'algorithme SW a été amélioré en réduisant la complexité et la charge de travail du calcul de la matrice de scoring sans sacrifier la précision de l'alignement. Ensuite, l'algorithme optimisé, noté Opti-SW, a été vérifié par des expérimentations. Les résultats montrent que l'Opti-SW se caractérise par une faible complexité temporelle, une vitesse de calcul rapide et une charge de calcul légère. Les résultats de la recherche ont jeté un nouvel éclairage sur la recherche dans la base de données des séquences de gènes.*

KEYWORDS: *gene sequence alignment, smith-waterman (SW) algorithm, optimization, opti-SW.*

MOTS-CLÉS: *alignement de séquences de gènes, algorithme de smith-waterman (SW), optimisation, opti-SW.*

DOI:10.3166/ISI.23.6.73-85 

## 1. Introduction

Sequence alignment, an important operation in bioinformatics, has been widely used in such field as disease diagnosis, drug engineering and biomaterial engineering (Chen, 2016). One of the most popular methods for sequence alignment is Smith-Waterman (SW) algorithm. This pairwise sequence alignment algorithm is known for its high accuracy (Zhang, 2015). However, the SW algorithm is not suitable for long sequence alignment and other big data scenarios, due to its high spatiotemporal complexity.

Many heuristic algorithms have been developed to achieve long sequence alignment, including Fasta algorithm, Blast algorithm and Burrows-Wheeler Transform (BWT)-SW (Gao *et al*., 2014). Nonetheless, none of these heuristic approaches has sufficient alignment accuracy to overtake the dominance of the SW algorithm. As a result, the SW algorithm has been optimized with the aid of various techniques, such as graphic processing unit (GPU) (Wei *et al*., 2009; Jain and Kumar, 2014; Liu *et al*., 2012; Cao *et al*., 2015; Liu *et al*., 2013), message passing interface (MPI) (Balaji *et al*., 2008; Xue, 2015), cluster (Feng and Gao, 2016; Feng, 2015; Li, 2011), single instruction multiple data (SIMD) (Xu *et al*., 2017; Daily, 2016; Zhao *et al*., 2013; Farrar, 2007) and field programmable gate array (FPGA) (Benkrid et al., 2009; Wang *et al.*, 2015). With heavy computing load and slow computing speed, these optimized SW algorithms still cannot satisfy the needs of largescale gene sequence alignment.

To overcome these difficulties, this paper optimizes the SW algorithm by reducing the complexity and task load of the computation of the scoring matrix without sacrificing the alignment accuracy. Then, the optimized algorithm, denoted as the Opti-SW, was verified through experiment. The results show that the Opti-SW boasts low time complexity, fast computing speed and light computing load.

The remainder of this paper is organized as follows: Sections 2 introduces the SW algorithm; Section 3 describes the philosophy of improving the SW algorithm into the Opti-SW; Section 4 verifies the accuracy of the Opti-SW through experiment and compares the performance between the Opti-SW and the SW algorithm; Section 5 wraps up this paper with meaningful conclusions.

## 2. SW algorithm

The SW algorithm is a dynamic programming strategy to search for the local optimal alignment between two gene/protein sequences. Inspired by the Needleman-Wunsch algorithm, the SW algorithm mainly supports double sequence alignment in the local range. This is the most accurate method for double and multiple sequence alignments. However, the high accuracy is achieved at the sake of excessively high spatiotemporal complexity.

Let s and t be two gene sequences (Figure 1), whose lengths are m and n, respectively. Then, the i-th character of sequence s can be denoted as $s_i$ ($1 \leq i \leq m$), and

the j-th character of sequence t can be denoted as $t_j$ ($1 \le j \le n$). Let D be the score matrix (Figure 2) of the two sequences, and $d_{ij}$ be the element in the i-th row and j-th column of the score matrix.

