I.J. Engineering and Manufacturing **2011, 4, 32-38**
Published Online August 2011 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijem.2011.04.05

Available online at http://www.mecs-press.net/ijem

Efficient Parallel Design for Edit distance algorithm in DNA Sequence Alignment

Xu Li, Zhenzhou Ji

Department of Computer Science and Engineering, Harbin Institute of Technology, 150001 Harbin, China

Abstract

The focus of Bioinformatics research is usually on two aspects—genomics and proteomics, specifically, it's starting from nucleic acid and protein sequences, analyzing the structural and functional biological information expressed in the sequences. Biological sequence alignment is one of the common problems, the Needleman-Wunsch algorithm based on dynamic programming is the most basic algorithm, and Edit Distance(Levenshtein Distance) algorithm is also widely used in DNA sequence alignment. Nowadays, there are large amount of improvements on the Needleman-Wunsch algorithm, while few on Edit Distance algorithm, so this paper focuses on revealing the effects of parallel design on optimizing the Edit Distance algorithm, and it also compares the two algorithms' different significances in DNA sequence alignment objectively.

Index Terms: Bioinformatics; sequence alignment; Needleman-Wunsch; Edit Distance; parallel design

1. Introduction

The early study of biology is the sequence alignment[1][2], modern biology considers that the constantly changing of the nucleic acid and protein sequences forms the foundation of the biological evolution. We can study the similarities among the modern molecular sequence to derive the process of evolution. Similarity[3] refers to the ratio of the same nucleotide or amino acid sequences between the source sequence and target sequence during the sequence alignment. Homology[3] means the sequences are formed through a divergent evolution from a same ancestor. We use the sequence alignment to find the similarities, and then to study the homology of sequences.

Double sequence alignment contains global alignment[1] and local alignment[1].The typical algorithm of global alignment is the Needleman-Wunsch algorithm based on dynamic programming, it fits the research of two sequences that have a high degree of similarity at the global level; The basic algorithm of local alignment is the Smith-Waterman algorithm, it's for the study of two sequences that are in low degree of genetic relationship and do not have similarities on the whole but in some small areas there exists similarities. In this paper we do not focus on the local alignment.

Dynamic programming algorithm[4] is dividing a problem into several smaller problems, and getting the results of the smaller problems to construct the final result of the big problem. Dynamic programming fits for the case that the sub-problems are not independent, which means containing public sub-problems. Dynamic programming algorithm is often applied to optimization problems, such problems may have many feasible solutions, each solution has a value, and we hope to find an optimal (maximum or minimum) value of the solution.

When we use dynamic programming algorithm in DNA sequence alignment, its core idea is to use iterative methods to calculate the similarity score of two sequences and to store the scores in a scoring matrix, based on the matrix we can turn back to find the optimal sequence alignment. We hope to find the best match to get the bioinformatics information contained in the sequences, and then it can provide the optimal result for our further study.

Edit Distance algorithm[5] is also based on dynamic programming algorithm, it is widely used in the comparison of strings, but we can take DNA sequences as strings made up of bases, it has a strong applicability to the sequences that having a high degree of similarity at the global level, and we can improve the efficiency by using parallel design.

2. Edit Distance algorithm

2.1. The definition of Edit Distance algorithm

Edit Distance algorithm can be also called Levenshtein Distance algorithm, it was put forward by Russian scientist Vladimir Levenshtein in 1965, its definition is as follows:

For the given two strings X and Y, the edit distance between X and Y-$D(X,Y)$is defined as the minimum required number of editing operations using the following three editing operations to convert string X to Y: ① delete a character from string X(or Y); ②insert a character into string X(or Y); ③replace a specified character in string X(or Y).

$D(X,Y)$ also has the following properties: ① Non-negative: $D(X,Y) \geq 0$, $D(X,Y) = 0$;②Symmetry: $D(X,Y) = D(Y,X)$.

2.2. The process of Edit Distance algorithm

In the global alignment of DNA sequence, as there is a high degree of similarity at the global level, we can take the DNA sequences as strings, and then we can use the Edit Distance algorithm to study the similarity, we can also turn back to get the process of the matching[6].

Similar with the Needleman-Wunsch algorithm, we have to construct a matrix to record the alignment of the two DNA sequences, the two DNA sequences are taken as the two-dimensional of the matrix, but different from the Needleman-Wunsch algorithm, the values in the matrix are not the similarities of the sequences but represent the number required to convert string X to Y.

