J Korea Industr Inf Syst Res Volume 17 Number 6
http://dx.doi.org/10.9723/jksiis.2012.17.6.041

Effective Biological Sequence Alignment Method using Divide Approach

Hae-Won Choi*, Sang-Jin Kim, and Su-Young Pi**

Abstract This paper presents a new sequence alignment method using the divide approach, which solves the problem by decomposing sequence alignment into several sub-alignments with respect to exact matching subsequences. Exact matching subsequences in the proposed method are bounded on the generalized suffix tree of two sequences, such as protein domain length more than 7 and less than 7. Experiment results show that protein sequence pairs chosen in PFAM database can be aligned using this method. In addition, this method reduces the time about 15% and space of the conventional dynamic programming approach. And the sequences were classified with 94% of accuracy.

Key Words : suffix tree, DNA sequence alignment, divide method, dynamic algorithm.

1. Introduction

The sequence alignment method that has been proposed to analyze human data and animal genetic is used for analysis of related functions or evolutional connections among sequences, domain analysis or structural prediction of protein sequences, open reading frame (ORF) analysis and scoring matrix reform for DNA sequences [1-8]. In these sequence alignment methods, one of representative methods is the pair-wise alignment method using dynamic programming, which is proposed by Needleman & Wunsch, and later modified and completed by Smith & Waterman [9-11]. When using the Needleman & Wunsch method for sequence alignment, the size of genetic data on human protein or DNA is very huge. Thus, lots of execution time and memory capacity are required due to unique characteristics of the algorithm for dynamic programming. This is a very important problem to be considered when putting the method in practice, but none of the previous studies dealt with it in terms of execution time and memory capacity.

This paper is intended to find a new method for aligning sequences. To achieve this goal, matched sequences look up in divided suffix tree according to a property of protein domain. Then a divide alignment method that appropriates for the characteristics of searched sequences will be found. The data structure, a suffix tree, used to analyze sequences, is a kind of tree including all suffixes of sequences and very effective in finding all matched sequences within linear time while traversing from root of the tree to its all leaf nodes [12-14].

The suffix tree can provides a clue or solving complicated string problems, such as a matching problem, longest common sub-string searching, all pairs' suffix-prefix searching, and circular string

* Department of Computer Engineering, Kyungwoon University (happychw@ikw.ac.kr)
** Department of Computer Engineering, Catholic University of Daegu

linearization problem, and is widely applied to various inheritance data analyses including sequence matching, DNA contamination, tandom repeating, and palindrome searching [15-19]. The method proposed in this paper was applied to a protein sequences alignment and, consequently, it could be applied to 94 pairs of protein sequences out of 100. In order to analyze performance of the proposed method, we were compared the proposed with the dynamic programming in 94% reliability. However the total processing time for alignments is improved by 15%. The remainder of this paper is organized as follows. In Section 2, a brief description of dynamic programming will be given. In Section 3, we shall present the divide alignment method in detail and in Section 4 we compare and analyze with the proposed scheme and dynamic programming. Finally, conclusion shall be given in Section 5.

2. Related Works

In this section, we will address dynamic programming and analyze the addressed problem.

2.1 Dynamic Programming

Dynamic programming is an efficient method for obtaining the last result and solving the problem to find the optimal solution. Point in the DNA sequence alignment will be added when aligning the same DNA, on the contrary, that point will be decreased when there are exchanging DNAs and void spaces in DNA. These processes repetitively are conducted to find the optimal sequence alignment.

