Journal of Theoretical Biology 263 (2010) 227–236

Contents lists available at ScienceDirect

Journal of Theoretical Biology

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

New method for global alignment of 2 DNA sequences by the tree data structure

Zhao-Hui Qi *, Xiao-Qin Qi, Chen-Chen Liu

School of Computer and Information Engineering, Shijiazhuang Railway Institute, Shijiazhuang, Hebei 050043, People's Republic of China

ARTICLE INFO

Article history:
Received 5 August 2009
Received in revised form
4 December 2009
Accepted 4 December 2009
Available online 16 December 2009

Keywords:
Scoring curve
Gaps
Alignment tree
Post-order traversal
Global alignment

ABSTRACT

We introduce a new approach to investigate problem of DNA sequence alignment. The method consists of three parts: (i) simple alignment algorithm, (ii) extension algorithm for largest common substring, (iii) graphical simple alignment tree (GSA tree). The approach firstly obtains a graphical representation of scores of DNA sequences by the scoring equation $R_0 * R - S_0 * S - T_0 * (a + bk)$. Then a GSA tree is constructed to facilitate solving the problem for global alignment of 2 DNA sequences. Finally we give several practical examples to illustrate the utility and practicality of the approach.

Crown Copyright © 2009 Published by Elsevier Ltd. All rights reserved.

1. Introduction

The decoding of different genomes, in particular the human genomes has triggered a great deal of bioinformatics research. The research of sequence similarity to a known protein sequence or DNA sequence has been an important method to provide the first clues about the function of a newly sequenced gene. The research becomes increasingly useful in the analysis of newly sequenced genes as the sequence databases, such as DNA and protein databases, continue to grow in size. There are a number of standard schemes widely used to search for homologous sequences in nucleotide and protein databases to distinguish biologically significant relationships from chance similarities (Smith and Waterman, 1981; Waterman, 1984; Pearson and Lipman, 1988; Altschul et al., 1990, 1997; Tatiana and Thomas, 1999). The dynamic programming algorithms (Smith and Waterman, 1981; Waterman, 1984) assign scores to insertions, deletions and replacement, and compute an alignment of two sequences. These algorithms are impractical for searching large database because of their computational requirements. Rapid heuristic algorithms (Pearson and Lipman, 1988; Altschul et al., 1990, 1997) compare protein and DNA sequences much faster than the above methods. They are widely used for large database searches. However, a number of important scientific contexts involve the comparison of only two sequences and do not require a time-consuming database search. In order to meet these needs, many important tools for large database searches (Tatiana and Thomas, 1999; http://www.ebi.ac.uk/Tools/emboss/align/index.html; http://blast.ncbi.nlm.nih.gov/bl2seq/wblast2.cgi) are further developed to employ global alignment of 2 protein and nucleotide sequences. These methods can provide the global profile of similarity degree. Recently, many graphical methods have also been used to examine the global similarities/dissimilarities among the coding sequences of different species (Qi et al., 2007; Qi and Qi, 2007, 2009; Qi and Fan, 2007; Yao et al., 2006, 2008a, b; Randić et al., 2003a, b). Because of the advantages in visualization, graphical methods have become a powerful bioinformatics tool for the analysis of complicated biological systems, such as enzyme-catalyzed system (Chou et al., 1979; Chou, 1980; Chou and Forsen, 1980, 1981; Chou and Liu, 1981; Myers and Palmer, 1985; Zhou and Deng, 1984; Chou, 1989, 1990; Kuzmic et al., 1992; Andraos, 2008), protein folding kinetics (Chou, 1990, 1993), condon usage (Chou and Zhang, 1992; Zhang and Chou, 1994), HIV reverse transcriptase inhibition mechanisms (see Althaus et al., 1993a, b, c, as well as a review article (Chou et al., 1994)), base frequency distribution in the anti-sense strands (Chou et al., 1996) and classifying organisms (Sorimachi and Okayasu, 2008; Okayasu and Sorimachi, 2009; Qi et al., 2009). Recently, the images of cellular automata were used to represent biological sequences (Xiao et al., 2005a), predict protein subcellular location (Xiao et al., 2006a), investigate HBV virus gene missense mutation (Xiao et al., 2005b) and HBV viral infections (Xiao et al., 2006b), predicting protein structural classes (Xiao et al., 2008) and G-protein-coupled receptor functional classes

* Corresponding author.
E-mail address: zhqi_yh2004@yahoo.com.cn (Z.-H. Qi).

0022-5193/$ - see front matter Crown Copyright © 2009 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.jtbi.2009.12.012

(Xiao et al., 2009), as well as analyze the fingerprint of SARS coronavirus (Wang et al., 2005; Gao et al., 2006).

In this paper, we suggest a heuristic approach to align a pair of DNA sequences with the tree data structure. Some graphical descriptions are also used to intuitively explain the scheme. The method consists of three parts: (i) simple alignment algorithm, (ii) extension algorithm for largest common substring, (iii) graphical simple alignment tree (GSA tree). Here, we firstly obtain a 2-dimension (2D) graphical curve by graphical representation of scores of DNA sequences. A good simple alignment of the DNA sequences is generated when the score of the scoring curve reaches its peak value. The 2D graphical curve can intuitively show the global change of simple alignment scores based on scoring matrix. Then the largest common substrings of the good simple alignment are found out. Because of the limit of the initial alignment reaching peak score, the largest common substrings of original two sequences may be split by the largest common substrings of the good simple alignment. In order to protect these common substrings from being split, an extension algorithm for the largest common substring is suggested. Then all largest common substrings are found out when the initial alignment reaches peak score. The process is repeatedly done until GSA tree comes into being. The tree can facilitate solving the problem for global alignment of 2 DNA sequences. At last, we give several practical examples to illustrate the utility and practicality of the approach.

2. Simple alignment algorithm and graphical representation of scores of DNA sequences based on scoring matrix

2.1. The scoring equation based on scoring matrix

In bioinformatics, scoring matrix is also called as substitution matrix. As for protein sequences, the matrix is often based on observed substitution rates, derived from the substitution frequencies seen in multiple alignments of sequences. Every possible identity and substitution is assigned a score based on the observed frequencies in alignments of related proteins. Similarly, every possible identity and substitution in alignments of related DNA sequences is also assigned a score. However, the scoring matrix in alignment of DNA sequences is relatively simple and intuitional. Table 1 is a scoring matrix. An identity in scoring matrix is assigned a positive score R ($R>0$). A substitution is assigned a positive score S ($S>0$). But the score S must be subtracted from the total score of an alignment. For example, an alignment of two short DNA sequences ATGGTGCAACTGACT and ATGGTGCACTTGACT is the following:

```
ATGGTGCAACTGACT
|||||||| |||||
ATGGTGCACTTGACT
```

The score of alignment should be $13R-2S$.

Another important problem for alignment is the treatment of gaps, i.e., spaces inserted to optimize the alignment score. A 'gap open' penalty is one that is the cost for the first space of each gap spaces. A 'gap extension' penalty is one that is the cost for one of each gap spaces except for the first space. Typically, the cost of extending a gap is set to be 5–10 times lower than the cost for opening a gap (http://www.ebi.ac.uk/Tools/emboss/align/index.html). There is one way to compute a penalty for a gap of n positions: gap opening penalty + $(n-1)^*$ gap extension penalty. Now, let parameter a be gap opening penalty and parameter b be gap extension penalty. And let parameter k be $n-1$. Then we have the scoring equation: $R_0*R-S_0*S-T_0*(a+bk)$, where R is the score of each match, and S is the score of each mismatch and $a+bk$ is the score of each gap. The parameters R_0, S_0 and T_0 denote the total amount of matches, the total amount of mismatches and the total amount of gaps, respectively.

Table 1
A usual scoring matrix.