The score function of Needleman-Wunsch algorithm[7] is as (1), the values of the matrix are calculated according to (2), while x and y represent the character in the sequences, S and T represent the given DNA sequences, $M(i, j)$ means the value of the matrix.

$$\delta(x,y)=\begin{cases}2 & x=y\,\&\&\,x\neq''\,\&\&\,y\neq'' \\ -1 & x\neq y\,\&\&\,x\neq''\,\&\&\,y\neq'' \\ -1 & x\neq y\,\&\&\,x=''\,\|\,y=''\end{cases} \tag{1}$$

$$M(i,j)=\max\begin{cases}(M(i-1,j-1)+\delta(T[i],S[j])) \\ (M(i,j-1)+\delta('',S[j])) \\ (M(i-1,j)+\delta(T[i],''))\end{cases} \tag{2}$$

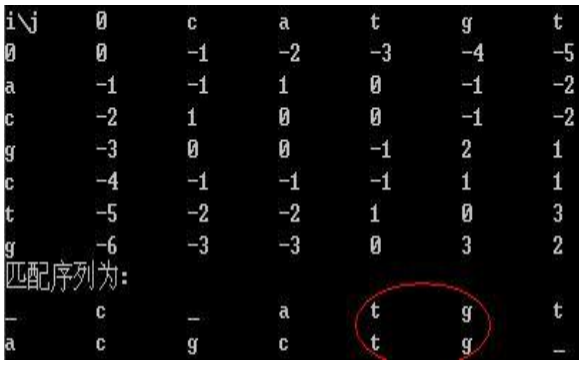
While in the Edit Distance algorithm we use (3) as score function, and we use (4) to calculate the values of the matrix.

$$d(x,y)=\begin{cases}0 & x=y \\ 1 & x\neq y\end{cases} \tag{3}$$

$$M(i,j)=\min\begin{cases}M(i-1,j-1)+d(T[i],S[j]) \\ M(i,j-1)+1 \\ M(i-1,j)+1\end{cases} \tag{4}$$

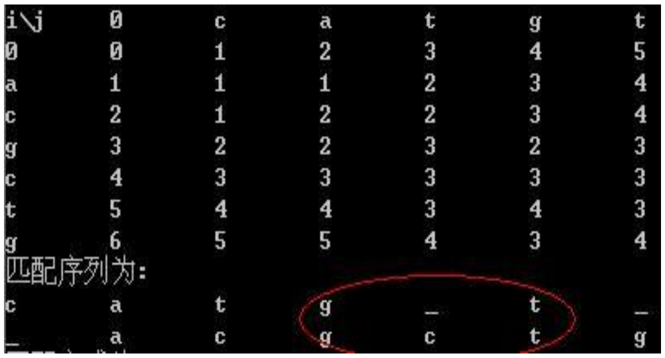
2.3. Comparison of the two algorithms

Take the strings S=catgt and T=acgctg as an example, the results are shown in Fig. 1 and Fig. 2



```
i\j    0     c     a     t     g     t
0      0     -1    -2    -3    -4    -5
a      -1    -1    1     0     -1    -2
c      -2    1     0     0     -1    -2
g      -3    0     0     -1    2     1
c      -4    -1    -1    -1    1     1
t      -5    -2    -2    1     0     3
g      -6    -3    -3    0     3     2
匹配序列为:
_      c     _     a     t     g     t
a      c     g     c     t     g     _
```

Fig. 1 Needleman-Wunsch algorithm



```
i\j     0     c     a     t     g     t
0       0     1     2     3     4     5
a       1     1     1     2     3     4
c       2     1     2     2     3     4
g       3     2     2     3     2     3
c       4     3     3     3     3     3
t       5     4     4     3     4     3
g       6     5     5     4     3     4
匹配序列为:
c       a     t     g     _     t     _
_       a     c     g     c     t     g
```

Fig. 2 Edit Distance algorithm

We can see that compared to the Needleman-Wunsch algorithms, edit distance algorithm destroy the original features of the DNA sequences larger, we should improve the algorithm.

3. Parallel design for Edit Distance algorithm with k-differences

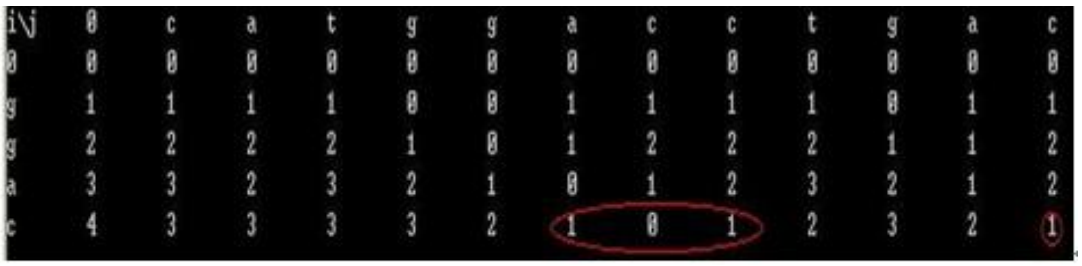
3.1. The principle of Edit Distance algorithm with k-differences

