International Journal of Applied Science and Engineering
2005. 3, 2: 89-100

Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer Techniques

Shyi-Ming Chen*, Chung-Hui Lin, and Shi-Jay Chen

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology
Taipei 106, Taiwan, R. O. C.

Abstract: Multiple DNA sequence alignment is one of the important research topics of bioinformatics. Because of the huge length of DNA sequences of advanced organisms, some researchers used divide-and-conquer techniques to cut the sequences for decreasing the space complexity for sequence alignment. Because the cutting points of sequences of the existing methods are fixed at the middle or the near-middle points, the performance of sequence alignment of the existing methods is not good enough. In this paper, we present a new method for multiple DNA sequence alignment using genetic algorithms and divide-and-conquer techniques to choose optimal cut points of multiple DNA sequences. The experimental results show that the proposed method is better than the existing method for dealing with multiple DNA sequence alignment.

Keywords: DNA sequences; sequence alignment; multiple sequence alignment; divide-and-conquer techniques; genetic algorithms.

1. Introduction

DNA (Deoxyribonucleic Acid) is the basic unit of an organism, and its length ranges from a few hundreds to several billions of nucleotides for different species. Thus, to find out the degrees of similarity and the degrees of difference between DNA sequences is a complicated task. From [15], we can see that sequence alignment, especially multiple sequence alignment, is an important research topic of bioinformatics. The results of multiple sequence alignment can be used for other more complicated genetic research. For example, we can use the results of multiple sequence alignment to find out the degrees of similarity of different species, understand the evolutionary history of species, or put a newly found species into a family that it may belong to. In recent years, some methods for multiple sequence alignment have been proposed [1, 6, 8, 11, 15, 16, 18].

In [1], Chellapilla et al. presented a method for multiple sequence alignment using evolutionary programming techniques. In [12], Needleman et al. used dynamic programming techniques for multiple sequence alignment. In [6], Isokawa et al. presented a method to deal with multiple sequence alignment using a genetic algorithm (GA). In [13], Notredame et al. presented a method called SAGA for sequence alignment by genetic algorithms. It

* Corresponding author: e-mail: smchen@et.ntust.edu.tw
Accepted for Publication: May 04, 2005

involved evolving a population of alignments in a quasi-evolutionary manner and gradually improving the fitness of the population by an objective function that measures the multiple sequence alignment quality. In [16], Stoye presented techniques of multiple sequence alignment using a divide-and-conquer method, where an increase of the speed compared to optimal multiple alignment by dynamic programming can be guaranteed. In [18], Thompson et al. presented a method for improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positioning specific gap penalties and weight matrix choice. In [20], Waterman presented general methods of sequence comparison. In [21], Zhang et al. presented a genetic algorithm for multiple molecular sequence alignment. In [22], Zhang et al. presented a method for efficient multiple molecular sequence alignment based on genetic algorithms and dynamic programming (DP) techniques. In [9], Lin used genetic algorithms to deal with the multiple sequence alignment problems, where every randomly generated multiple sequence alignment is transformed into an individual chromosome. However, the sequence lengths of advanced organisms are extremely large. For example, the length of human beings' DNA sequences is longer than three billions of "base pairs". In this situation, it is a very tedious work for dealing with multiple sequence alignment by using the DP or the GA due to the fact that they are very time-consuming and their time complexities are too high. Therefore, some methods using divide-and-conquer techniques are presented to deal with the multiple sequence alignment problems for reducing the space complexities [4, 15]. Because the cutting points of the sequences of the existing methods are fixed at the middle or the near-middle points, the performance of the multiple DNA sequence alignment of the existing methods is not good enough.

In this paper, we present a new method for multiple DNA sequence alignment using genetic algorithms and divide-and-conquer techniques to choose optimal cut points of multiple DNA sequences. The experimental results show that the proposed method is better than the method presented in [4] to deal with multiple DNA sequence alignment.

The rest of this paper is organized as follows. In Section 2, we briefly review basic concepts of divide-and-conquer techniques [4, 15] and genetic algorithms [2, 5, 9, 10]. In Section 3, we present a new method for multiple DNA sequence alignment based on genetic algorithms and divide-and-conquer techniques. In Section 4, we use an example to illustrate the multiple DNA sequence alignment process of the proposed method. In Section 5, we show the experimental results of the proposed method. The conclusions are discussed in Section 6.