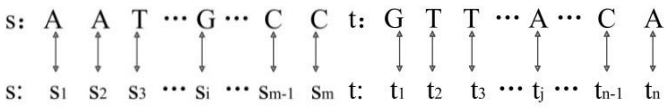


*Figure 1. Gene sequences a and t*

| | | Seq t: | $t_1$ | $t_2$ | ⋯ | $t_j$ | ⋯ | $t_{n-1}$ | $t_n$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $d_{00}$ | $d_{01}$ | $d_{02}$ | ⋯ | $d_{0j}$ | ⋯ | $d_{0,n-1}$ | $d_{0n}$ |
| Seq s: | $s_1$ | $d_{10}$ | $d_{11}$ | $d_{12}$ | ⋯ | $d_{1j}$ | ⋯ | $d_{1,n-1}$ | $d_{1n}$ |
| | $s_2$ | $d_{20}$ | $d_{21}$ | $d_{22}$ | ⋯ | $d_{2j}$ | ⋯ | $d_{2,n-1}$ | $d_{2n}$ |
| | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ |
| | $s_i$ | $d_{i0}$ | $d_{i1}$ | $d_{i2}$ | ⋯ | $d_{ij}$ | ⋯ | $d_{i,n-1}$ | $d_{in}$ |
| | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ |
| | $s_{m-1}$ | $d_{m-1,0}$ | $d_{m-1,1}$ | $d_{m-1,2}$ | ⋯ | $d_{m-1,j}$ | ⋯ | $d_{m-1,n-1}$ | $d_{m-1,n}$ |
| | $s_m$ | $d_{m0}$ | $d_{m1}$ | $d_{m2}$ | ⋯ | $d_{mj}$ | ⋯ | $d_{m,n-1}$ | $d_{mn}$ |

*Figure 2. Score matrix D*

The workflow of SW algorithm can be described below:

**Step 1: Initialization**

Assign zero to the element in the first row and first column of score matrix D, that is, $d_{0j}=0(1 \le j \le n)$ and $d_{i0}=0(1 \le i \le m)$ (Figure 3).

| | Seq t: | $t_1$ | $t_2$ | ⋯ | $t_j$ | ⋯ | $t_{n-1}$ | $t_n$ | |
|---|---|---|---|---|---|---|---|---|---|
| | $d_{00}=0$ | $d_{01}=0$ | $d_{02}=0$ | ⋯ | $d_{0j}=0$ | ⋯ | $d_{0,n-1}=0$ | $d_{0n}=0$ | first row |
| Seq s: $s_1$ | $d_{10}=0$ | | | | | | | | |
| $s_2$ | $d_{20}=0$ | | | | | | | | |
| ⋯ | ⋯ | | | | | | | | |
| $s_i$ | $d_{i0}=0$ | | | | | | | | |
| ⋯ | ⋯ | | | | | | | | |
| $s_{m-1}$ | $d_{m-1,0}=0$ | | | | | | | | |
| $s_m$ | $d_{m0}=0$ | | | | | | | | |
| | first column | | | | | | | | |

*Figure 3. Initialization of the elements in the first row and first column*

**Step 2: Calculation of score matrix D**

The score function of score matrix D can be expressed as:

$$\begin{cases} p(a,a)=1 \\ p(a,b)=0,\ (a\neq b) \\ p(a,-)=p(-,b)=-1 \end{cases} \quad (1)$$

where p(a,a)=1 indicates that the score is 1 when the two characters match; p(a,b)=0, (a≠b) indicates that the score is 0 when the two characters do not match; p(a,-)=p(b,-)=-1 indicates that the score is -1 when one of the two characters is vacant.

The element $d_{ij}$ in score matrix D can be calculated as:

$$d_{ij} = \max \begin{cases} d_{i-1,j}+p(s_i,-) \\ d_{i,j-1}+p(-,t_j) \\ d_{i-1,j-1}+p(s_i,t_j) \\ 0 \end{cases} (1\leq i\leq m, 1\leq j\leq n) \quad (2)$$

According to Equations 1 and 2, each element $d_{ij}$ of the score matrix can be calculated, and used to comprise the score matrix.

**Step 3: Backtracking**

Find the largest element $d_{ij}$ in score matrix D, determine whether $d_{ij}$ is calculated from $d_{i,j-1}$, $d_{i-1,j}$ or $d_{i-1,j-1}$ elements, and write down the result. Repeat this process until reaching an element whose value is zero. In this way, a backtracking path can be obtained (Figure 4).