A sub-path in the basic concept of dynamic programming is a subset of the best path, in order to obtain the best path, when optimal sub-paths connect; we can find the best optimal path. Eq.(1) represents the basic equation for dynamic programming as follows:

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(a_i b_j) \\ \max_{x \geq 1} (S_{i-x,j} - W_x) \\ \max_{y \geq 1} (S_{i,j-y} - W_y) \end{array} \right\} \quad (1)$$

i, j : index of sequence a and b
S : scoring matrix of sequence a and b
$S_{i,j}$: score of scoring matrix (i, j)
W_x gap penalty of sequence a with length x
W_j : gap penalty of sequence b with length y

In Eq.(1), $S_{i,j}$ means the highest score in scores to reach an arbitrary point (i, j) through various paths. In order to calculate $S_{i,j}$ in DNA sequence alignment, point will be added in case of the same DNA, contrarily, point will be decreased when there are exchanging DNAs or void spaces in DNA. Fig. 1 represents a movement on the matrix to compute $S_{i,j}$. In Fig.1, in order to reach at the point $S_{i,j}$ the path that satisfies with one of three conditions passes through the point, (i-1, j-1), the same column and row in the matrix. In three paths above, a diagonal movement, e.g., from (i-1, j-1) to (i, j) do not impose gap penalty, but it imposes gap penalty when the movement passed through the same column or row. Through these processes, we can choose an optimal local alignment with the highest $S_{i,j}$.

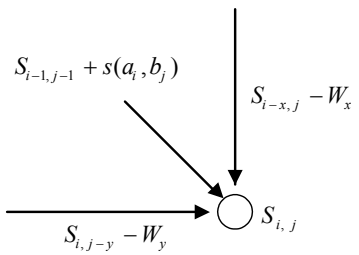


Fig. 1. the moving path to obtain $S_{i,j}$ in sequence alignment

2.2 The Problem in Dynamic Programming

Dynamic programming is efficient for sequence alignment. However, when dynamic programming is applied to very large DNA or protein sequence, alignment performance decreases. In addition, the amount of data related to biology is explosively increasing and improving performance needs.

3. Divide Alignment Method

3.1 Basic Concept of Proposed Method

This section presents a proposed alignment method to solve the problem of dynamic programming in biological application. Dynamic programming was to save execution time and memory by dividing sequences into matched parts and non-matched parts when aligning two protein sequences: *GAATTCAGTTA* and *PGGTTCAQRS*, as shown in Fig. 2, and then calculating match score of the matched parts directly using score matrix of BLOSUM62[20-21] without aligning (part A) and aligning only non-matched parts (part B).

The key point of proposed method is to find an appropriate divide alignment method according to the features of the matched sequences. To find the method, 100 pairs of sequences that are consisted of 5 pairs each selected from top 20 families with well-known domains at random among total 6,190 families of PFAM database[22-23], used in the experimentation. The proposed method is described below. Fig. 2 represents the process of a decrease in range after applying the divide alignment approach. First of all, it finds all sub-sequences matched between two sequences by searching the tree after building a suffix tree with one pair of sequences with the same domain with the one obtained from the PFAM. Let the set of subsequence is $S=\{S_1, S_2, S_3, \cdots, S_n\}$ and the set of subsequence matched between sequences by applying the dynamic programming is $T=\{T_1, T_2, T_3, \cdots, T_m\}$.

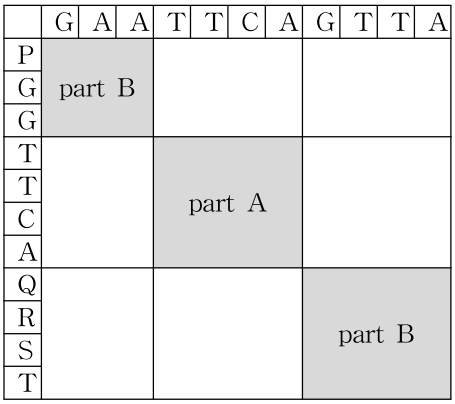


Fig. 2. A decrease in calculation range after applying the divide alignment approach

The alignment ratio, calculated according to the lengths of subsequences consisting of the set S through the experimentation, is shown in Fig. 3. If there were i numbers of the subsequences with the same lengths in the set S, and j numbers in the set T, the alignment ratio would be determined as $(j / i) \times 100$. As shown in Fig. 3, when the subsequences is very short like 1 or 2, the probability that the subsequence will be matched in dynamic programming was only 0.06. On the other hand, when the alignment ratio of the subsequences is less than 4 to 7 in length, the ratio increased exponentially.