	A	C	G	T
A	R	−S	−S	−S
C	−S	R	−S	−S
G	−S	−S	R	−S
T	−S	−S	−S	R

2.2. Simple alignment algorithm and graphical representation of scores of DNA sequences based on scoring matrix

There are two primary DNA sequences: G_1 ($g_1g_2\cdots g_M$) and G_2 ($g_1g_2\cdots g_N$), where M and N denote the length of G_1 and G_2, respectively. Fig. 1 shows the building steps of all possible simple alignments without spaces within G_1 and G_2. The gap formed by spaces lies in the hanging ends of the overlap. Step 1 of Fig. 1 gives the initial alignment that the last base of G_1 overlaps the first base of G_2. Then every time G_1 moves one base position along the direction of G_2. Step $M+N-1$ of Fig. 1 gives the last alignment that the first base of G_1 overlaps the last base of G_2. Every alignment has a score according to the scoring equation $R_0*R-S_0*S-T_0*(a+bk)$. Then we can obtain a serial of dots (x,y), where x denotes index of steps and y denotes the score value corresponding to x. When one connects adjacent dots with lines, then one obtains a zigzag like curve of definite geometrical shape. In Fig. 2 we illustrate the graphical score representation of simple alignments of sequences G_1 and G_2 (G_1, GGCCTCTGCCTAATCACACAGATCTAACAGGATTATTTC; G_2, GGCCTCT GCCTTATTACACAAATCTTAACAGGACTATTTC). The scoring equation is $R_0*R-S_0*S-T_0*(a+bk)$, where identity score R is 9, substitution score S is 1, gap opening penalty a is 15, and gap extension penalty b is 1.

The values about the parameters in the above scoring equation are chosen according to a choice of 'EMBOSS Pairwise Alignment Algorithms-needle' (http://www.ebi.ac.uk/Tools/emboss/align/index.html). It is well known that changing the values of the parameters may change the number and length of gaps in an alignment. However, there is no analytical formula that determines the 'best' gap values to use, so that one may wish to experiment with values in order to explore more of the alignment 'space' (Tatiana and Thomas, 1999). As for needle, one can experiment with different combinations of parameters. Here, we choose some typical values of parameters, such as $R=9$, $S=1$, $a=15$ and $b=1$. Of course, one can choose different values by his

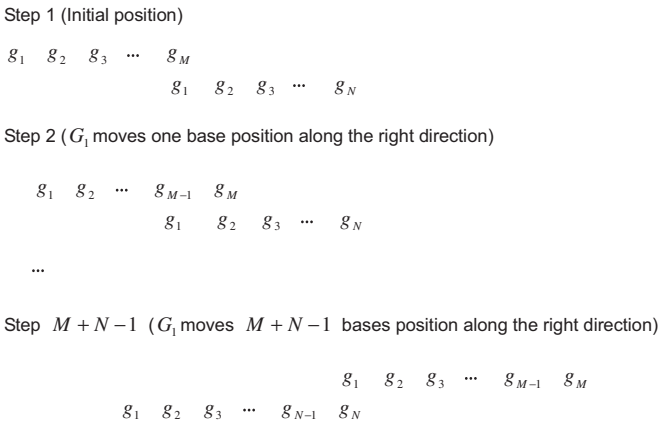


Fig. 1. The building steps of all possible alignments without spaces within G_1 and G_2.

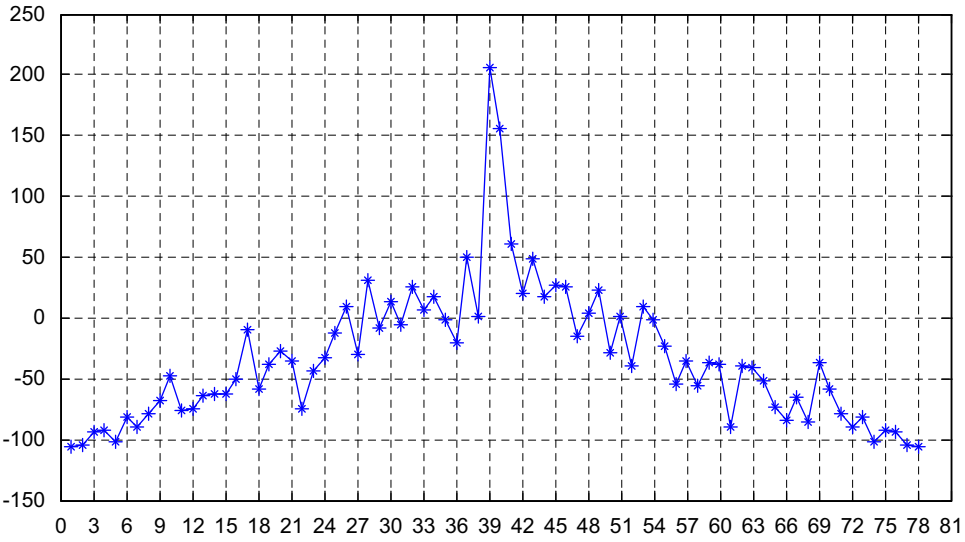


Fig. 2. The figure illustrates the graphical score representation of simple alignments of sequences G_1 and G_2. The scoring equation is $\sum R - \sum S - \sum(a+bk)$, where identity score R is 9, substitution score S is 1, gap opening penalty a is 15, and gap extension penalty b is 1.

experiment. In this paper, we choose these values anywhere in order to maintain consistency in the context.

Fig. 2 clearly shows the graphical 'signatures' of simple alignments of sequences G_1 and G_2. Obviously, graphical 'signatures' enable much easier visual inspection of simple alignments of DNA sequences than their representation by strings over the DNA alphabet {A, T, G, C}. A close look at Fig. 2, one can easily find out that the score of the simple alignment reaches its peak score when the index is 39. In this paper, we call the simple alignment with peak score as a good simple alignment of sequences G_1 and G_2.

The 2D scoring curve can intuitively show the global change of simple alignment scores based on scoring matrix. One can easily obtain the peak point by observing the curve. Of course, it is not necessary to draw the graphical scoring curve when one dose not want to observe the global change of simple alignment scores or need deal with thousands of pairs of DNA sequences. He can also determine the peak point by doing data comparison.

2.3. Improved simple alignment algorithm with less consuming time

The above simple alignment algorithm shown in Fig. 1 is a time-consuming algorithm. Its time complexity is $O(N^2)$ if the two aligned sequences have equal length N. Here, we provide an improved alignment process to obtain a lower time complexity. A close look at Fig. 1 shows that the beginning steps and the ending steps of the sliding process are not necessary. These unnecessary steps consume some time. Now, we need to find out the unnecessary steps to save time.

Fig. 3 shows the improved process. Step (1) of Fig. 3 gives the initial alignment that the first base of G_1 overlaps the first base of G_2. Then the sliding process is done along the left and the right, respectively. Firstly, we consider the sliding process is done along the right, as shown in step (2). Every alignment has a score S according to the scoring equation $R_0 * R - S_0 * S - T_0 * (a+bk)$. Let score S_i be the score of the ith step. Let $S_{\max}$ be the top score within all the scores (i.e., $S_{\max} = \max\{S_1, S_2, \ldots, S_{i-1}\}$, $i > 1$). As for every step, we still define another score S'_i. The score S'_i is obtained by another scoring equation $R_0 * R + S_0 * R - T_0 * (a+bk)$. In the new scoring equation we assume all overlapped bases are matched each other. The sliding process is stopped when $S_{\max} \geq S'_i$. Obviously, the remaining steps along the right are unnecessary because all scores of the remaining steps have no chance to obtain a higher score than $S_{\max}$.