In order to prevent the damage to the original features of the DNA sequences, we introduce the Edit Distance algorithm with k-differences[4].It means for any given newly discovered DNA sequence of length m and source DNA sequence of length n, m<n, give a positive integer k, $0 \le k < m$, to find the end location j with the edit distance less or equal to k letting the newly discovered DNA sequence match the source sequence, $1 \le j \le n$.

The values of the k-differences edit distance matrix are calculated according to (5).

$$M(i,j)=\begin{cases} 0, i=0 \\ i, j=0 \\ \min\begin{cases} (M(i,j-1)+1) \\ (M(i-1,j)+1) \\ M(i-1,j-1)+C \end{cases} \\ C=\begin{cases} 0, T[i]=S[j] \\ 1, else \end{cases} \end{cases} \quad (5)$$

Take the source string S=catggacctgac and newly discovered string T=ggac, k=1 as an example, the results are shown in Fig. 3.



```
i\j  0  c  a  t  g  g  a  c  c  t  g  a  c
0    0  0  0  0  0  0  0  0  0  0  0  0  0
g    1  1  1  1  0  0  1  1  1  1  0  1  1
g    2  2  2  2  1  0  1  2  2  2  1  1  2
a    3  3  2  3  2  1  0  1  2  3  2  1  2
c    4  3  3  3  3  2  1  0  1  2  3  2  1
```

Fig. 3 Edit Distance algorithm with k-differences

The number circled by the red color means the newly discovered DNA sequence matching the source sequence with the edit distance less or equal to 1.We can change the positive integer k to get an acceptable similarity to study the relationship between the DNA sequences. This method can also be used in local alignment.

3.2. Efficient parallel design for Edit Distance algorithm with k-differences

The time complexity of Edit Distance algorithm with k-differences is still $O(mn)$, in order to improve the efficiency, we use the parallel design, but from the principle of recursion it's easy to find the calculation of the value $M(i, j)$ in the edit distance matrix is strictly dependent on $M(i-1, j-1)$, $M(i-1, j)$, $M(i, j-1)$, usually the calculations are linear, but when calculating $M(i, j)$, we can observe the "anti-diagonal", if the values of $M(i-1, j-1)$, $M(i-1, j)$, $M(i, j-1)$ are known, we can get the value of $M(i, j)$. What is more favorable, the values of $M(0, j)$, $M(i,0)$ are known under initial condition, it's convenient for us to calculate the value of $M(i, j)$ on the "anti-diagonal", then we can use parallel design to let the calculation of edit distance advance along the lower right corner direction as shown in Fig. 4, and the red numbers indicate the values are computing at the same time each parallel computing.

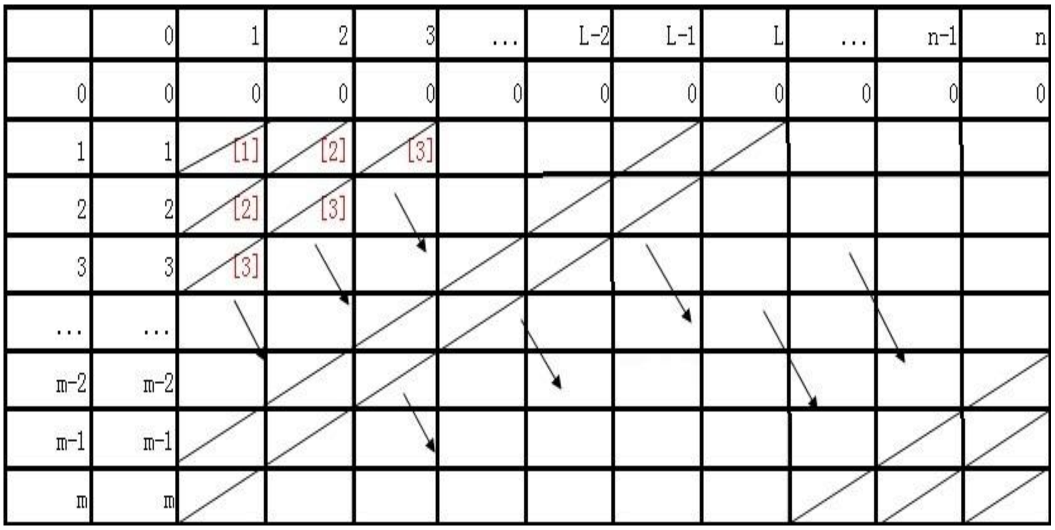


Fig. 4 Parallel advancing

We can find that there is only one matrix element on the first "anti-diagonal", the initial value is $M(0,0)$;The elements on the second "anti-diagonal" is $M(1,0), M(0,1)$,the values are known; Any $M(i,j)$ on the third "anti-diagonal", the values of $M(i-1,j-1)$, $M(i-1,j)$, $M(i,j-1)$ are known so we can parallel computing the values on this "anti-diagonal".