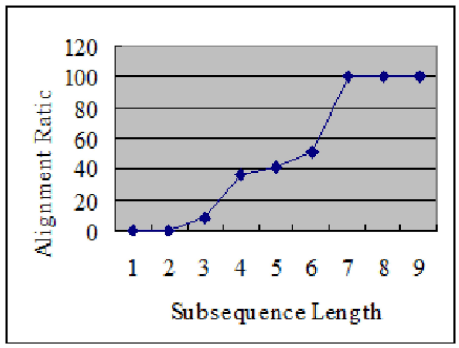


Fig. 3. The alignment ratio according to the lengths of subsequences

Additionally, it can be also known that the subsequences more than 7 long are always matched during alignment. Therefore, among all sequences

found by traversing the suffix tree from root to leaf nodes, it can be seen that the subsequences more than 7 long can be chosen as a divide-point, the basis point that sequences are divided.

This paper suggests divide alignment methods for two cases, subsequence length difference method and match distance difference method, to divide sequences.

3.2 Subsequence Length Difference Method

As shown in Fig. 3, the first method is to determine the subsequence as a divide-point if there exist more than one subsequence with more than 7 long after examining the subsequences matched among sequences.

However, the experimentation analysis showed that among the protein sequences with the same domains, sequences with aligned subsequences with more than 7 long were 39 pairs out of 100 (39%). Consequently, another method will be required to find a divide-point appropriate to the remaining 61 pairs (61%).

3.3 Match Distance Difference Method

The match distance difference method, there were no subsequences matched between sequences more than 7 in length, calculate the difference in distances between matched subsequences during sequence alignment. If the starting location of matched sequences set i and j, then the match distance difference(Δ) between i and j is represented as Eq.(2) and Fig.4.

$$\Delta = |(i_2 - i_1) - (j_2 - j_1)| \qquad (2)$$

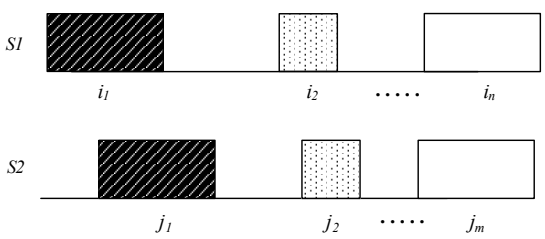


Fig. 4. Match distance difference between sequences

Fig. 5 showed the mean values of Δ between protein family sequences. The mean values of matched subsequences may be different although the sequences consist of the same family. This results from a us similarity of domains lengths and the characters according to sequences composing each family.

As a result of the experimentation shown in Fig. 5, the mean value of Δ between matched subsequences of the S_1 and S_2 was 8, and this value would be used as the basis value for the match distance difference method.

Like this way, if the Δ is calculated based on all matched subsequences between two sequences and the result is less than 8, it will be selected as a candidate for sequence division. If there were no subsequences that the value of Δ is less than 8 among subsequences matched between sequences the S_1 and S_2 , the sequences wouldn't be divided.

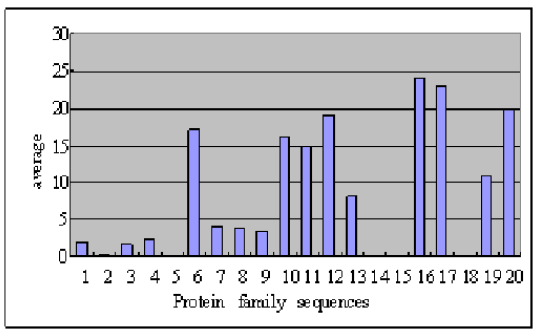


Fig. 5. The mean value of Δ between protein family sequences

Now, the example of applying two methods above to protein domain sequence will give and analyze.

3.3.1 Case 1: Protein domain that the length of subsequence is more than 7

An alignment result of two protein sequences, S_1 and S_2, with the domain gp120[20] that were applied to the dynamic programming is shown in Fig. 6. The part B in quadrangle shape, has the length of subsequence is more than 7.