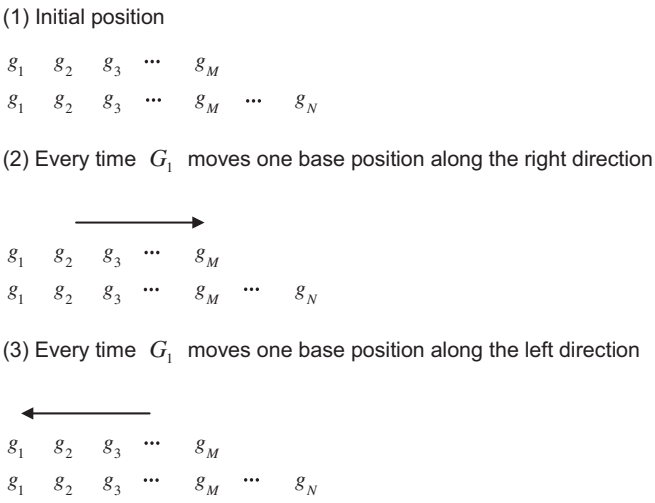


Fig. 3. Improved simple alignment process with less consuming time (The length of G_1 is M. The length of G_2 is N. And let $M \leq N$).

Then we consider the sliding process is done along the left. The top score within all the scores $S'_{\max}$ is $\max\{S_{\max}, S_1, S_2, \ldots, S_{j-1}\}$ $(j > 1)$. The score S'_j is obtained by the scoring equation $R_0 * R + S_0 * S - T_0 * (a+bk)$. The sliding process is stopped when $S'_{\max} \geq S'_j$. The remaining steps along the left are unnecessary.

By the above sliding process the improved simple alignment algorithm can save some time. Now we discuss the time complexity. Let N and M be the length of sequences G_1 and G_2, respectively. Let k be the total amount of the sliding steps when the sliding process is stopped. Then we can draw a conclusion that the time complexity is $O(k * M)$ if $M \leq N$. Obviously, the more two sequences are similar, the smaller the parameter k is. Especially, the time complexity is $O(N)$ if $G_1 = G_2$.

3. Methods

3.1. Extension algorithm for the largest common substring

By the simple alignment algorithm a good simple alignment R of G_1 and G_2 is determined when the score of the simple

alignment reaches its peak. Let C be the largest common substrings of R, where $C=\phi\cup\{C_1,C_2,\ldots,C_m\}$. If $C=\phi$, there is no largest common substring. When $C=\{C_1,C_2,\ldots,C_m\}$, there are m largest common substrings, where $|C_1|=|C_2|=\cdots=|C_m|=k$ and k denotes the number of matches within a largest common substring. Here, we can see that the largest common substring C_i $(i=1,2,\ldots,m)$ of R devotes its maximal score to the initial alignment reaching peak score. The alignment shows the approximate overall alignment feature of G_1 and G_2. However, some C_i of the largest common substrings of R may be a part of a larger common substring than C_i because of the limit of the initial alignment reaching peak score. In order to protect a larger common substring than C_i from being split by C_i and find out the substring, an extension algorithm for the largest common substring is suggested as the following.

(1) Let K be the length of the largest common substring of G_1 and G_2. Its value is generated when the simple alignment algorithm applies to the sequences G_1 and G_2.
(2) When $k=K$, none of the largest common substrings C_i of R can be extended into larger common substring. The largest common substrings of R are, $C_1,C_2,\ldots C_m$.
(3) When $k<K$, there exists at least a larger common substring than C_i. There are several sub-steps to find out the larger common substrings as the following:
   (a) Let L_L be the number of mismatches from the right end of C_{i-1} to the left end of C_i. When $i=1$, L_L denotes the number of mismatches from the left end of G_1 and G_2 to the left end of C_1. Similarly, let L_R be the number of mismatches from the right end of C_i to the left end of C_{i+1}. When $i=1$, L_R denotes the number of mismatches from the right end of C_1 to the right end of G_1 and G_2.
   (b) When $K<L_L$, the K mismatches are extracted from the left of C_i. Otherwise, the L_L mismatches are extracted from the left of C_i. Similarly, when $K<L_R$, the K mismatches are extracted from the right of C_i. Otherwise, the L_R mismatches are extracted from the right of C_i. Then the sequences extracted from the left of C_i, C_i and from the right of C_i are connected into two new sub-sequences S_i^1 and S_i^2.
   (c) Now, we apply the simple alignment algorithm to S_i^1 and S_i^2. If there exist a new larger common substring within S_i^1 and S_i^2 than C_i, we will face a choice: the new larger common substring or C_i. If there is an increment of sore when the new larger common substring comes into being, we will replace the original C_i with the new substring also called as C_i.
(4) As for every C_i of R, the original C_i is replaced by the new largest common substring if the new substring exists.

Now, we give a practical example to illustrate the above steps for a better understanding. There are two random sequences: G_1 (GCCTAGTTCCCCCA) and G_2 (GCCTCGCATCCCCCA). By the simple alignment algorithm a good simple alignment R of G_1 and G_2 is determined, where the alignment R is

GCCTAGTTCCCCCA
GCCTCGCATCCCCCA.

The largest common substring C of R is "GCCT" (C_1) and "CCCC" (C_2). However, the largest common substring of G_1 and G_2 is "TCCCCCA". Its length K is 7. Obviously, there exists at least a larger common substring than "GCCT" or "CCCC". As for "GCCT", the two new sub-sequences are "GCCTAGTTC" (S_1^1) and "GCCTCGCAT" (S_1^2), respectively. Because there is no increment of sore, the original common substring "GCCT" (C_1) is unchanged. As for "CCCC", the two new sub-sequences are "AGTTCCCCCA" (S_2^1) and "CGCATCCCCCA" (S_2^2), respectively. Because there is an increment of sore, the original common substring "CCCC" (C_2) is replaced with the new largest common substring "TCCCCCA".

In sum, the extension algorithm for largest common substring protects a larger common substring than C_i from being split by C_i.

Now, we let U denote the substrings spaced by C, where $U=\phi\cup\{U_1,U_2,\ldots,U_n\}$. Then the strings G_1 and G_2 are aligned into two types of substrings: (i) The largest common substrings C with continuous matches: $G_1[i]=G_2[i]$ (of course, a single match is also permitted) and (ii) The substrings U with continuous mismatches: $G_1[i]\neq G_2[i]$ and both without spaces. Obviously, a good simple alignment R of G_1 and G_2 is alternately organized by C and U (e.g. $R=C_1^1U_2^1C_3^1U_4^1$, where $|C_1^1|=|C_3^1|$. Here, the superscript of C or U denotes the level of sub-alignment. The superscript "1" denotes the first level. And the subscript denotes the index of substrings in the current sub-alignment. The alignment R is the result of the first level sub-alignment).

In order to expressly explain the above parameters, we give a practical example. There are two random sequences: G_1 (GCCTAGTTCCCCCA) and G_2 (GCCTCGCATCCCCCA). The $G_1[i]$ and $G_2[i]$ denote the base of G_1 and G_2, respectively. They become a match when $G_1[i]=G_2[i]$. By the simple alignment algorithm a good simple alignment R of G_1 and G_2 is determined, where the alignment R is

GCCTAGTTCCCCCA
GCCTCGCATCCCCCA.

And by the extension algorithm the largest common substring C of R is "GCCT" and "TCCCCCA". Then G_1 and G_2 are organized by C into three substrings: C_1^1 is "GCCT", U_2^1 is "AGT" and "CGCA", C_3^1 is "TCCCCCA".

3.2. Graphical simple alignment tree (GSA tree)

Given sequences G_1 and G_2, a graphical simple alignment (GSA) tree is built up by the aforesaid simple alignment algorithm and extension algorithm for largest common substring. In the following, we will show how to use the algorithms to construct a GSA tree of G_1 and G_2.

Let R be a good simple alignment of G_1 and G_2. Let C_i^1 and U_j^1 be the largest common substring of R and be the substring spaced by C_i^1, respectively. We can see that the largest common substring C_i^1 of R devotes its maximal score to the global alignment. However, the U_j^1 of R may provide a increment of the score to the global alignment if given appropriate gaps within U_j^1, though the scores of these U_j^1 are low in the first level sub-alignment.