2. Preliminaries

In this section, we briefly review some basic concepts of divide-and-conquer techniques [4, 15] and genetic algorithms [2, 5, 9, 10].

(A) Divide-and-conquer techniques: In [4], Hirschberg presented algorithms for multiple sequence alignment by using divide-and-conquer techniques. The divide-and-conquer techniques can effectively reduce the space complexity for multiple sequence alignment. First, cutting points of sequences are found, and the sequences are separated into two sets of subsequences, according to the cutting points. If the length of a subsequence in a subset is larger than a predefined threshold value, then the subsequence will be cut again until the length of any subsequence in the subset is smaller than the threshold value. Then, the dynamic programming techniques are used for aligning the subsequences in the subsets. Finally, the aligned subsequences in the subsets are merged into a complete multiple sequence alignment.

In [4], Hirschberg used the middle points as

the cutting points of the sequences. He pointed out that the time complexity of the dynamic programming algorithm for sequence alignment is roughly the number of edges in the edit graph; the space complexity is roughly the number of vertices in the edit graph,. However, if we only want to compute the score of the alignment, then the space can be reduced to just twice the number of vertices in a single column of the edit graph. Furthermore, In [16], Stoye presented a divide-and-conquer method for multiple sequence alignment, where dynamic near-middle points are used as the cutting points of the sequences. The alignment method based on the divide-and-conquer techniques has the benefit for increasing the speed compared to multiple sequence alignment by using the dynamic programming method.

(B) Genetic algorithms: In 1975, Holland proposed the genetic algorithms [5] based on the theory of evolution of Darwin. Based on the adaptation of organisms of complicated nature, he created an evolutionary system with the operators of selection, crossover and mutation. In genetic algorithms, a population consists of chromosomes, where a chromosome consists of genes. In the following, we briefly review basic operations of genetic algorithms from [2], [10] and [11]:

(1) ***Selection operation****:* The selection operator chooses better chromosomes in a population according to the fitness values of chromosomes. The larger the fitness value of a chromosome is, the more the opportunity of the chromosome to be chosen. The roulette wheel selection method and the ranking selection method are commonly used selection methods.

(2) ***Crossover operation****:* The system chooses a pair of chromosomes from a population by the selection operation, and then randomly generates a crossover point. A crossover rate CR $\in$ [0, 1] will be given by the user for determining whether the pair of chromosomes will crossover or not. If a random number between zero and one generated by the system is smaller than or equal to the value of CR, then the crossover operation will be performed. The single-point crossover, the double-point crossover, and the uniform crossover are commonly used crossover methods.

(3) ***Mutation operation****:* The crossover operation may find a solution falling into a local optimum because of the spatial localization of the crossover operation. Thus, we can use the mutation operation for jumping the solution out from a local optimum into a global one. First, the mutation rate MR $\in$ [0, 1] will be set to decide whether a chromosome will mutate or not. If the random number, generated by the system, is smaller than or equal to the value of MR, then the mutation operation will be performed. The binary mutation and the real value mutation are commonly used mutation methods.

3. A new method for multiple DNA sequence alignment based on genetic algorithms and divide-and-conquer techniques

In this section, we present a new method for multiple DNA sequence alignment based on genetic algorithms and divide-and-conquer techniques. It uses genetic algorithms for finding the best cut-points of a set of sequences, and then aligns the subsequence sets by dynamic programming techniques. It can greatly reduce the space complexity to deal with the multiple DNA sequence alignment problem. Assume that there is a set S of sequences shown as follows:

$$S = \{s_i \mid i = 1,2,...,n\},$$

where s_i denotes the ith DNA sequence in the set S. There are totally n DNA sequences in the sequence set S, and the DNA sequences have different lengths. A user-defined threshold value δ should be given. For example, if

we do not allow the length of any subsequence in the sequence set S to be larger than 100, then we can set the threshold value δ to 100. The proposed method for dealing with the multiple DNA sequence alignment problem is now presented as follows:

Step 1: Choose the shortest DNA sequence in the sequence set S. If the length of the chosen sequence is smaller than or equal to the threshold value δ, then go to Step 2. Otherwise, the sequences in the set S need to be cut by using a genetic algorithm to get the set C of cut-points of the sequences in the sequence set S, C = {c_1, c_2, ⋯, c_n}, where c_i denotes the cut-point of the ith sequence in the set S. In the following, we present the chromosome format, the fitness function, the selection operator, the crossover operator, and the mutation operator of the genetic algorithm as follows:

(1) Chromosome format: Assume that the set S contains a set of DNA sequences and the set C contains a set of cut-points of the DNA sequences,

$$S = \{s_i \mid i = 1,2,...,n\},$$
$$C = \{c_i \mid i = 1,2,...,n\},$$

where c_1, c_2, ⋯, and c_n denote the cut-points of the DNA sequences $s_1, s_2, ..., and\ s_n$, respectively, c_i is a positive integer, and 1 < i ≤ n. Then, $(c_1, c_2, ..., c_n)$ is defined as a chromosome, where c_1, c_2, ⋯, and c_n are the cut-points of the DNA sequences s_1, s_2, ⋯, and s_n, respectively. The format of a chromosome is shown in Figure 1.

Chromosome	c_1	c_2	…	c_n

Figure 1. The format of a chromosome

(2) Fitness function: In [16], Stoye pointed out that a set of good cutting points minimizes the following formula:

$$\sum_{1 \le p \le q \le n} \alpha_{p,q} C_{s_p,s_q}(c_p, c_q), \quad (1)$$

where c_p and c_q are cut-points of the sequences s_p and s_q, respectively, C_{s_p,s_q} denotes the cost function between sequences s_p and s_q (Note: In [17], Stoye et al. pointed out that $C_{s_p,s_q}(c_p, c_q) = w_{opt}(prefix) + w_{opt}(suffix) - w_{opt}(total)$ denotes the additional cost to estimate the compatibility of cut-points c_p and c_q of the two sequences s_p and s_q, where "prefix" is the subsequence of s_p (or s_q) with indices running from 1 to c_p (or c_q), "suffix" denotes the subsequence of s_p (or s_q) with indices running from $c_p + 1$ (or $c_q + 1$) to n_p (or n_q), "total" is the sequence s_p (or s_q), and w_{opt} denotes an appropriate function to measure the degree of similarity between two sequences [8]), and $\alpha_{p,q}$ denotes the weight between sequences s_p and s_q according to the degrees of similarity between s_p and s_q, where $1 \le p \le q \le n$. For example, assume that s_p denotes the DNA sequence of human beings and s_q denotes the DNA sequence of gorillas. Because the two DNA sequences s_p and s_q are very similar, we can set $\alpha_{p,q}$ to one. In this paper, we let $\alpha_{p,q} = 1$. From Equation (1), we can see that good cut-points should be able to minimize the difference between "the summation of the degrees of similarity between subsequences in front of the cut-points (called the prefix subsequences) and the degrees of similarity after the cut-points (called the suffix subsequences)" and "the degrees of similarity between sequences before performing the cut operations". Therefore, in this paper, we define the degree of effect "effect(C)" of a set C of cut-points, shown as follows:

$$\text{effect(C)} = \sum_{1 \le p < q \le n} [\text{Sim}(S_p-, S_q-) + \text{Sim}(S_p+, S_q+) - \text{Sim}(S_p, S_q)], \quad (2)$$

where S_i- and S_i+ denote the prefix subsequence and the suffix subsequence of se-

quence S_i, based on the ith cut-point c_i in the cut-point set C, $C = \{c_1, c_2, \cdots, c_n\}$, and $1 \le i \le n$. Sim denotes a similarity function to calculate the degree of similarity between sequences. Based on Equation (2), we can calculate the degree of effect "effect(C)" of a set C of cut-points. In this paper, we use a simple way for calculating the degree of similarity between two sequences described as follows. First, by aligning the two sequences to the left, we let the longer sequence be the "reference sequence" and let the shorter sequence be the "comparison sequence". Because the lengths of these two sequences are not the same, we insert the symbols "—" at the right-hand side of the comparison sequence for making these two sequences have the same length. Initially, we set the evaluation score *P* to zero. Then, we calculate the degree of similarity between sequences by columns. For each column in the comparison sequence, if the symbol of a column in the comparison sequence is the same as the symbol at the same column of the reference sequence, then we add 3 points into *P*; if the symbol of a column *x* in the comparison sequence is the same as the symbol at the previous column or the next column of the column *x* of the reference sequence, then we add 1 point into *P*. Otherwise, we let *P* unchanged. Finally, *P* is divided by (3 × the number of columns in the reference sequence), and then we can get the degree of similarity between the two sequences.