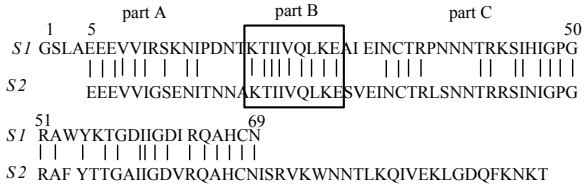
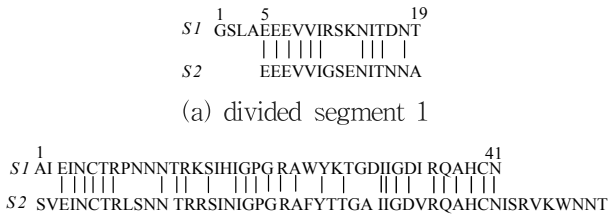


Fig. 6. The gp120 domain sequences alignment without using the divide alignment method

As result of proposed method, the sequence S_1 and S_2 were divided and then each divided part was aligned, as shown in Fig. 7.



(a) divided segment 1

(b) divided segment 2

Fig. 7. The gp120 domain sequences alignment after being applied to the divide alignment methods

3.3.2 Case 2: The length of subsequence is less than 7

The sequences with *Cytochrome B* domain in Fig.8 were aligned result using the dynamic programming. In this case, since there are subsequences with the length of less than 7, sequences S_1 and S_2 are divided based on the subsequences with Δ value less than 8. A divide-point was indicated using a square in Fig. 8.

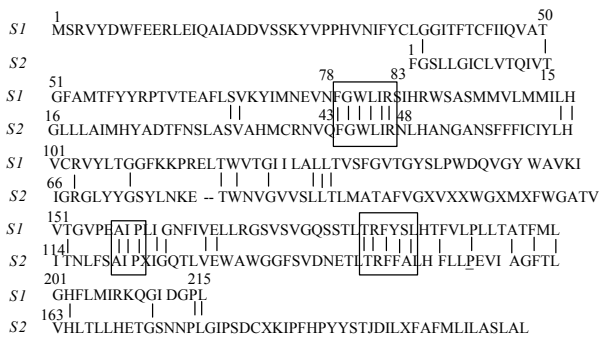
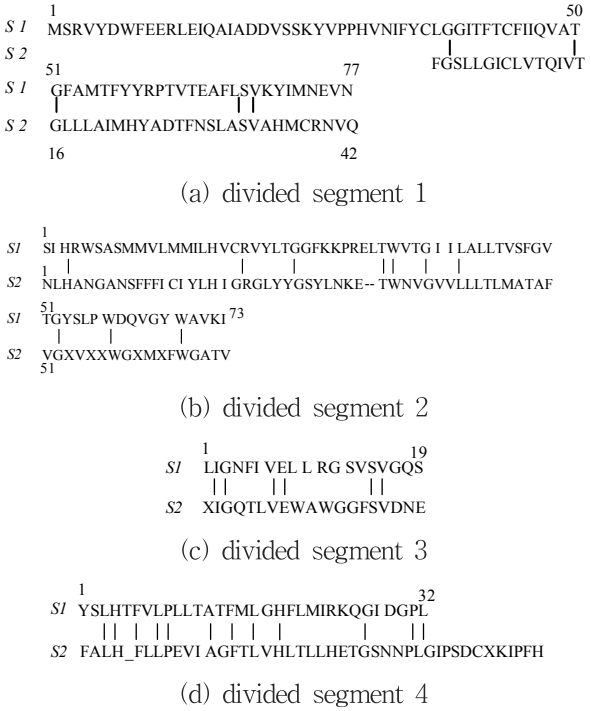


Fig. 8. The *Cytochrome B* domain sequences alignment without using the divide alignment method

The result of sequences in Cytochrome B domain in Fig. 9 were divided and then aligned using the dynamic programming alignment method. The key idea of divide approach is to consider characteristics of protein sequence alignment.