In order to explore the appropriate gaps in U_j^1, we give the following several operation steps.

(1) Compute the scores of all simple alignments of U_j^1 by the simple alignment algorithm. A good simple alignment R_j^1 of U_j^1 is generated when its score reaches its peak.
(2) If there is a increment of the score due to appropriate gaps within U_j^1, U_j^1 can be further divided into the second level sub-alignment. When and how to add the gaps in the sequences? Now, let C_i^2 be the largest common substrings of R_j^1, where $C_i^2=\phi\cup\{C_{i+1}^2,C_{i+2}^2,\ldots,C_{i+m}^2\}$. Then there are two sub-steps as the following:
   (a) If $C_i^2=\{C_{i+1}^2,C_{i+2}^2,\ldots,C_{i+m}^2\}$, there are m largest common substrings. Then U_j^1 can be further divided into the second level sub-alignment by C_i^2. Let U_j^2 be the substrings spaced

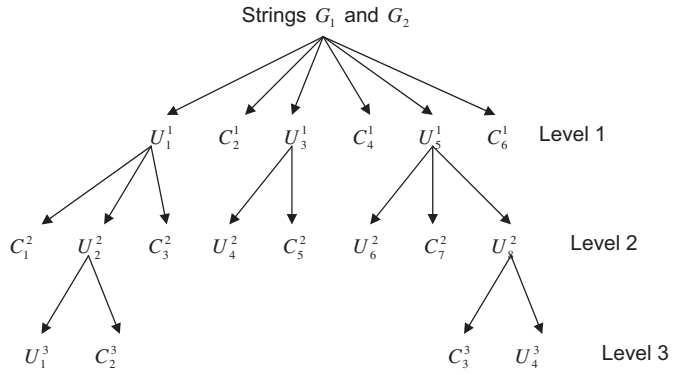


Fig. 4. An example of graphical simple alignment tree (Strings G_1 and G_2 includes 6 substrings in the first level sub-alignment: $U_1^1C_2^1U_3^1C_4^1U_5^1C_6^1$. The substring U_1^1 includes 3 substrings in the second level sub-alignment: $C_1^2U_2^2C_3^2$. The substring U_3^1 includes 2 substrings in the second level sub-alignment: $U_4^2C_5^2$. The substring U_5^1 includes 3 substrings in the second level sub-alignment: $U_6^2C_7^2U_8^2$. The substring U_2^2 includes 2 substrings in the third level sub-alignment: $U_1^3C_2^3$. The substring U_8^2 includes 2 substrings in the third level sub-alignment: $C_3^3U_4^3$).

by C_i^2, where $U_j^2=\phi\cup\{U_{j+1}^2,U_{j+2}^2,\ldots,U_{j+n}^2\}$. Then every substring C_{i+k}^2 $(k=1,2,\ldots,m)$ of C_i^2 becomes a leaf node in GSA tree. There are no gaps within them. As for every substring U_{j+k}^2 $(k=1,2,\ldots,n)$ of U_j^2, the operation flow goes back to the step (1). And the level of sub-alignment enters the next.

(b) If $C_i^2=\phi$, there is no the largest common substrings in R_j^1. Then U_j^1 can not be further broken down. The good simple alignment R_j^1 of U_j^1 becomes a leaf node in GSA tree. The two sequences of R_j^1 might be entirely overlapping, or partially overlapping, or one sequence might be aligned entirely internally to the other. When the two sequences of R_j^1 are entirely overlapping, there are no gaps within R_j^1. Otherwise, the hanging ends of the overlap come into being the gaps of R_j^1. The relative position of these gaps is fixed, and becomes the gaps within the final global alignment.

We repeatedly do the above steps until all U in the last level sub-alignment cannot be further decomposed by the simple alignment algorithm and the extension algorithm. Then we can obtain a graphical simple alignment tree for strings G_1 and G_2, consisting of a series of substrings. The Fig. 4 illustrates an example of GSA tree.

3.3. The global alignment problem based on GSA tree

We can obtain a global alignment of strings G_1 and G_2 when their GSA tree is generated. In the following, we will show how to use GSA tree to generate a global alignment. Observing the GSA tree (e.g. Fig. 4), we can see that there are two types of nodes: inner nodes and leaf nodes. The inner nodes consist of substrings U that can be aligned to more substrings. The leaf nodes include substrings C and U, where U cannot be further aligned by the GSA tree method. The global alignment of strings G_1 and G_2 should be composed of all leaf nodes. In order to obtain the global alignment, the GSA tree is traversed by post-order traversal of tree. Then all inner nodes are deleted from the result of post-order traversal. We will achieve the global alignment of strings G_1 and G_2. For example, the result of post-order traversal of Fig. 4 is

$C_1^2U_1^3C_2^3U_2^2C_3^2U_1^1C_2^1U_4^2C_5^2U_3^1C_4^1U_6^2C_7^2C_3^3U_4^3U_8^2U_5^1C_6^1$.

The global alignment is $C_1^2U_1^3C_2^3C_3^2C_2^1U_4^2C_5^2C_4^1U_6^2C_7^2C_3^3U_4^3C_6^1$ by removing all of inner nodes.

4. Application and discussion

4.1. The application of GSA tree

In this section, we give three examples to see the validity of GSA tree for the global alignment of 2 DNA sequences. As for every example, we compare the results by GSA tree with the results by the important heuristic approach "EMBOSS Pairwise Alignment Algorithms—needle" (http://www.ebi.ac.uk/Tools/emboss/align/index.html).

We firstly apply the proposed method to the discussed 2 sequences: G_1 (GGCCTCTGCCTAATCACACAGATCTAACAGGATTATTTC) and G_2 (GGCC TCTGCCTTATTACACAAATCTTAACAGGACTATTTC). The scoring equation is $R_0*R-S_0*S-T_0*(a+bk)$, where identity score R is 9, substitution score S is 1, gap opening penalty a is 15, and gap extension penalty b is 1. Of course, one may also choose other values as the parameters of the scoring equation according to practical requirement. Here, we choose the same values in order to keep the context consistency. In Fig. 2, we have described the graphical score representation of simple alignments of sequences G_1 and G_2. One can easily find out that the score of the simple alignment reaches its peak when index of step in Fig. 1 is 39. Then we can obtain the first level sub-alignment results according to the simple alignment algorithm and the extension algorithm. By similar rules, we can get all possible sub-alignment results. Fig. 5 shows the GSA tree to be used to align the sequences G_1 and G_2. Table 2 lists all substrings of Fig. 5 and the max score of each pair of substrings. Then we obtain the global alignment of the sequences: $C_1^1U_1^4C_2^4U_3^4C_2^3U_4^4C_5^4U_6^4C_2^2U_4^3C_5^3$. According to the results of Table 2, we illustrate the global alignment results as the following:

```
GGCCTCTGCCTAATCACACAGATC-TAACAGGATTATTTC
|||||||||||  ||  |||||  |||  ||||||||  ||||||
GGCCTCTGCCTTATTACACAAATCTTAACAGGACTATTTC
```

The notation "-" denotes the gap within sequence. Then we apply "needle" program to G_1 and G_2. The values about gap opening penalty a and gap extension penalty b are the same as the values proposed the GSA tree algorithm. The default values about identity score R and substitution score S are chosen because of no other choice. The alignment results are shown as the following:

```
EMBOSS_001     1 GGCCTCTGCCTAATCACACAGATC-TAACAGGATTATTTC    39
                 ||||||||||.||.||||||.|||  ||||||||.|||||||
EMBOSS_001     1 GGCCTCTGCCTTATTACACAAATCTTAACAGGACTATTTC    40
```

Obviously, we get consistent results by two different methods. Moreover, similar results can be found out in Table 1 of Randić et al. (2006).