In this paper, we define the fitness value "fitness(C)" of a set C of cut-points as follows:

$$fitness(C) = \frac{100}{effect(C)}, \quad (3)$$

where $C = \{c_1, c_2, \cdots, c_n\}$, c_i denotes the cut-point of the sequence s_i, and $1 \le i \le n$.

(3) Selection operation: In this paper, the roulette wheel selection method [2, 9] is used to deal with the selection operation. First, we sort the chromosomes in a population according to their fitness values. Then, we assign the area to every chromosome, where the chromosome with the highest fitness value has the largest area. The system randomly generates a value between zero and one, and the chromosome corresponding to the area that the number falls into is chosen. For example, let g_i denote the fitness value of the ith chromosome in a population and let g_t denote the summation of the fitness values of all chromosomes, i.e. $g_t = \sum_{i=1}^{n} g_i$. The selected probability P_{g_i} of the ith chromosome is denoted by $P_{g_i} = \frac{g_i}{g_t}$. When the selection operation is performed, the system chooses a chromosome from a population according to its degree of probability as shown in Figure 2 [2, 9]. Therefore, the larger the selected probability of a chromosome is, the more the opportunity of the chromosome to be chosen for performing the crossover operation.

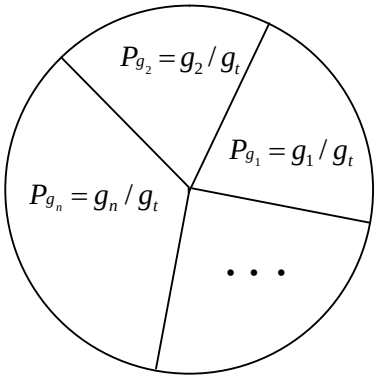

Figure 2. The selection probabilities of chromosomes using the roulette wheel method [2, 9]

(4) Crossover operation: When the system performs the crossover operation of chromosomes, it must choose a pair of chromosomes from a population by the selection operation. After a pair of chromosomes has been chosen by the selection operation, the system randomly generates

a value between zero and one and compares it with a predefined crossover rate CR determined by the user, where CR $\in$ [0, 1]. If it is smaller than or equal to the value of CR, then the crossover operation is performed to exchange the genes before a randomly generated crossover point of the chromosomes. Then, their offspring are put into a mating pool. Otherwise, the selected pair of chromosomes will directly be copied into the mating pool, where the size of the mating pool is equal to the size of the population. The crossover operation will be performed repeatedly until the mating pool is full. An example of the crossover operation is shown in Figure 3.

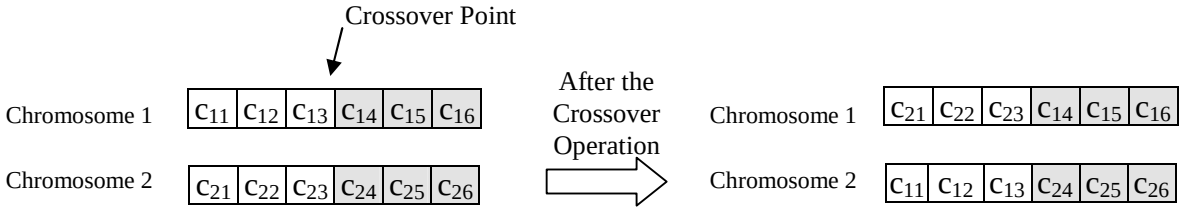

Figure 3. The crossover operation

(5) Mutation operation: After performing the crossover operation, the mutation operation is performed to prevent the system from falling into a local optimum. The system randomly chooses a gene of a chromosome to form the mating pool and randomly generates a value between zero and one. Then, the system compares the randomly generated value with a predefined mutation rate MR determined by the user, where MR $\in$ [0, 1]. If the randomly generated value is smaller than or equal to the value of MR, the mutation operation is performed to replace the value of the selected gene by a value randomly generated by the system, and then the offspring is put into the mating pool. Otherwise, the chromosome will be left unchanged. An example of the mutation operation is shown in Figure 4.

The steps of the genetic algorithm are shown in Figure 5, where the symbol "genSize" denotes the number of generations necessary to be evolved in the genetic algorithm; the symbol "popSize" denotes the number of chromosomes in a population; X and Y denote chromosomes chosen by the selection operation.