Analysis shows there are n+m+1 "anti-diagonals" and the max number of elements on any anti-diagonal is m+1.

In order to reduce storage space, we use three vectors named L, PL, PP to store values of the elements on the Lth, PLth, PPth "anti-diagonals" with the size m+1, the parallel algorithm is as follows:

① initialize the vectors' values, and PL[0] is constant 1.

② from L=2 to n+m, calculate the number of the elements on the Lth "anti-diagonals", there are three conditions : $L \le m$, $m < L \le n$, $L > n$

③ parallel computing the values of the elements on the Lth "anti-diagonals", according to (6).

$$\begin{cases} L \le m, \quad L[i] \leftarrow \min\begin{cases}(PL[i-1]+1)\\(PL[i]+1)\\(PP[i-1]+C)\end{cases} \\ L = m+1, L[i-1] \leftarrow \min\begin{cases}(PL[i-1]+1)\\(PL[i]+1)\\(PP[i-1]+C)\end{cases} \\ L > m+1, L[i-1] \leftarrow \min\begin{cases}(PL[i-1])\\(PL[i]+1)\\(PP[i]+C)\end{cases} \end{cases} \tag{6}$$

④parallel advance : $PP[i] \leftarrow PL[i]; PL[i] \leftarrow L[i]$

⑤output the result.

While parallel computing we detect the fact that several processors need to read the same data at the same time ,so we can use the PRAM-CREW(parallel random access machine with concurrent read and exclusive write) model[8], because under this model concurrent read conflict can be dealt with in a short time.

4. Performance Prediction

By analyzing we know that the time complexities of Needleman-Wunsch algorithm and Edit Distance algorithm are $O(mn)$, the running times are almost same, but under SIMD-CREW(Single Instruction Multiple Data)computing model with m+1 processors the Edit Distance algorithm with k-differences need n+m+1 iterations, the time complexity is $O(m+n)=O(n)$, implementation costs is $O(mn)$, the speedup is $S_p(n)=O(mn)/O(n)=O(m)$, and the efficiency is $E_p(n)=O(m)/O(m)=O(1)$, there is an obvious improvement.

5. conclusion

In this paper, we study the double sequences alignment algorithms in Bioinformatics, dynamic programming-based Edit Distance algorithm is applied to sequence alignment, and we compared the Needleman-Wunsch algorithm with the traditional Edit Distance algorithm, based on this situation we put forward the Edit Distance algorithm with k-differences to save time and improve efficiency. After the parallel optimization, the Edit Distance algorithm with k-differences can also be used in local alignment and it prevents the damage to the original features of the DNA sequences, so the algorithm has a high significance in application. We can also put it into the use of protein sequence alignment and get further improvement.

References

[1] Zhongneng Xu, "Bioinformatics" ,Tsinghua University Press, Sep 2008,pp.134-137,150-164(in Chinese).
[2] Yangde Zhang, "Bioinformatics", Science Press, Jan 2009(in Chinese).
[3] T.K Attwood, D.J Parry-Smith, "Introduction to Bioinformatics ",Peking University Press, Apr 2002(in Chinese).
[4] T.H Cormen, "Introduction to Algorithms", China Machine Press, Jan 2008,pp.192-221.
[5] Guolinag Chen, "Design and Analysis of Parallel Algorithm," Higher Education Press, Aug 2009,pp.314-320(in Chinese).
[6] N.C Jones, P.A Pevzner, "An Introduction to Bioinformatics Algorithms" ,Chemical Industry Press, Jul 2007,pp.118-146(in Chinese).
[7] Lilan Tu, "Biological Sequence Alignment Based on Fast Walsh Transform", Hua zhong University of Science and Technology, Wuhan, 430074, P.R.China, Apr 2004,pp17-21(in Chinese).
[8] Kai Huang, Zhiwei Xu, "Scalable Parallel Computing Technology, Architecture, Programming", China Machine Press, May 2000, pp.8-18(inChinese).