The aforesaid example about the validity examination gives rise to a question: Is it possible to use the GSA tree in order to facilitate solving less similar or longer DNA sequence alignment problem? The answer is positive. Now, we randomly give two less similar sequences: G_1 and G_2 (G_1, GCCCTCGCGGGCAACATTTAATT-

CACAGCCAGTTCTCTCAACAGTGATTATC; G_2, CTGGGTCTTCAGGTCCTTTATGCTTAACACAAATCTATCGTTA ACAGGACTATTCT). Like the above example, the scoring equation is $R_0 * R - S_0 * S - T_0 * (a + bk)$, where identity score R is 9, substitution score S is 1, gap existence penalty a is 15, and gap extension penalty b is 1. By GSA tree algorithm, we can get the GSA tree and all possible sub-alignment results. Fig. 6 shows the GSA tree to be used to align the sequences G_1 and G_2. Table 3 lists all substrings in Fig. 6 and the max score of each pair of substrings. Then we obtain the global alignment of the sequences:

$U_1^4 C_2^4 U_3^4 C_4^4 U_5^4 C_2^3 U_3^3 C_2^2 U_6^4 C_7^4 U_8^4 C_5^3 U_6^3 C_4^2 U_5^2 C_2^1 U_7^3 C_8^3 U_9^3 C_{10}^3 U_{11}^3 C_7^2 U_9^4 C_{10}^4 U_{11}^4 C_{13}^3 U_{14}^3 C_9^2 U_{10}^2 C_4^1 U_5^1$.

According to Table 3, we illustrate the global alignment results as the following:

```
------GCCCTCGCGGGGCAACATTTAA--TTCACAGCCAGTTCTC-TCAACAGTGATTATC--
      |  | ||    ||    | ||||   || |||    |  | || | |||||  || |||
CTGGGTCTTCA--GGT--CCTTTATGCTTAACACAAATCTATCGTTAACAG-GACTATTCT
```

According to the results, we can obtain some conclusions of the global alignment between the sequences G_1 and G_2: Length: 61; Identities: 31/61 (50.8%); Gaps: 14/61 (22.9%); Score: 169. Then we apply "needle" program to G_1 and G_2. The alignment results are shown as the following:

```
EMBOSS_001   1  GCCCTCGCGGGGCAACA------TTTA--ATTCACAGCCAGTTCTCTC--A  40
                |..||.|..||      ||||  .||.|||  ||..|||.||    |
EMBOSS_001   1  CTGGGTCTTCAGGTCCTTTATGCTTAACA--CAAATCTATCGTTA  43

EMBOSS_001  41  ACAGTGATTATC      52
                ||||  ||.|||.
EMBOSS_001  44  ACAG-GACTATTCT    56.
```

Then we can obtain some conclusions of the global alignment between the sequences G_1 and G_2: Length: 64; Identities: 32/64 (50%); Gaps: 20/64 (31.2%); Score: 29.

Obviously, in this example we get different results by two different methods. A close look at the alignments reveals that there are two different areas (the bold areas illustrate the almost identical ones): the bases from 1 to 20 by GSA tree vs. the bases from 1 to 21 by needle, and the bases from 34 to 46 by GSA tree vs. the bases from 35 to 49 by needle. As for the first area, GSA tree obtains 7 matches by 3 gap openings and 5 gap extensions while needle gets 6 matches by 2 gap openings and 8 gap extensions. In the second area GSA tree gets 5 matches by 1 gap openings and 0 gap extensions while needle obtains 7 matches by 2 gap openings and 3 gap extensions. From the global view, GSA tree receives 31 matches by 7 gap openings and 7 gap extensions while needle gets 32 matches by 7 gap openings and 13 gap extensions. However, it is difficult for us to determine which one is the better when it comes to the best global alignment of the sequences considered. In fact, a close look at the alignments discovers that both of the methods can find out those very similar local areas, such as the bold areas of the alignments.

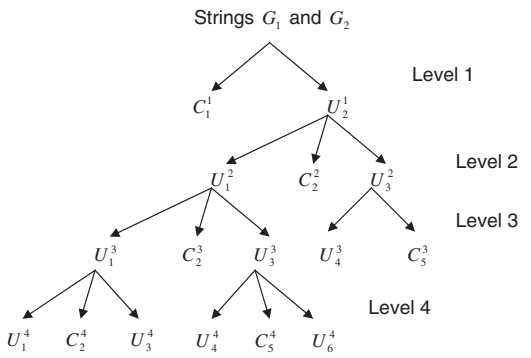


Fig. 5. The graphical simple alignment tree to be used to align the sequences G_1 and G_2 (G_1, GGCCTCTGCCTAATCACACAGATCTAACAGGATTATTTC; G_2, GGCCTCTGCCTTATTACACAAATCTTAACAGGACTATTTC).

Finally, we give the third example with two long and very similar sequences to see the validity of GSA tree for the global alignment of 2 DNA sequences. Here, we apply the proposed method to the complete coding sequence part of beta globin gene of Human (ACCESSION U01317) and Opossum (ACCESSION J03643) as shown in Table 4. The scoring equation is $R_0 * R - S_0 * S - T_0 * (a + bk)$, where identity score R is 9, substitution score S is 1, gap existence penalty a is 15, and gap extension penalty b is 1. For simplification, we do not show the detail of GSA tree and the results of all substrings. In Fig. 7, we illustrate the final result. The notation "-" denotes the gap within sequence. According to the figure, we can obtain some conclusions of the global alignment between string G_1 (the complete coding sequence of ACCESSION U01317) and string G_2 (the complete coding sequence of ACCESSION J03643): Length: 448; Identities: 328/448 (73.2%); Gaps: 8/448 (1.8%); Score: 2772. Then we apply the web tool for "needle" program to G_1 and G_2. The conclusions about the global alignment is the following: Length: 448; Identities: 328/448 (73.2%); Gaps: 8/448 (1.8%); Score: 1128.0. From the global view, the two methods almost get the same statistics results except for their scores. There is only one different area: the bases from 33 to 34 by GSA tree vs. the bases from 33 to 34 by needle. In fact, the two different areas are equivalent to each other. This example shows that it is possible to use the GSA tree in order to facilitate solving long and very similar DNA sequence alignment problem.

4.2. Discussion

The GSA tree algorithm is a gradual algorithm that step by step explores suitable gaps to improve the score of the alignment between 2 DNA sequences. The scheme firstly uses the simple alignment algorithm and the extension algorithm to construct a GSA tree. The substrings denoted by leaf node of GSA tree are looked on as a part of the final global alignment. Obviously, GSA tree method is a heuristic algorithm. There is no analytical

Table 2
All substrings in Fig. 5 and the max score of each substring.

Substrings				The max score
Level 1	G_1, G_2	$C_1^1(G_1)=C_1^1(G_2)$: GGCCTCTGCCT		99
		$U_2^1(G_1)$: AATCACACAGATCTAACAGGATTATTTC		127
		$U_2^1(G_2)$: TATTACACAAATCTTAACAGGACTATTTC		
Level 2	$U_2^1(G_1)$	$U_1^2(G_1)$: AATCACACAGATC		72
	$U_2^1(G_2)$	$U_1^2(G_2)$: TATTACACAAATCT		
		$C_2^2(G_1)=C_2^2(G_2)$: TAACAGGA		72
		$U_3^2(G_1)$: TTATTTC	$U_3^2(G_2)$: CTATTTC	53
Level 3	$U_1^2(G_1)$	$U_1^3(G_1)$: AATC	$U_1^3(G_2)$: TATT	16
	$U_1^2(G_2)$	$C_2^3(G_1)=C_2^3(G_2)$: ACACA		45
		$U_3^3(G_1)$: GATC	$U_3^3(G_2)$: AATCT	11
	$U_3^2(G_1)U_3^2(G_2)$	$U_4^3(G_1)$: T	$U_4^3(G_2)$: C	−1
		$C_5^3(G_1)=C_5^3(G_2)$: TATTTC		54
Level4	$U_1^3(G_1)U_1^3(G_2)$	$U_1^4(G_1)$: A	$U_1^4(G_2)$: T	−1
		$C_2^4(G_1)=C_2^4(G_2)$: AT		18
		$U_3^4(G_1)$: C	$U_3^4(G_2)$: T	−1
	$U_3^3(G_1)U_3^3(G_2)$	$U_4^4(G_1)$: G	$U_4^4(G_2)$: A	−1
		$C_5^4(G_1)=C_5^4(G_2)$: ATC		27
		$U_6^4(G_1)$: –	$U_6^4(G_2)$: T	−15