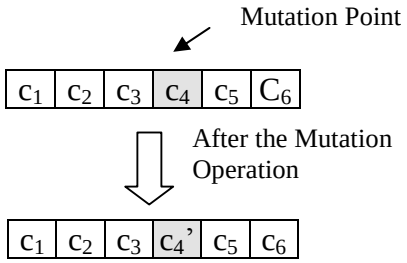

Figure 4. The mutation operation

Step 2: Align each subsequence obtained from Step 1 by the dynamic programming method [3], described as follows. First, calculate the shortest distance between any two sequences in the sequence set S, where the distance between two sequences is calculated as follows. Assume that there are two sequences s_1 and s_2 shown as follows:

s_1 = ACTC,
s_2 = AC.

We can get an alignment of s_1 and s_2, respectively, shown as follows:

s_1 = ACTC,

s_2 = A--C.

Then, we can calculate the distance $d(s_1, s_2)$ between the sequences s_1 and s_2 shown as follows:

$d(s_1, s_2)$ = the number of columns in which the corresponding symbol is different
= 2.

Assume that the set S contains a set of sequences to be aligned, where $s_i \in S$ and $S = \{s_1, s_2, \cdots, s_n\}$. If the sum of distances between s_i and the other sequences in the set S is minimal, then s_i is called the center of the set *S*. Assume that s_1 is the center of the set S, then based on [15], after performing the following steps:

for *j* := 2 to n do

for the sequence s_1 and the sequence s_j in the set S, find out the best sequence alignment for s_1 with s_j;

adding the symbols "—" into appropriate positions of sequences which have been aligned if the length of the sequences are smaller than the length of s_1

end,

we can get an optimal multiple DNA sequence alignment.

```
Generate the initial population P;
for i = 1 to genSize do
   for j = 1 to (popSize/2) do
      Select two chromosomes X and Y
      from P by Selection operation;
      Crossover(X ,Y) and let X' and Y'
      be the offspring;
      Mutate(X');
      Mutate(Y');
      Put X' and Y' into the mating pool
      M
   end
   Let M be the new population P
```

Figure 5. Steps of the genetic algorithm

Step 3: Merge all these subsequence alignments obtained from Step 2 to get the complete multiple DNA sequence alignment. In this step, we use a technique to optimize the sequence alignments, described as follows. Assume that the two sequence sets S_1 and S_2 are being merged. First, we scan every position of symbols in the last column of sequence S_1 and the first column of sequence S_2. If there is at least one "—" in these two symbols, we know that the sets S_1 and S_2 can be optimized. Then, we replace the symbols "—"in the first column of sequence set S_2 with the symbol in the last column of sequence set S_1. Then, we delete the last column of sequence set S_1, and repeat Step 3 until there are no sets that can be optimized. Finally, we can get the multiple DNA sequence alignment.

After we got a multiple DNA sequence alignment, we can calculate its score by the column-pair scoring method described as follows. First, we calculate the scores of every column of the multiple DNA sequence alignment. For each pair of symbols in a column, if the two symbols are the same, we give a "match pair score" to the pair; if they are different and they are not the symbol "—" either, we give a "mismatch pair score" to the pair; if at least one symbol of them is the symbol "—", we give a "space pair score" to the pair. In this paper, the match pair score, the mismatch pair score, and the space pair score are 2, -1, and-2, respectively [3]. Then, the score of a column of the alignment is equal to the summation of all the pair scores of the pairs in the column, and the score of a multiple DNA sequence alignment is equal to the summation of all the column scores of columns in the multiple DNA sequence alignment.

In summary, the proposed method for multiple DNA sequence alignment is shown in Figure 6.

4. An example

In this section, we use an example to illustrate the multiple DNA sequence alignment process of the proposed method. Assume that there is a sequence set $S = \{s_1, s_2, s_3\}$ of DNA sequences to be aligned, where

s_1 = AATTCCTGGATCGTCGG,
s_2 = AATCTGGTCGTCTG,

s_3 = ATTCGATGGATCCGG.