(a) divided segment 1

(b) divided segment 2

(c) divided segment 3

(d) divided segment 4

Fig. 9. The *Cytochrome B* domain sequences alignment after using the divide alignment method

3.4 Similarity Measurement Analysis

Let the score of similarity between sequences in the existing dynamic programming is S_n and the score of similarity by the divide alignment method is S_p. To examine the difference in similarity between S_n and S_p, the difference ratio P was defined as Eq.(3) and the experimentation was conducted. When there is no difference between S_n and S_p, P becomes 0.

$$P = \frac{S_p - S_n}{S_n} \tag{3}$$

Table 1. The number of sequences according to divide alignment methods and comparison of difference ratio

Method	Divided Sequences Number	Difference Ratio
Subsequence Length Difference Method	39	0.01
Match Distance Difference Method	57	0.63
Sum/Average	96 (Sum)	0.37 (Average)

Table 1 is the result of the difference ratio and means the divided number of sequences according to divide alignment methods and comparison of difference ratio. The difference ratio of the sequences using subsequence length and difference method is similar to the dynamic programming method (0.01).

Comparing with this, the difference ratio of the sequences using the match distance difference method is relatively high by 0.63. This is because the subsequences in match distance difference method are relatively short, while the ones in subsequence length difference method are more than 7 long. In other words, the part where the subsequences are long is more likely to match also in the dynamic programming method. Paying attention on this, modification proposal for the match distance difference method has been suggested to minimize the difference between difference ratios.

The key idea of modification proposal is to constitute sets in order that all subsequences with less than 8 of match distance difference are matched, and then search all more than fixed lengths among them and use them as divide-points, instead of dividing subsequences unconditionally when the match distance difference is less than 8. This is to give limit to the sum total of lengths of subsequences.

In the sequences, S_1 and S_2, in Fig. 10, the match distance difference of the strings, designated as i and j , was matched less than 8. Index i and j represent the lengths of Δi and Δj each, respectively. But, i and j have sequential values in the order that they are matched. *lim_i* and *lim_j* have cumulative values of Δi and Δj, that is, the sum total of lengths of matched subsequences respectively.

w limits a divide-point of sequences. A limit-window is bounded to w. Using w, the match distance difference method with w is operated as follows. Starting from the first matched subsequence of sequences, the values of Δi and Δj are compared with predetermined values of w, increasing i and j and searching the sequences matched in order. If the values of Δi and Δj are smaller than those of w, those of Δi and Δj are accumulated to *lim_i* and *lim_j*.

This process repeats until there comes out a set of all subsequences satisfying the value of *lim_i* and *lim_j*. When the total lengths of subsequences satisfies w, corresponding subsequences form a set bounded to limit-window and use these many numbers of sets as divide-points. Table 2 is the number of sequences divided using the match distance method and change in difference ratios.

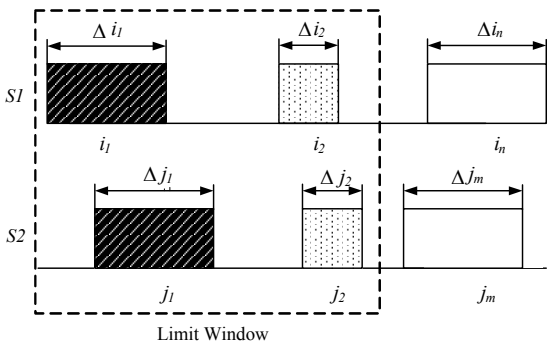


Fig. 10. match distance difference method with limit-window

Table 2. The number of sequences divided using the match distance difference method and difference ratios

Limit	Divided Sequences Number	Difference Ratio
150	57	0.54
100	57	0.49
50	57	0.38
40	57	0.37
30	54	0.41
20	50	0.39
10	10	0.25

As shown in Table 2, the match distance difference method showed better difference ratio than the method without limits. In the case that the limit is over 40, the number of divided sequences were same with that of previous method, but the difference ratio is improved. Based on the results shown in Table 2, we choose 40 as the most optimal limit value that the number of divided sequences is maximized and the difference ratio is the most adequate.