| | Seq t: | ··· | $t_q$ | ··· | $t_{j-1}$ | $t_j$ | ··· | $t_n$ |
|---|---|---|---|---|---|---|---|---|
| | $d_{00}$ | ··· | $d_{0q}$ | ··· | $d_{0,j-1}$ | $d_{0j}$ | ··· | $d_{0n}$ |
| Seq s: ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· |
| $s_p$ | $d_{p0}$ | ··· | $d_{pq}=0$ | ··· | $d_{p,j-1}$ | $d_{pj}$ | ··· | $d_{pn}$ |
| $s_{p+1}$ | $d_{p+1,0}$ | ··· | $d_{p+1,q}$ | ··· | $d_{p+1,j-1}$ | $d_{p+1,j}$ | ··· | $d_{p+1,n}$ |
| ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· |
| $s_{i-2}$ | $d_{i-2,0}$ | ··· | $d_{i-2,q}$ | ··· | $d_{i-2,j-1}$ | $d_{i-2,j}$ | ··· | $d_{i-2,n}$ |
| $s_{i-1}$ | $d_{i-1,0}$ | ··· | $d_{i-1,q}$ | ··· | $d_{i-1,j-1}$ | $d_{i-1,j}$ | ··· | $d_{i-1,n}$ |
| $s_i$ | $d_{i0}$ | ··· | $d_{i,q}$ | ··· | $d_{i,j-1}$ | $d_{ij}$ | ··· | $d_{in}$ |
| ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· |
| $s_m$ | $d_{m0}$ | ··· | $d_{m,q}$ | ··· | $d_{m,j-1}$ | $d_{m,j}$ | ··· | $d_{mn}$ |

*Figure 4. Backtracking path*

**Step 4: Solving local optimal alignment**

Starting with the largest element $d_{ij}$ of scoring matrix D, search for the local optimal alignment through reverse backtracking according to the path generated in

Step 3 until reaching an element whose value is zero. During the backtracking process, if $d_{ij}$ comes from $d_{i-1,j}$, $s_i$ should be compared with "-"; if $d_{ij}$ comes from $d_{i,j-1}$, "-" should be compared with $t_j$; if $d_{ij}$comes from $d_{i-1,j-1}$, $s_i$ should be compared with $t_j$.

## 3. Improvement of the SW algorithm

Recent years has seen some improvements to the SW algorithm. For instance, Step 2 of the SW algorithm has been modified as: determine the value of element $d_{i-1,j-1}$ in the score matrix according to the upper element $d_{i-1,j}$, front element $d_{i,j-1}$ and diagonal element $d_{i-1,j-1}$ of the matrix; Step 3 has been revised into: starting with the largest element $d_{ij}$ of matrix D, determine whether $d_{ij}$ is calculated from $d_{i,j-1}$, $d_{i-1,j}$ or $d_{i-1,j-1}$ elements, and write down the result; repeat this process until reaching an element whose value is zero. After these improvements, the SW algorithm can record the source of element $d_{ij}$ when its value is calculated in Step 2, eliminating the need to compute its value in source tracking of Step 3. Thus, the improved algorithm features much less time complexity than the original one. This train of thought is followed in this paper to improve the SW algorithm.

### *3.1. Philosophy of algorithm improvement*

#### *3.1.1. Reducing computing load*

(1) According to equations 1 and 2, the elements $d_{00}$, $d_{10}$ and $d_{01}$ of the score matrix and the score functions $p(s_1,-)$, $p(-,t_1)$ and $p(s_1,t_1)$ can be obtained as:

$$\begin{cases} d_{00}=0 \\ d_{10}=0 \\ d_{01}=0 \\ p(s_1,-)=-1 \\ p(-,t_1)=-1 \\ p(s_1,t_1)=0 \text{ or } 1 \end{cases}$$

The above conclusion can be used to calculate element $d_{11}$ in the score matrix:

$$d_{11} = \max \begin{cases} d_{00}+p(s_1,t_1) \\ d_{10}+p(s_1,-) \\ d_{01}+p(-,t_1) \\ 0 \end{cases} = d_{00}+p(s_1,t_1) = p(s_1,t_1)$$

It can be inferred from equation $d_{11}=d_{00}+p(s_1,t_1)$ that element $d_{11}$ of the score matrix is calculated from $d_{00}$. In other words, $d_{11}$ is derived from $d_{00}$. According to equation $d_{11}=p(s_1,t_1)$, if the first elements of the two sequences do not match, then $p(s_1,t_1)=0$ and $d_{11}=p(s_1,t_1)=0$ ; otherwise, $p(s_1,t_1)=1$ and $d_{11}=p(s_1,t_1)=1$ , which correspond to the maximum value of $d_{11}$. For simplicity, $d_{11}$is considered as equal to or smaller than 1 here.