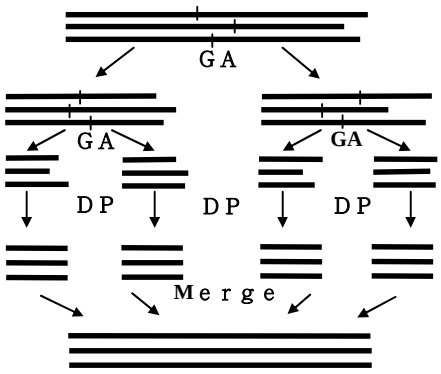

Figure 6. The proposed multiple DNA sequence alignment process

Assume that the threshold value determined by the user is 6. Then,

[Step 1] First, we can see the shortest sequence in the sequence set S is s_2, where the length of s_2 is 10. Because all sequences in the set S are longer than the threshold value (i.e., 6), we must cut each sequence in the set S by the genetic algorithm. In this paper, the roulette wheel selection operator, the single-point crossover operator, and the real value mutation operator are used. Therefore, we can get a set C_1 of cut-points, where C_1 = {7, 5, 8}. According to the set C_1 of the cut-points, the set S is separated into two subsequence sets A = {a_1, a_2, a_3} and B = {b_1, b_2, b_3}, where

a_1 = AATTCCT, b_1= GGATCGTCGG,
a_2 = AATCT, b_2 = GGTCGTCTG,
a_3 = ATTCGATG, b_3 = GATCCGG.

Then, we can see that the shortest sequence b_3 in the subsequence set B is still longer than the threshold value. Therefore, the subsequence set B needs to be cut to get a set of cut-points. In this case, we get a set C_2 of cut-points by the genetic algorithm again, where C_2 = {5, 6, 4}. Then, we cut the subsequence set B into two subsequence sets D = {d_1, d_2, d_3} and E = {e_1, e_2, e_3} based on the set C_2 of cut-points, where

d_1 = GGATC, e_1 = GTCGG,
d_2 = GGTCGT, e_2 = CTG,
d_3 = GATC, e_3 = CGG.

[Step 2] Because the set S has been separated into several subsequence sets, where the length of each subsequence in each subsequence set is smaller than the threshold value, we continue to align these subsequence sets by dynamic programming techniques to find out their optimal multiple sequence alignments. For example, let us consider the subsequence set A. First, we can see that the summation of distances of the subsequences a_1, a_2 and a_3 with respect to other sequences are 8, 6 and 6, respectively. Therefore, the subsequence a_2 is the center of the subsequences set A. Then, we align a_2 with other sequences, respectively. For aligning a_2 with a_1, we can get their optimal sequence alignment, shown as follows:

a_2 = AAT-C-T,
a_1 = AATTCCT.

In the same way, we can get the optimal sequence alignment of a_2 and a_3, shown as follows:

a_2 = AAT-C--T-,
a_3 = A-TTCGATG.

Then, by inserting the symbols "—" into a_1 which is already aligned, we can get

a_1 = AATTCC-T-,
a_2 = AAT-C--T-,
a_3 = A-TTCGATG.

By repeating the same process, we can get the optimal sequence alignments of the sequences in the subsequence sets D = {d_1, d_2, d_3} and E = {e_1, e_2, e_3}, shown as follows:

d_1 = GGATC--, e_1 = GTCGG,
d_2 = GG-TCGT, e_2 = --CTG,
d_3 = G-ATC--, e_3 = --CGG.

[Step 3] By merging the sequences in the three aligned subsequence sets A, D and E, we can get a multiple sequence alignment S' = {s_1', s_2', s_3'}, where s_i' is obtained by

merging the sequences a_i, d_i and e_i and $1 \leq i \leq 3$, shown as follows:

s_1' = AATTCC-T-GGATC--GTCGG,
s_2' = AAT-C--T-GG-TCGT--CTG,
s_3' = A-TTCGATGG-ATC----CGG.

In this paper, for each column of an alignment, the score of "match pair", the score of "mismatch pair" and the score of "space pair" are +2, -1 and –2, respectively. First, we calculate the score of each column of the alignment. For example, for the 6th column of the multiple DNA sequence alignment S', there are three pairs {C, —}, {—, G} and {G, C}. Therefore, we can calculate the score of the 6th column of the alignment as follows:

Score of the 6th column = (-2) + (-2) + (-1)
= -5.

In the same way, we can get the score of other columns of the alignment, shown as follows:

Score of the 1st column = 6, Score of the 2nd column = -2, Score of the 3rd column = 6,

Score of the 4th column = -2, Score of the 5th column = 6, Score of the 6th column = -5,

Score of the 7th column = -6, Score of the 8th column = 6, Score of the 9th column = -6,

Score of the 10th column = 6, Score of the 11th column = -2, Score of the 12th column = -2,

Score of the 13th column = 6, Score of the 14th column = 6, Score of the 15th column = -6,

Score of the 16th column = -6, Score of the 17th column = -6, Score of the 18th column = -6,

Score of the 19th column = 6, Score of the 20th column = 0, Score of the 21th column = 6.