3.5 Divide Alignment Approach Process According to Step

This section describes the entire process of applying two divide alignment approach according to steps after finding all subsequences matched between two sequences using a suffix tree.

Step 1: Suffix Tree Construction

When constructing a suffix tree, all internal nodes excepting for leaf nodes have indexes indicating the start and end of characters on edges. Each internal node of a suffix tree consists of three fields:

- the number of a sequence : *seq_num*
- the beginning of a string consisting of an edge: *start_index*
- the last index of a string consisting of an edge: *end_index*

Step 2: Making a Match List Using Suffix Tree Search

When a suffix tree is constructed using two protein sequences, each internal node of the suffix tree has different numbers. After a suffix tree is built, the subsequences more than 3 long in all matched subsequences by traversing the tree finds. The searched subsequences are stored on the match list after getting the number of a sequence.

Fig.11 is an example of constructing a suffix tree of two protein sequences, *AAAAPSCCC*(S_1) and *AAPPACCC*(S_2). It can be seen that there exist matched subsequences more than 3 long on the 2nd and 7th nodes. In this case, storing the information of subsequences in a match list is exampled in Fig.12. In Fig 12, *ml* indicates matched length between two sequences and *si* indicates the start location of a suffix, and *no* indicates the number of a node.

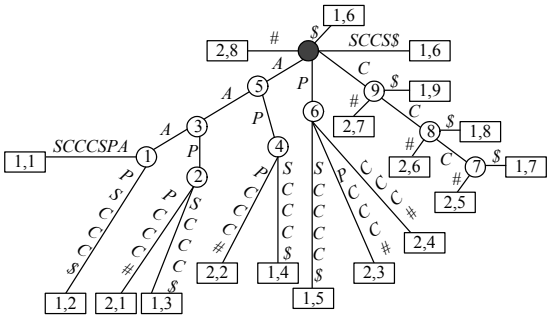


Fig. 11. An example of traversing a suffix tree with *AAAAPSCCC*(S_1) and *AAPPACCC*(S_2)

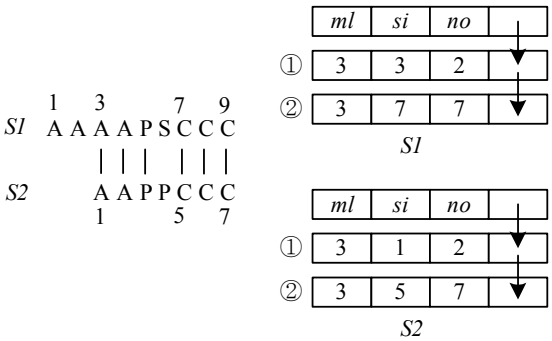


Fig. 12. match list with *AAAAPSCCC*(S_1) and *AAPPACCC*(S_2)

Step 3: Making a Linked List

In a suffix tree, the path from root to a random internal node represents the subsequences matched between two sequences, and these subsequences have the same node number. Therefore, steps for making a linked list of the subsequences with the same number from different match lists are as follows.

1) Form an adjacent list between the subsequences with the same node numbers from the match list.
2) The subsequences with the same node numbers from the match lists make a linked list.

Step 4: Subsequence Length Examination

The lengths of matched subsequences in the linked list are examined one after another to examine if there are subsequences matched between two sequences more than 7 long.

Step 5: Application of a Divide Alignment Method

As earlier mentioned, if there are more than one subsequence over 7 long, the subsequence length difference method will be used. If there are no such subsequences, the match distance difference method with limits(40) will be applied. If Δ subsequences are less than 8, the sequences would not be divided.

Step 6: Similarity Measurement

Similarity is measured in two ways. The first way in Fig.13 directly gets a match score using a score matrix of BLOSUM62 without alignment, when the subsequences are selected as divide-points of two sequences. The second one obtains a similarity score between sequences by applying the dynamic programming method to the remaining part of divided sequences, as shown in part B of Fig.13. Finally, the entire similarity between sequences is measured by adding the scores of parts A and B.