(2) According to equations 1 and 2 and the above conclusion that $d_{11} \leq 1$, the elements $d_{02}$ and $d_{01}$ of the score matrix and the score functions $p(s_1, -)$, $p(-, t_2)$ and $p(s_1,t_2)$ can be obtained as:

$$\begin{cases} d_{11} \leq 1 \\ d_{02}=0 \\ d_{01}=0 \\ p(s_1,-)=-1 \\ p(-,t_2)=-1 \\ p(s_1,t_2)=0 \text{ or } 1 \end{cases}$$

Then, element $d_{12}$ in the score matrix can be calculated as:

$$d_{12} = \max \begin{cases} d_{02}+p(s_1,-) \\ d_{11}+p(-,t_2) \\ d_{01}+p(s_1,t_2) \\ 0 \end{cases} = d_{01}+p(s_1,t_2)=p(s_1,t_2)$$

It can be inferred from equation $d_{12}=d_{01}+p(s_1,t_2)$ that element $d_{12}$ of the score matrix is calculated from $d_{01}$. In other words, $d_{12}$ is derived from $d_{01}$. If the proper elements of the two sequences do not match, then $p(s_1,t_2)=0$ and $d_{12}=p(s_1,t_2)=0$; otherwise, $p(s_1,t_2)=1$ and $d_{12}=p(s_1,t_2)=1$, which correspond to the maximum value of $d_{12}$. For simplicity, $d_{12}$is considered as equal to or smaller than 1 here.

Similarly, it can be proved that all elements $d_{1j}(1<j\leq n)$ in the second row of the score matrix are calculated from $d_{0,j-1}$, that is, $d_{13}$ is derived from $d_{02}$, $d_{14}$ is derived from $d_{03}$, …, $d_{1j}$ is derived from $d_{0,j-1}(1 < j \leq n)$. If the proper elements of the two sequences do not match, then $d_{13} = 0$, $d_{14} = 0$, …, $d_{1j}=0(1<j\leq n)$; otherwise, $d_{13} = 1$, $d_{14} = 1$, …, $d_{1j}=1(1<j\leq n)$. To sum up, for all elements $d_{1j}(1<j\leq n)$ in the second row of the score matrix, if $p(s_1,t_j)=1$, then $d_{1j}=1(1<j\leq n)$ and if $p(s_1,t_j)=0$, then $d_{1j}=0(1<j\leq n)$:

$$d_{1j}=\begin{cases} 1, p(s_1,t_j)=1 \\ 0, p(s_1,t_j)=0 \end{cases} \quad (1<j\leq n) \tag{3}$$

(3) According to equations 1 and 2 and the above conclusion that $d_{11} \leq 1$, the elements $d_{20}$ and $d_{10}$ of the score matrix and the score functions $p(s_2,-)$, $p(-, t_1)$ and $p(s_2,t_1)$ can be obtained as:

$$\begin{cases} d_{11} \leq 1 \\ d_{20}=0 \\ d_{10}=0 \\ p(s_2,-)=1 \\ p(-,t_1)=1 \\ p(s_2,t_1)=0 \text{ or } 1 \end{cases}$$

Then, element $d_{21}$ in the score matrix can be calculated as:

$$d_{21}= \max \begin{cases} d_{11}+p(s_2,-) \\ d_{20}+p(-,t_1) \\ d_{10}+p(s_2,t_1) \\ 0 \end{cases} =d_{10}+p(s_2,t_1)=p(s_2,t_1)$$

It can be inferred from equation $d_{21}=d_{10}+p(s_2,t_1)$ that element $d_{21}$ of the score matrix is calculated from $d_{10}$. In other words, $d_{21}$ is derived from $d_{10}$. If the proper elements of the two sequences do not match, then $p(s_2,t_1)=0$ and $d_{21}=p(s_2,t_1)=0$; otherwise, $p(s_2,t_1)=1$ and $d_{21}=p(s_2,t_1)=1$, which correspond to the maximum value of $d_{21}$. For simplicity, $d_{21}$is considered as equal to or smaller than 1 here.