Then, we can calculate the score of the multiple DNA sequence alignment S' by taking the summation of the score of each column of the sequences, shown as follows:

Score of S' = 6 + (-2) + 6 + (-2) + 6 + (-5) + (-6) + 6 + (-6) + 6 + (-2) + (-2)+ 6 + 6 + (-6) + (-6) + (-6) + (-6) + 6 + 0 + 6
= 5,

where the 15th, the 16th, the 17th and the 18th columns of the multiple sequence alignment S' = {s_1', s_2', s_3'} are as follows:

--GT,
GT--,
----.

The original score of these four columns of the alignment S' is (-6) + (-6) + (-6) + (-6) = -24. However, if we merge these four columns of the alignment S' as follows:

GT,
GT,
--,

then the score will become (-2) + (-2) = -4. In this case, the score of the multiple sequence set S' is increasing from 5 to 25, and the length of the alignment is decreasing from 21 to 19. Thus, we merge these four columns of the alignment and get a complete multiple DNA sequence alignment, shown as follows:

s_1' = AATTCC-T-GGATCGTCGG,
s_2' = AAT-C--T-GG-TCGTCTG,
s_3' = A-TTCGATGG-ATC--CGG.

5. Experimental results

We have implemented the proposed method using JCreator Version 2.5 [23] on a Pentium 4 PC for dealing with the multiple DNA sequence alignment problem. We use several DNA sequence sets shown in Table 1 [1, 21] as the benchmark for making the experiment. We let 1/8 of the shortest length of the sequence in the sequence set be the threshold value. In this paper, the number of generations, the population size, the crossover rate, and the mutation rate are 100, 120, 0.7 and 0.01, respectively. In the scoring process, for each column of the alignment, the match pair score, the mismatch pair score, and the space pair score are +2, -1 and –2, respectively [3]. Table 2 shows the experimental results, where the field "Score" shows the score of the complete multiple sequence alignment, and the field "Match Columns" shows the number of columns, which are completely matched. From Table 2, we can see that the performance of the multiple DNA sequence alignment of the proposed method is better than the method presented in [4] from the viewpoints of the scores and the match columns. Furthermore, because the proposed method is based on the techniques of cutting a sequence set into several subsequence sets and then aligning them individually, it provides a useful way to deal with the huge length of DNA sequences of advanced organisms for multiple DNA sequence alignment. That is, when we find a set of cut-points of the DNA sequences by the genetic algorithm, the sequence set can be cut into two subsequence sets, where the length of each sequence in the subsequence set is only about half the length of a sequence in the original sequence set. Then, the subsequence sets can be handled individually for effectively reducing the space requirement of loading huge lengths of DNA sequences into the system. Therefore, the proposed method also has the advantage of low space complexity for dealing with the multiple DNA sequence alignment problem.

Table 1. Sequence sets

Sequence set number	Number of sequences	Average length (Max, Min)
S1 [1]	10	211 (211, 211)
S2 [1]	21	122 (122, 122)
S3 [21]	5	1093 (1098, 1088)
S4 [21]	5	1092 (1142, 1006)
S5 [21]	5	1780 (1782, 1775)
S6 [21]	4	2433 (2477, 2340)
S7 [21]	8	1071 (1074, 1056)
S8 [21]	6	1456 (1463, 1430)

Table 2. Experimental results

Sequence set number	Hirschberg' method [4]		The proposed method	
	Score	Match columns	Score	Match columns
S1	17655	187	17768	193
S2	44949	101	45957	105
S3	17821	904	17912	908
S4	7884	633	11755	703
S5	32750	1608	31926	1606
S6	24256	2201	24608	2205
S7	56197	987	56972	990
S8	34635	1135	34784	1139

6. Conclusions

Multiple sequence alignment is an important research topic of bioinformatics. Its result can be used for solving many other problems in biology. However, the lengths of the DNA sequences in recent bioinformatics research are often from millions to billions. It is very important to develop a new method to effec-

tively deal with the multiple DNA sequence alignment problem. In this paper, we have presented a new method to deal with multiple DNA sequence alignment based on genetic algorithms and divide-and-conquer techniques. We have used an example to illustrate the processes of multiple DNA sequence alignment. We also have made an experiment to compare the proposed method with the method presented in [4]. The experimental results show that the proposed method is better than the method presented in [4] to deal with multiple DNA sequence alignment from the viewpoints of the scores and the match columns.