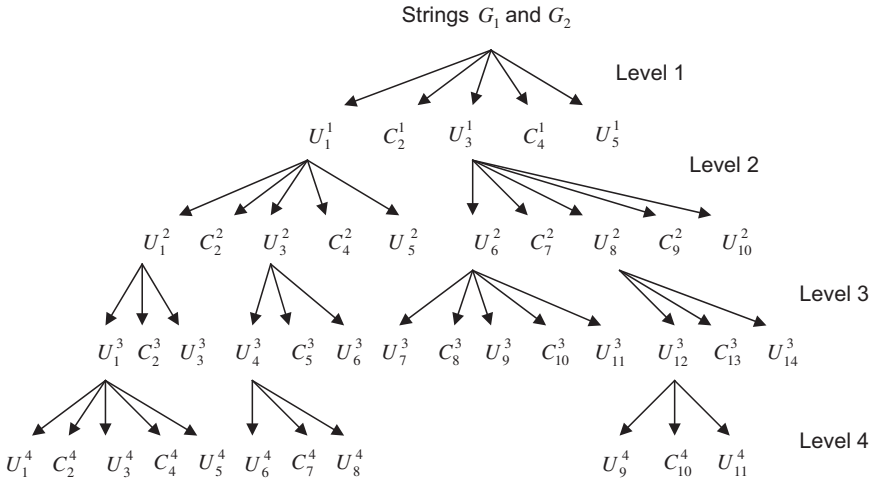


Fig. 6. The graphical simple alignment tree to be used to align the sequences G_1 and G_2 (G_1, GCCCTCGCGGGCAACATTTAATTCACAGCCAGTTCTCTCAACAG TGATTATC; G_2, CTGGGTCTTCAGGTCCTTTATGCTTAACACAAATCTATC GTTAACAGGACTATTCT).

formula to determine the 'best' gap values. Though GSA tree algorithm is a heuristic method, and could not always get the optimal alignment, the validity and practicality of the results can be ensured.

Firstly, we discuss the influence about the initial simple alignment of GSA tree algorithm. In Section 3.2, we describe in detail the construction steps of GSA tree by the initial simple alignment. As for the overall result, the initial simple alignment is very important. It determines the extent to which the consolidated results close to optimal results. Next, we discuss the initial simple alignment and its optimized features.

By the simple alignment algorithm a good simple alignment R of sequences G_1 and G_2 is determined when the score of the simple alignment reaches its peak. Then the largest common substrings C of R is determined, where $C=\phi\cup\{C_1,C_2,\ldots,C_m\}$. When $C=\{C_1,C_2,\ldots,C_m\}$, there are m the largest common substrings. Because of the limit of the initial alignment reaching peak score, some C_i of the largest common substrings of R may be a part of a larger common substring than C_i. An extension algorithm for the largest common substring is suggested to protect a larger common substring than C_i from being split by C_i if the larger common substring exists. Then we replace the original C_i with the new larger common substring also called as C_i. Once the largest common substrings C_i is determined, it is looked on as a part of the overall result. Then in an optimal alignment of G_1 and G_2 (This hypothetical alignment may be obtained by an absolutely optimized algorithm), the two sequences of C_i by GSA tree might be entirely overlapping in the optimal alignment, or partially overlapping, or one sequence might be entirely isolated from the other. Obviously, for the latter two cases, a great deal of gaps and mismatches may be introduced. So the substring C_i is most likely to appear in the optimal alignment.

The above explanation gives some approximate analysis instead of strict mathematical reasoning, but the conclusion is reasonable from the biological point of view. As the sequences under comparison are protein coding, gaps with lengths other than multiples of three are highly unlikely, whereas the GSA tree algorithm can avoid many single or two-base gaps by the approximate approach. Unlike "EMBOSS Pairwise Alignment Algorithms—needle (global)" or "—water (local)", the proposed GSA tree algorithm is to find a balance between the global and local. The algorithm is for aligning two sequences over their entire

Table 3
All substrings in Fig. 6 and the max score of each substring.

Substrings				The max score
Level 1	G_1, G_2	$U_1^1(G_1)$: GCCCTCGCGGGCAACATTTAATTC		40
		$U_1^1(G_2)$: CTGGGTCTTCAGGTCCTTTATGCTTA		
		$C_2^1(G_1)=C_2^1(G_2)$: ACA		27
		$U_3^1(G_1)$: GCCAGTTCTCTCAACAGTGAT		49
		$U_3^1(G_2)$: CAAATCTATCGTTAACAGGAC		
		$C_4^1(G_1)=C_4^1(G_2)$: TAT		27
		$U_5^1(G_1)$: C–;	$U_5^1(G_2)$: TCT	−17
Level 2	$U_1^1(G_1)$	$U_1^2(G_1)$: GCCCTCGCG	$U_1^2(G_2)$: CTGGGTCTTCA	−1
	$U_1^1(G_2)$	$C_2^2(G_1)=C_2^2(G_2)$: GG		18
		$U_3^2(G_1)$: CAACATTTAA	$U_3^2(G_2)$: TCCTTTATGC	10
		$C_4^2(G_1)=C_4^2(G_2)$: TT		18
		$U_5^2(G_1)$: C	$U_5^2(G_2)$: A	−1
	$U_3^1(G_1)$	$U_6^2(G_1)$: GCCAGTTC	$U_6^2(G_2)$:CAAATCTA	12
	$U_3^1(G_2)$	$C_7^2(G_1)=C_7^2(G_2)$: TC		18
		$U_8^2(G_1)$: TCAACAGT	$U_8^2(G_2)$: GTTAACAG	23
		$C_9^2(G_1)=C_9^2(G_2)$: GA		18
		$U_{10}^2(G_1)$: T	$U_{10}^2(G_2)$: C	−1
Level 3	$U_1^2(G_1)$	$U_1^3(G_1)$: GCCC	$U_1^3(G_2)$: CTGGGTCT	−2
	$U_1^2(G_2)$	$C_2^3(G_1)=C_2^3(G_2)$: TC		18
		$U_3^3(G_1)$: GCG	$U_3^3(G_2)$: A–	−17
	$U_3^2(G_1)$	$U_4^3(G_1)$: CAACA	$U_4^3(G_2)$: TCC	−9
	$U_3^2(G_2)$	$C_5^3(G_1)=C_5^3(G_2)$: TTTA		36
		$U_6^3(G_1)$: A–	$U_6^3(G_2)$:TGC	−17
	$U_6^2(G_1)$	$U_7^3(G_1)$:GCC	$U_7^3(G_1)$: CAA	−3
	$U_6^2(G_2)$	$C_8^3(G_1)=C_8^3(G_2)$:A		9
		$U_9^3(G_1)$:GT	$U_9^3(G_1)$:TC	−2
		$C_{10}^3(G_1)=C_{10}^3(G_2)$: T		9
		$U_{11}^3(G_1)$: C	$U_{11}^3(G_2)$: A	−1
	$U_8^2(G_1)$	$U_{12}^3(G_1)$: TC	$U_{12}^3(G_2)$: GTT	−7
	$U_8^2(G_2)$	$C_{13}^3(G_1)=C_{13}^3(G_2)$: AACAG		45
		$U_{14}^3(G_1)$: T	$U_{14}^3(G_2)$: –	−15
Level 4	$U_1^3(G_1)$	$U_1^4(G_1)$: –	$U_1^4(G_2)$: CTGG	−18
	$U_1^3(G_2)$	$C_2^4(G_1)=C_2^4(G_2)$: G		9
		$U_3^4(G_1)$: C	$U_3^4(G_2)$: T	−1
		$C_4^4(G_1)=C_4^4(G_2)$: C		9
		$U_5^4(G_1)$: C	$U_5^4(G_2)$: T	−1
	$U_4^3(G_1)$	$U_6^4(G_1)$: CAA	$U_6^4(G_2)$: T–	−17
	$U_4^3(G_2)$	$C_7^4(G_1)=C_7^4(G_2)$: C		9
		$U_8^4(G_1)$: A	$U_8^4(G_2)$: C	−1
	$U_{12}^3(G_1)U_{12}^3(G_2)$	$U_9^4(G_1)$: –	$U_9^4(G_2)$: G	−15
		$C_{10}^4(G_1)=C_{10}^4(G_2)$: T		9
		$U_{11}^4(G_1)$: C	$U_{11}^4(G_2)$: T	−1