Similarly, it can be proved that all elements $d_{1j}(1<j\leq m)$ in the second column of the score matrix are calculated from $d_{i-1,0}$, that is, $d_{31}$ is derived from $d_{20}$, $d_{41}$ is derived from $d_{30}$, …, $d_{i1}$ is derived from $d_{i-1,0}(1<i\leq m)$. If the proper elements of the two sequences do not match, then $d_{31}=0$, $d_{41}=0$, …, $d_{i1}=1(1<i\leq m)$; otherwise, $d_{31}=1$, $d_{41}=1$, …, $d_{i1}=1(1<i\leq m)$. To sum up, for all elements $d_{i1}(1<i\leq m)$ in the second column of the score matrix, if $p(s_i,t_1)=1$, then $d_{i1}=1(1<i\leq m)$ and if $p(s_i,t_1)=0$, then $d_{i1}=0(1<i\leq m)$:

$$d_{i1}=\begin{cases} 1, p(s_i,t_1)=1 \\ 0, p(s_i,t_1)=0 \end{cases} \quad (1<i\leq m) \tag{4}$$

To sum up, all elements $d_{1j}(1<j\leq n)$ of the second row in the score matrix are derived from $d_{0,j-1}(1<j\leq n)$, all elements $d_{i1}(1<i\leq m)$ of the second column in the score matrix are derived from $d_{i-1,0}(1<i\leq m)$ and $d_{11}$ is derived from $d_{00}$. If the proper elements of the two sequences match, it means $p(s_i,t_j)=1$ and $d_{11}=1$, and then $d_{1j}=1(1<j\leq n)$ and $d_{i1}=1(1<i\leq m)$; otherwise, it means $p(s_i,t_j)=0$ and $d_{11}=0$, then $d_{1j}=0(1<j\leq n)$ and $d_{i1}=0(1<i\leq m)$ (Figure 5).

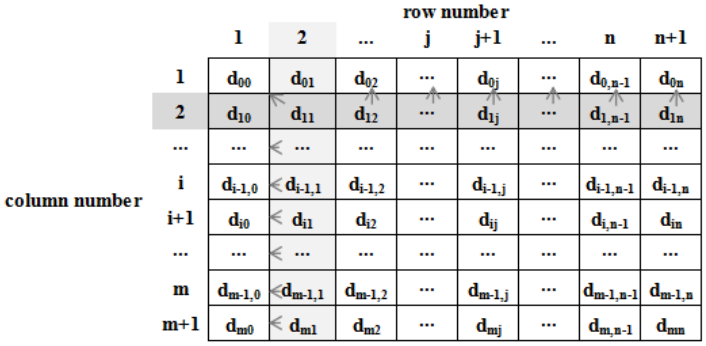


*Figure 5. Source of each element*

*3.1.2. Simplifying computing complexity*

According to equations 1 and2, the following conclusion can be drawn:

$$\begin{cases} p(s_i,-)=-1 \\ p(-,t_j)=-1 \\ d_{i-1,j-1} \geq 0 \\ p(s_i,t_j)\leq 1 \end{cases}$$

On this basis, the element $d_{ij}$ in the score matrix can be calculated as:

$$d_{ij} = max \begin{cases} d_{i-1,j} + p(s_i,-) \\ d_{i,j-1} + p(-,t_j) \\ d_{i-1,j-1} + p(s_i,t_j) \\ 0 \end{cases} = max \begin{cases} d_{i-1,j} - 1 \\ d_{i,j-1} - 1 \\ d_{i-1,j-1} + p(s_i,t_j) \end{cases}$$

Considering $num = \max(d_{i-1,j}, d_{i,j-1})$, the calculation formula of element $d_{ij}$ above can be simplified as $d_{ij}=\max \begin{cases} num-1 \\ d_{i-1,j-1}+ p(s_i,t_j) \end{cases}$.

Since $p(s_i,t_j)\leq 1$, if $num - 2 \geq d_{i-1,j-1}$, we have $num - 1 \geq d_{i-1,j-1}+1 \geq d_{i-1,j-1}+p(s_i,t_j)$ and $d_{ij}=\max \begin{cases} num-1 \\ d_{i-1,j-1}+ p(s_i,t_j) \end{cases}$=num-1 ($num-2 \geq d_{i-1,j-1}$).

If $num - 2 < d_{i-1,j-1}$, then $num - 1 < d_{i-1,j-1}+1$. Since num is an integer, then the maximum value of $num - 1$ is $d_{i-1,j-1}$, that is, $num - 1 \leq d_{i-1,j-1}$.