The following two research directions are worth of pursuing for future research:

(1) In this paper, in the process of calculating the degree of similarity between two sequences in the fitness function of the proposed algorithm, we aligned two sequences to the "left" by inserting the symbols "-" to the right-hand side of the shorter sequence. It is worthy that we try to align two sequences to the "right" by inserting the symbols "-" to the left-hand side of the shorter sequence.
(2) The proposed method only aligned multiple deoxyribonucleic acid sequences. It is worthy to extend the proposed method for dealing with the alignment of multiple protein and nucleic acid sequences.

Acknowledgements

This work was supported in part by the National Science Council, Republic of China, under Grant NSC 92-2213-E-011-074.

References

[1] Chellapilla, K. and Fogel, G. B. 1999. Multiple sequence alignment using evolutionary programming. *Proceedings of* the 1999 *Congress on Evolutionary Computation*, Washington D. C.: 445-452.

[2] Gen, M. and Cheng, R. 1997. "*Genetic Algorithms and Engineering Design*". John Wiley & Sons, New York, U.S.A.

[3] Gusfield, D. 1977. "*Algorithms on Strings, Trees, and Sequences*". Cambridge University Press, New York, U.S.A.

[4] Hirschberg, D. S. 1977. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24: 664-675.

[5] Holland, J. H. 1975. "*Adaptation in Natural and Artificial Systems*". MIT Press, Ann Arbor, U.S.A.

[6] Isokawa, M., Wayama, M., and Shimizu, T. 1996. Multiple sequence alignment using a genetic algorithm. *Proceedings of the Seventh Workshop on Genome Informatics*, 7: 176-177.

[7] Krane, D. E. and Raymer, M. L. 2001. "*Fundamental Concepts of Bioinformatics*". Benjamin Cummings, New York, U.S.A.

[8] Lin, C. H., Chen, S. J., and Chen, S. M. 2003. A new method for multiple DNA sequence alignment based on genetic algorithms. *Proceedings of the 2003 Joint Conference of AI, Fuzzy System, and Grey System*, Taipei, Taiwan, Republic of China.

[9] Lin, C. M. 2000. Using genetic algorithms to solve multiple sequence alignments. *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, Las Vegas, Nevada: 883-890.

[10] Man, K. F., Tang, K. S., and Kwong, S. 1996. Genetic algorithms: concepts and applications. *IEEE Transactions on Industrial Electronics*, 43: 519-534.

[11] Michalewicz, Z. 1992. "*Genetic Algorithms + Data Structure = Evolution Programs*". Springer, Berlin, Germany.

[12] Needleman, S. B. and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid

sequence of two proteins. *Journal of Molecular Biology*, 48: 443-453.
[13] Notredame, C. and Higgins, D. G. 1996. SAGA: Sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24: 1515-1524.
[14] Pevzner, P. A. 2000. "*Computational Molecular Biology: An Algorithmic Approach*". MIT Press, Massachusetts, U.S.A.
[15] Setubal, J. and Meidanis, J. 1997. "*Introduction to Computational Molecular Biology*". PWS Publishing Company, Boston, U.S.A.
[16] Stoye, J. 1998. Multiple sequence alignment with the divide-and-conquer method. *Gene*, 211: 45-56.
[17] Stoye, J., Perrey, S.W., and Dress, A. W. M. 1997. "*Winter Seminar on Molecular Biology and Biophysical Chemistry of the Cell*". Klosters, Switzerland.
[18] Thompson, J. D., Higgins, D. G., and Gibson, T. J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22: 4673-4680.
[19] Tönges, U., Perrey, S. W., Stoye, J., and Dress, A. W. M. 1996. A general method for fast multiple sequence alignments. *Gene*, 172: 33-41.
[20] Waterman, M. S. 1984. General methods of sequence comparison. *Bulletin of Mathematical Biology*, 46: 473-500.
[21] Zhang, C. and Wong, A. K. C. 1997. A genetic algorithm for multiple molecular sequence alignment. *Computer Applications in the Biosciences*, 13: 565-581.
[22] Zhang, C. and Wong, A. K. C. 1997. Toward efficient multiple molecular sequence alignment: A system of genetic algorithm and dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 27: 918-932.
[23] JCreator Version 2.5. 2000. Xinox Software. (http://www.jcreator.com/)