Table 4
The complete coding sequence part of beta globin gene of Human (ACCESSION U01317) and Opossum (ACCESSION J03643).

Species	Complete coding sequence
Human	ACCESSION U01317; ATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGG CTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCCACTCCTGATGCTGTTATGGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTG CTCGGTGCCTTTAGTGATGGCCTGGCTCACCTGGACAACCTCAAGGGCACCTTTGCCACACTGAGTGAGCTGCACTGTGACAAGCTGCACGTGGATCCTGAGAACTTCAGG CTCCTGGGCAACGTGCTGGTCTGTGTGCTGGCCCATCACTTTGGCAAAGAATTCACCCCACCAGTGCAGGCTGCCTATCAGAAAGTGGTGGCTGGTGTGGCTAATGCCCTGG CCCACAAGTATCACTAA
Opossum	ACCESSION J03643; ATGGTGCACTTGACTTCTGAGGAGAAGAACTGCATCACTACCATCTGGTCTAAGGTGCAGGTTGACCAGACTGGTGGTGAGGCCCTTGGCAGGATG CTCGTTGTCTACCCCTGGACCACCAGGTTTTTTGGGAGCTTTGGTGATCTGTCCTCTCCTGGCGCTGTCATGTCAAATTCTAAGGTTCAAGCCCATGGTGCTAAGGTGTTGAC CTCCTTCGGTGAAGCAGTCAAGCATTTGGACAACCTGAAGGGTACTTATGCCAAGTTGAGTGAGCTCCACTGTGACAAGCTGCATGTGGACCCTGAGAACTTCAAGATGCT GGGGAATATCATTGTGATCTGCCTGGCTGAGCACTTTGGCAAGGATTTTACTCCTGAATGTCAGGTTGCTTGGCAGAAGCTCGTGGCTGGAGTTGCCCATGCCCTGGCCCA CAAGTACCACTAA

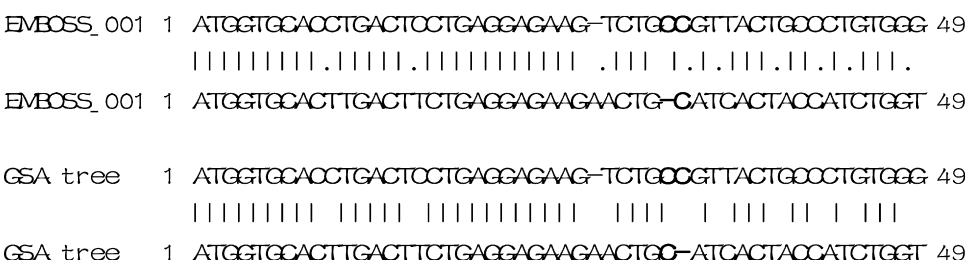
length. The match areas with local superiority are produced in the global context. In bioinformatics, it may be reasonable to assume that in the global context the local areas with closely related sequences should be reflected because of the stability of these areas. As for two possibly related sequences, giving priority to these local areas may be a better choice. The accurate optimization

Fig. 7. The global alignment between string *a* and string *b*. Identities: 328/448 (73.2%); Gaps: 8/448 (1.8%); Score: 2772.

result could not show these local features because of the global optimization. The proposed GSA tree algorithm is for aligning two sequences over their entire length. This works best with closely related sequences. If one uses GSA tree to align very distantly related sequences, it will produce a result but much of the alignment may have little or no biological significance. The three examples of 4.1 give practical proofs for the validity and practicality of the GSA tree algorithm. The Example 1 gives two very similar but very short sequences. The GSA tree achieves the same results as needle. The Example 3 gives two similar and long sequences. There is only one different area:

This example shows that giving priority to these local areas may be a better choice. The Example 2 gives two random sequences. There are two different areas in their results. A close look at the alignments discovers that both of the methods can find out those very similar local areas. The main differences between them lie in the very distantly related areas.

Finally, the following analysis shows why we consider the substring C_i in the only initial alignment reaching peak score. Because of the limit of the initial alignment reaching peak score, some C_i of the largest common substrings of R may be a part of a larger common substring than C_i. Then an extension algorithm for



the largest common substring is suggested to protect a larger common substring than C_i from being split by C_i if the larger common substring exists. In the extension algorithm we introduce three parameters: K, L_L and L_R, where K be the length of the largest common substring of G_1 and G_2. When $k=K$ ($k=|C_i|$), none of the largest common substrings C_i of R can be extended into larger common substring. The largest common substrings of R are $C_1, C_2, \ldots, C_m$, respectively. There is no larger common substring than C_i split by C_i. Otherwise, when $k<K$, there exists at least a larger common substring than C_i. The larger common substring and C_i might be partially overlapping, or one might be entirely isolated from the other. When they are entirely isolated from each other, the larger common substring can be completely preserved and found out in the next sub-alignment. As for being partially overlapping, we face a choice: the larger common substring or C_i. In order to give a better choice, we use parameters K, L_L and L_R to construct two new sub-sequences S_i^1 and S_i^2. We will replace the original C_i with the new substring also called as C_i if there is an increment of sore when the new larger common substring comes into being. So we can protect a larger common substring than C_i^1 from being split by C_i^1 and eliminate the limitations caused by the only initial alignment reaching peak score.

References

Althaus, I.W., Chou, J.J., Gonzales, A.J., Diebel, M.R., Chou, K.C., Kezdy, F.J., Romero, D.L., Aristoff, P.A., Tarpley, W.G., Reusser, F., 1993a. Kinetic studies with the nonnucleoside HIV-1 reverse transcriptase inhibitor U-88204E. Biochemistry 32, 6548–6554.

Althaus, I.W., Chou, J.J., Gonzales, A.J., Diebel, M.R., Chou, K.C., Kezdy, F.J., Romero, D.L., Aristoff, P.A., Tarpley, W.G., Reusser, F., 1993b. Steady-state kinetic studies with the non-nucleoside HIV-1 reverse transcriptase inhibitor U-87201E. Journal of Biological Chemistry 268, 6119–6124.

Althaus, I.W., Chou, J.J., Gonzales, A.J., Diebel, M.R., Chou, K.C., Kezdy, F.J., Romero, D.L., Aristoff, P.A., Tarpley, W.G., Reusser, F., 1993c. The quinoline U-78036 is a potent inhibitor of HIV-1 reverse transcriptase. Journal of Biological Chemistry 268, 14875–14880.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., 1990. Basic local alignment search tool. Journal of Molecular Biology 215, 403–410.

Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J., 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Research 25, 3389–3402.

Andraos, J., 2008. Kinetic plasticity and the determination of product ratios for kinetic schemes leading to multiple products without rate laws: new methods based on directed graphs. Canadian Journal of Chemistry 86, 342–357.