Whereas $p(s_i,t_j)$=0 or 1, then $d_{i-1,j-1} \leq d_{i-1,j-1} + p(s_i,t_j)$ and $num - 1 \leq d_{i-1,j-1} \leq d_{i-1,j-1}+p(s_i,t_j)$. Therefore, if $num - 2 < d_{i-1,j-1} + p(s_i,t_j)$, we have:

$$d_{ij} = \max \begin{cases} num-1 \\ d_{i-1,j-1}+p(s_i,t_j) \end{cases} = d_{i-1,j-1}+p(s_i,t_j) \ (num-2<d_{i-1,j-1})$$

Thus, the following equation is valid:

$$d_{ij} = \begin{cases} num-1, \ (num-2 \geq d_{i-1,j-1}) \ (num = \max(d_{i-1,j}, d_{i,j-1})) \\ d_{i-1,j-1}+ p(s_i,t_j), \ (num-2<d_{i-1,j-1}) \end{cases} \quad (5)$$

### *3.2. Design of Opti-SW*

Following the above philosophy of improvement, this paper proposes the Opti-SW algorithm that reduces the computing load of the score matrix through simultaneous initialization of the elements in the first row and first column and of those in the second row and second column. In this algorithm, the computing complexity is further reduced by optimizing the calculation formula of the score matrix. The workflow of the Opti-SW is illustrated in Figure 6 below.

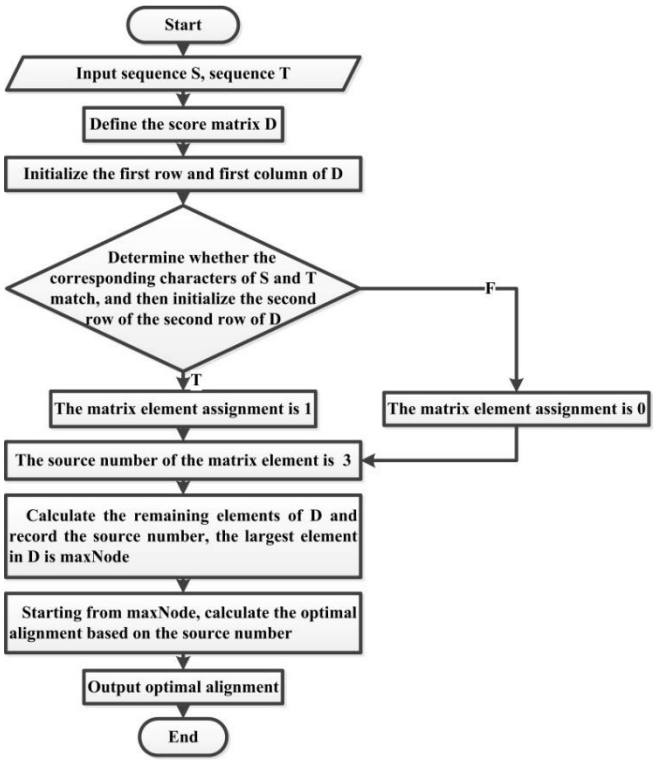


*Figure 6. Workflow of the Opti-SW*

The specific process of the proposed Opti-SW is explained as follows:

**Step 1:** initialization of the elements in the first row and those in the first column

Initialize the elements in the first row of the score matrix $d_{0j}= 0$ ($1\leq j\leq n$), and those in the first column of the score matrix $d_{i0}= 0$ ($1\leq i\leq m$).

**Step 2:** initialization of elements in the second row and the elements in the second column

Initialize the elements in the second row and second column in score matrix according to equations 3 and 4, and record the source of each element. Specifically, derive the elements in the second row from $d_{0,j-1}$ ($1<j\leq n$), and those in the second column from $d_{i-1,0}$($1<i\leq m$). If an element comes from $d_{i-1,j}$ , $d_{i,j-1}$ or $d_{i-1,j-1}$, it should be denoted as 1, 2 or 3, respectively.

**Step 3:** Calculate score matrix

Calculate score matrix D according to equation 3-3, and record the source of each element. If an element comes from $d_{i-1,j}$ , $d_{i,j-1}$ or $d_{i-1,j-1}$, it should be denoted as 1, 2 or

3, respectively.

**Step 4:** Solving local optimal alignment

Starting with the largest element $d_{ij}$ of scoring matrix D, search for the local optimal alignment through reverse backtracking according to the path generated in Step 3 until reaching an element whose value is zero. During the backtracking process, if $d_{ij}$ comes from $d_{i-1,j}$, $s_i$ should be compared with "-"; if $d_{ij}$ comes from $d_{i,j-1}$, "-" should be compared with $t_j$; if $d_{ij}$comes from $d_{i-1,j-1}$, $s_i$ should be compared with $t_j$.