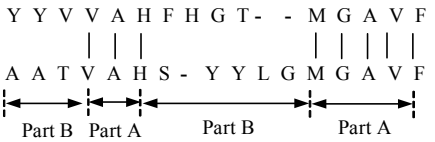


Fig. 13. Divide measurement of similarity

4. Classification Analysis

In this experiment, we actually implemented the previous methods and compared with them. Simulation is conducted on 4GB main memory using Intel i5 2.5GHz CPU. Data in the experimentation was selected from 100 hundred pairs of sequences with the same domain and 200 pairs of sequences with different domains at random. Simulation showed the classification accuracy of sequences with a similarity score obtained by the divide method, which has been presented. In the experimentation, when the similarity score of sequences with the same domain was higher than that of the sequences with different domains, it was concluded that classification was more accurate.

Table 4 shows the comparison of the classification results via the divide alignment method presented in this paper and via the dynamic programming method. In Table 4, the number of pairs of divided sequences using a subsequence length difference method were 40 in total and they were all classified accurately. The dynamic programming method classified 94 pairs(94%) out of 100 pairs and made 6 pairs of error. On the contrary, the proposed scheme exactly classified 94 pairs out of 100 pairs in sequence and compared with dynamic programming method. The total alignment time outperformed by 15% approximately due to the divide approach based on protein domain property. Both the dynamic program and the proposed scheme took 6 pairs errors as the

range of domain is very short and there exists short domain between subsequences.

Table 4. Comparison of the classification results via proposed divide alignment method and dynamic programming method

Method	Divide Alignment		Dynamic program method
	Divide alignment using subsequences length	Divide alignment using segments	
Classify Sequence Number	40 / 40	54 / 60	-
	94 / 100		94 / 100
Process Time	3.57 sec.		4.2 sec.

5. Conclusion

This paper proposed a new alignment method using the divide approach in order to meet much execution time and requirements in memory capacity for application, such as dynamic programming. In order to show the perform of the proposed scheme, this paper used protein sequences to expand analyze the execution time, memory capacity and divide-point parameters of the DNA with much larger data. We show that this new scheme reduce alignment time about 15% then dynamic programming method. And the sequences were classified with 94% of accuracy. Thereby, the proposed method can provide effective alignment approach to much larger protein sequence.

Reference

[1] David W., Bioinformatics, sequences and Genome Analysis, MOUNT Press, 2001.

[2] Younshin Oh, Dinh Truong Nguyen, "Identification of 1,531 cSNPs from Full-length Enriched cDNA Libraries of the Korean Native Pig Using in Silico Analysis," Genomics & Informatics, vol. 7, no. 2, 2009, pp. 65-84.

[3] Audry P. G., Alan M.M., "Conservation and Evolution of Cis-Regulatory Systems in Ascomycete Fungi," PLOS Biology, vol. 2, no. 12, 2004, pp. 398-405.

[4] Josue Samayoa1, Fitnat H. Yildiz and Kevin Karplus, "Identification of prokaryotic small proteins using a comparative genomic approach," Bioinformatics, vol.27, no.13, 2011, pp.1765-1771.

[5] Chan Park, Ji-Seong Jeong, "Design and Implementation of Bio-Medical Data Measurement System through the Stereo Microscope," Korea Contents Association KISTI-KOCON ICCC2009, November, vol.7, no.2, 2009, pp.357-360.

[6] Young-Ohk Song, Sung-young Kim and Duk-Jin Chang, "Design of the System and Algorithm for the Pattern Analysis of the Bio-Data," Korea Contents Association, November, vol.10, no.8, 2008, pp.104-110.

[7] 이성열, "A Modified Heuristic Algorithm for the Mixed Model Assembly Line Balancing ," 산업정보학회논문지, vol.15, no.3, 2010, pp.51-57.

[8] 유기동, "문서 자동요약 기술을 적용한 클라우드 스토리지 기반 지능적 아카이빙 시스템," 산업정보학회논문지, vol.17, no.3, 2012, pp.59-68.

[9] P. Agarwal, "Comparative accuracy of methods for protein sequences similarity search," Bioinformatics, vol.14, no.1, 1998, pp.40-47.