Chou, K.C., 1980. A new schematic method in enzyme kinetics. European Journal of Biochemistry 113, 195–198.

Chou, K.C., 1989. Graphical rules in steady and non-steady enzyme kinetics. Journal of Biological Chemistry 264, 12074–12079.

Chou, K.C., 1990. Review: applications of graph theory to enzyme kinetics and protein folding kinetics. Steady and non-steady state systems. Biophysical Chemistry 35, 1–24.

Chou, K.C., 1993. Graphic rule for non-steady-state enzyme kinetics and protein folding kinetics. Journal of Mathematical Chemistry 12, 97–108.

Chou, K.C., Forsen, S., 1980. Graphical rules for enzyme-catalyzed rate laws. Biochemical Journal 187, 829–835.

Chou, K.C., Forsen, S., 1981. Graphical rules of steady-state reaction systems. Canadian Journal of Chemistry 59, 737–755.

Chou, K.C., Jiang, S.P., Liu, W.M., Fee, C.H., 1979. Graph theory of enzyme kinetics: 1. Steady-state reaction system. Scientia Sinica 22, 341–358.

Chou, K.C., Kezdy, F.J., Reusser, F., 1994. Review: steady-state inhibition kinetics of processive nucleic acid polymerases and nucleases. Analytical Biochemistry 221, 217–230.

Chou, K.C., Liu, W.M., 1981. Graphical rules for non-steady state enzyme kinetics. Journal of Theoretical Biology 91, 637–654.

Chou, K.C., Zhang, C.T., 1992. Diagrammatization of codon usage in 339 HIV proteins and its biological implication. AIDS Research and Human Retroviruses 8, 1967–1976.

Chou, K.C., Zhang, C.T., Elrod, D.W., 1996. Do antisense proteins exist? Journal of Protein Chemistry 15, 59–61.

Gao, L., Ding, Y.S., Dai, H., Shao, S.H., Huang, Z.D., Chou, K.C., 2006. A novel fingerprint map for detecting SARS-CoV. Journal of Pharmaceutical and Biomedical Analysis 41, 246–250.

〈http://blast.ncbi.nlm.nih.gov/bl2seq/wblast2.cgi〉.

〈http://www.ebi.ac.uk/Tools/emboss/align/index.html〉.

Kuzmic, P., Ng, K.Y., Heath, T.D., 1992. Mixtures of tight-binding enzyme inhibitors. Kinetic analysis by a recursive rate equation. Analytical Biochemistry 200, 68–73.

Myers, D., Palmer, G., 1985. Microcomputer tools for steady-state enzyme kinetics. Bioinformatics (Original: Computer Applied Bioscience) 1, 105–110.

Okayasu, T., Sorimachi, K., 2009. Organisms can essentially be classified according to two codon patterns. Amino Acids 36 (2), 261–271.

Pearson, W.R., Lipman, D.J., 1988. Improved tools for biological sequence comparison. In: Proceedings of the National Academy of Sciences of the United States of America, vol. 85, pp. 2444–2448.

Qi, X.Q., Wen, J., Qi, Z.H., 2007. New 3D graphical representation of DNA sequence based on dual nucleotides. Journal of Theoretical Biology 249, 681–690.

Qi, Z.H., Qi, X.Q., 2007. Novel 2D graphical representation of DNA sequence based on dual nucleotides. Chemical Physics Letters 440, 139–144.

Qi, Z.H., Fan, T.R., 2007. PN-curve: a 3D graphical representation of DNA sequences and their numerical characterization. Chemical Physics Letters 442, 434–440.

Qi, Z.H., Qi, X.Q., 2009. Numerical characterization of DNA sequences based on digital signal method. Computers in Biology and Medicine 39, 388–391.

Qi, Z.H., Wang, J.M., Qi, X.Q., 2009. Classification analysis of dual nucleotides using dimension reduction. Journal of Theoretical Biology 206, 104–109.

Randić, M., Vracko, M., Lers, N., Plavsic, D., 2003a. Analysis of similarity/dissimilarity of DNA sequences based on novel 2-D graphical representation. Chemical Physics Letters 371, 202–207.

Randić, M., Vracko, M., Lers, N., Plavsic, D., 2003b. Novel 2-D graphical representation of DNA sequences and their numerical characterization. Chemical Physics Letters 368 (1–2), 1–6.

Randić, M., Zupan, J., Drazen, V.T., Plavsic, D., 2006. A novel unexpected use of a graphical representation of DNA: Graphical alignment of DNA sequences. Chemical Physics Letters 431, 375–379.

Smith, T.F., Waterman, M.S., 1981. Identification of common molecular subsequences. Journal of Molecular Biology 147, 195–197.

Sorimachi, K., Okayasu, T., 2008. Universal rules governing genome evolution expressed by linear formulas. Open Genomics Journal 1, 33–43.

Tatiana, A.T., Thomas, L.M., 1999. Blast 2 sequences—a new tool for comparing protein and nucleotide sequences. FEMS Microbiology Letters 174, 247–250.

Waterman, M.S., 1984. General methods of sequence comparison. Bulletin of Mathematical Biology 46, 473–500.

Wang, M., Yao, J.S., Huang, Z.D., Xu, Z.J., Liu, G.P., Zhao, H.Y., Wang, X.Y., Yang, J., Zhu, Y.S., Chou, K.C., 2005. A new nucleotide-composition based fingerprint of SARS-CoV with visualization analysis. Medicinal Chemistry 1, 39–47.

Xiao, X., Shao, S., Ding, Y., Huang, Z., Chen, X., Chou, K.C., 2005a. Using cellular automata to generate image representation for biological sequences. Amino Acids 28, 29–35.

Xiao, X., Shao, S., Ding, Y., Huang, Z., Chen, X., Chou, K.C., 2005a. An application of gene comparative image for predicting the effect on replication ratio by HBV virus gene missense mutation. Journal of Theoretical Biology 235, 555–565.

Xiao, X., Shao, S.H., Ding, Y.S., Huang, Z.D., Chou, K.C., 2006b. Using cellular automata images and pseudo amino acid composition to predict protein subcellular location. Amino Acids 30, 49–54.

Xiao, X., Shao, S.H., Chou, K.C., 2006b. A probability cellular automaton model for hepatitis B viral infections. Biochemical and Biophysical Research Communication 342, 605–610.

Xiao, X., Wang, P., Chou, K.C., 2008. Predicting protein structural classes with pseudo amino acid composition: an approach using geometric moments of cellular automaton image. Journal of Theoretical Biology 254, 691–696.

Xiao, X., Wang, P., Chou, K.C., 2009. GPCR-CA: a cellular automaton image approach for predicting G-protein-coupled receptor functional classes. Journal of Computational Chemistry 30, 1414–1423.

Yao, Y.H., Nan, X.Y., Wang, T.M., 2006. A new 2D graphical representation-classification curve and the analysis of similarity/dissimilarity of DNA sequences. Journal of Molecular Structure: THEOCHEM 764, 101–108.

Yao, Y.H., Qi, D., Nan, X.Y., He, P.A., Nie, Z.M., Zhou, S.P., Zhang, Y.Z., 2008a. Analysis of similarity/dissimilarity of DNA sequences based on a class of 2D graphical representation. Journal of Computational Chemistry 29, 1632–1639.

Yao, Y.H., Qi, D., Li, C., He, P.A., Zhang, Y.Z., 2008b. Analysis of similarity/dissimilarity of protein sequences. PROTEINS: Structure, Function, and Bioinformatics 73, 864–871.

Zhang, C.T., Chou, K.C., 1994. Analysis of codon usage in 1562 E. coli protein coding sequences. Journal of Molecular Biology 238, 1–8.

Zhou, G.P., Deng, M.H., 1984. An extension of Chou's graphical rules for deriving enzyme kinetic equations to system involving parallel reaction pathways. Biochemical Journal 222, 169–176.