## 4. Experiment and results analysis

### *4.1. Accuracy test*

The accuracy of the Opti-SW was tested through a contrastive experiment against the SW algorithm on a single-node platform. Table 1 lists the results of the two algorithms under different query sequences and database sequences. Note that comparison ratio =(comparison score/query sequence length) * 100%.

The input data of the Opti-SW are as follows:

The query sequence (queryFile) length is 2, 4, 8, 16, 32 and 64; the database sequence (dbFile) length is 64, 32, 16, 8, 4 and 2; the number of fragments (splitNum) is 32; The number of tasks (taskNum) is 1; the number of outputs (topK) is 1; the comparison rate of identity is 0.0 (i.e. any sequence of comparison ratios may be outputted).

The input data of the SW algorithm are as follows:

The query sequence (queryFile) length is 2, 4, 8, 16, 32 and 64; the database sequence (dbFile) length is 64, 32, 16, 8, 4 and 2; the number of fragments (splitNum) is 32; The number of tasks (taskNum) is 1.

*Table 1. Accuracy test results*

| Query sequence length | Target sequence length | Comparison ratio | | Is the optimal local contrast of the Sw and Opti-SW output consistent? | Opti-SW Accuracy |
|---|---|---|---|---|---|
| | | SW | Opti-SW | | |
| 2 | 64 | 100% | 100% | √ | 100% |
| 4 | 32 | 75% | 75% | √ | 100% |
| 8 | 16 | 50% | 50% | √ | 100% |
| 16 | 8 | 37.5% | 37.5% | √ | 100% |
| 32 | 4 | 18.75% | 18.75% | √ | 100% |
| 64 | 2 | 9.375% | 9.375% | √ | 100% |

As shown in Table 1, the comparison ratio and the local optimal alignment of the Opti-SW algorithm and the SW algorithm are both consistent with the optimal local alignment under different sequences and comparison ratios. Thanks to the improvements, the Opti-SW outperformed the SW algorithm in the accuracy of sequence alignment.

### *4.2. Performance comparison*

Next, another experiment was conducted in the single-node environment to further compare the performance between the Opti-SW and the SW algorithm. In the experiment, the query sequence has 48 characters, and the database sequence files are 10MB, 20MB, 40MB, 80MB and 160MB, respectively. The experimental results are presented in Figure 7, where T is the mean execution time of each algorithm measured in three tests, and S is the size of the database sequence file.

It can be seen from Figure 7 that the Opti-SW consumed less time than the SW at the database sizes of 10MB~20MB, 20MB~40MB and 40MB~80MB, and the edge was increasingly obvious with the growth in data size.

As the data size increased, the execution time of each algorithm grew slowly rather than exponentially. When the data size fell between 10MB and 80MB, the execution time of the Opti-SW increased at a slower rate than that of the SW algorithm, and the rate difference widened with the growth in data size. When the data size fell between 80MB and 160MB, the execution time of each algorithm grew at an increasing rate, but the difference between the two algorithms remained constant.

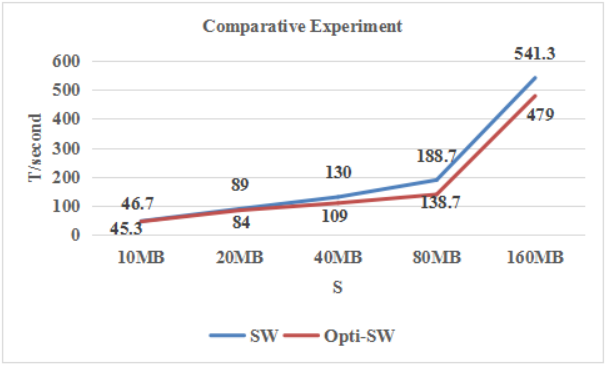


*Figure 7. Comparison between the Opti-SW and the SW algorithm*

In Figure 7, the SW algorithm processed 160MB data in an execution time nearly 2.8 times that needed to process 80MB data, while the Opti-SW processed 160MB data in an execution time almost 3.4 times that needed to process 80MB data. These results show that the Opti-SW still have some problems in handling large-scale gene sequence alignment.

## 5. Conclusions

As a strategy pairwise sequence alignment, the SW algorithm has been widely used in database search thanks to its high accuracy. This paper improves the SW algorithm into the Opti-SW by reducing the computing complexity. The performance of the proposed algorithm was contrasted with the SW algorithm through a number of experiments. The results show that the Opti-SW realizes faster computation than the SW algorithm without sacrificing the alignment accuracy.