[10] X. Guan and L. Du, "Domain identification by clustering sequences alignment," Bioinformatics, vol.14, no.9,1998, pp.783-788.

[11] D. Gusfield, Algorithms on strings, trees, and sequences : Computer science and Computational biology, CAMBRIDGE University Press, 1997.

[12] Data Structure and Algorithm: Tree and Suffix trees, Mcgill University ,1997.

[13] J. Karkkainen and E. Ukkonen, "Sparse Suffix Tree," COCOON , 1996, pp.219-233.

[14] E. Ukkonen, "On-line Construction of Suffix-Trees," Algorithmica, vol.14, 1995, pp.249-260.

[15] Mark Nelson, "Fast String Searching With

Suffix Trees," Dr. Dobb's Journal, 1996.
[16] M.I. Abouelhoda, S. Kurtz, and E. Ohkebusch, "Replacing suffix trees with enhances suffix arrays," Journal of Discrete Algorithms, vol. 2, no. 1, 2004, pp. 53-86.
[17] D.K.Kim, M.Kim, and H.Park, "Linearized suffix tree: an efficient index data structure with the capabilities of suffix trees and suffix arrays," Algorithmica, vol. 52, no. 3, 2008, pp. 350-377.
[18] L.Russo, G.Navarro, and A.Oliveria, "Fully-Comoressed suffix trees," LATIN, 2008, pp. 362-373.
[19] Batzoglou, S., Pachter, L., Mesirov, J. P., Berger, B. and Lander, E. S. "Human and mouse gene structure: comparative analysis and application to exon prediction," Genome Research, vol. 10, 2000, pp. 950 - 958.
[20] Sean R. Eddy, "Where did the BLOSUM62 alignment score matrix come from?," Nature Biotechnology, vol.22, 2004. pp.1035-1046.
[21] I. Mihalek1, I. Res and O. Lichtarge, "Background frequencies for residue variability estimates: BLOSUM revisited," BMC Bioinformatics, vol.8, 2007, pp.488-498.
[22] Alex Bateman, "The PFAM Protein Family Database," Nucl. Acids Res. vol. 30, no. 1, 2002, pp. 276-280
[23] Marco Punta1,Penny C. Coggill, "The Pfam protein families database," Nucleic Acids Research, November, 2011, pp.1-12.

최 해 원 (Hae-Won Choi)

• 정회원
• 1996년 2월 : 경일대학교 컴퓨터공학과 (공학사)
• 2000년 2월 : 경북대학교 컴퓨터공학과 (공학석사)
• 2009년 2월 : 경북대학교 컴퓨터공학과 (공학박사)
• 2006년 3월 ~ 현재 : 경운대학교 컴퓨터공학과 조교수
• 관심분야 : Bioinformatics, 유비쿼터스 컴퓨팅

김 상 진 (Sang-Jin Kim)

• 1994년 2월 : 계명대학교 컴퓨터공학과(공학사)
• 1996년 2월 : 경북대학교 컴퓨터공학과(공학석사)
• 2000년 8월 : 경북대학교 컴퓨터공학과(공학박사)
• 1999년 9월~현재 : 경운대학교 컴퓨터공학과 교수
• 관심분야 : 알고리즘, 게임이론

피 수 영 (Su-Young Pi)

• 1987년 2월 : 대구가톨릭대학교 전산통계학과(이학사)
• 1989년 2월 : 대구가톨릭대학교 전산통계학과(이학석사)
• 2000년 2월 : 대구가톨릭대학교 전산통계학과(이학박사)
• 2000년 7월 ~ 2012년 2월 : 대구가톨릭대학교 실용컴퓨터 책임교수
• 2012년 3월~현재 : 대구가톨릭대학교 교양교육원 교수
• 관심분야 : 지능형 시스템, 퍼지이론 및 응용

논 문 접 수 일 : 2012년 10월 05일
1차수정완료일 : 2012년 10월 25일
2차수정완료일 : 2012년 11월 14일
게 재 확 정 일 : 2012년 11월 14일