*Acknowledgement*

*The work is funded in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61462070, the Doctoral research fund project of Inner Mongolia Agricultural University under Grant No. BJ09-44 and the Inner Mongolia Autonomous Region Key Laboratory of big data research and application for agriculture and animal husbandry.*

## References

Balaji P., Feng W., Archuleta J., Lin H. (2008). Semantics-based distributed I/O for mpiBLAST. *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 293-294. https://doi.org/10.1145/1345206.1345262

Benkrid K., Liu Y., Benkrid A. S. (2009). A highly parameterized and efficient FPGA-based skeleton for pairwise biological sequence alignment. *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 17, No. 4, pp. 561-570. http://dx.doi.org/10.1109/TVLSI.2008.2005314

Cao L., Xu Y., Deng C. (2015). Algorithm for DNA double sequence alignment problem. *Application of Computer System*, Vol. 24, No. 9, pp. 112-117.

Chen M. (2016). Bioinformatics. *Beijing: The Science Publishing Company*, pp. 13-18.

Daily J. (2016). Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC Bioinformatics*, Vol. 17, pp. 124-129. https://doi.org/10.1186/s12859-016-0930-z

Farrar M. (2007). Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, Vol. 23, pp. 156-161. https://doi.org/10.1093/bioinformatics/btl582

Feng B. (2015). Distributed parallel optimization of needleman-wunsch algorithm for double sequence alignment. *Hohhot: Inner Mongolia Agricultural University*, pp. 38-41.

Feng B., Gao J. (2016). Distributed parallel Needleman-Wunsch algorithm on heterogeneous cluster system. *International Conference on Network and Information Systems for Computers.IEEE*, pp. 358-361. https://doi.org/10.1109/ICNISC.2015.145

Gao J., Jiao Y., Zhang W. G. (2014). Review of high throughput sequencing sequence alignment. *Life Science Research*, Vol. 18, No. 5, pp. 458-464.

Jain C., Kumar S. (2014). Fine-grained GPU parallelization of pairwise local sequence alignment. *International Conference on High Performance Computing. IEEE Computer Society*, pp. 1-10. http://dx.doi.org/10.1109/HiPC.2014.7116912

Li D. W. (2011). Research of Parallel algorithm for sequence alignment based on dynamic programming. *Journal of Jinggangshan University (Natural Science Edition)*, Vol. 32, No. 3, pp. 80-84.

Liu Y., Wang X., Li J., Mao Y., Zhao D. (2012). Research progress of local sequence alignment algorithm and parallel acceleration. *Military Medicine*, Vol. 36, No. 7, pp. 556-560. Http://dx.chinadoi.cn/10.3969/j.issn.1674-9960.2012.07.018

Liu Y., Wirawan A., Schmidt B. (2013). CUDASW++3.0: Accelerating Smith–Waterman protein database search by coupling CPU and GPU SIMD instructions. *BMC Bioinformatics*, Vol. 14, pp. 117. https://doi.org/10.1186/1471-2105-14-117

Wang C., Li X., Chen P., Wang A., Zhou X., Yu H. (2015). Heterogeneouscloud framework for big data genome sequencing. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 12, pp. 166-178. https://doi.org/10.1109/TCBB.2014.2351800

Wei G., Ma C., Pei S., Wu B. (2009). The accelerating implementation of BLAST with stream processor. *IEEE, International Conference on Computer-Aided Industrial Design & Conceptual Design, Caid & Cd. IEEE,* pp. 2245-2250. http://dx.doi.org/10.1109/CAIDCD.2009.5375228

Xu B., Li C., Zhuang H., Wang J., Wang Q., Zhou X. (2017). Efficient distributed Smith-Waterman algorithm based on apache spark. *IEEE International Conference on Cloud Computing*, pp. 608-615. http://dx.doi.org/10.1109/CLOUD.2017.83

Xue Q. F. (2015). Research and application of parallel algorithm for DNA sequence alignment. *Shanghai: Shanghai University*, pp. 7-11.

Zhang D. Y. (2015). Bioinformatics. *Beijing: The Science Publishing Company*, pp. 28-43.

Zhao M., Lee W. P., Garrison E. P., Marth G. T. (2013). SSW library: An SIMD Smith-Waterman C/C++ library for use in genomic applications. *Plo S one*, Vol. 8, pp. 2138-2142. https://doi.org/10.1371/journal.pone.0082138