Korean J. Chem. Eng., **22**(3), 345-352 (2005)

A Gene Clustering Method with Masking Cross-matching Fragments Using Modified Suffix Tree Clustering Method

Sang Il Han, Sung Gun Lee, Bo Kyeng Hou, Sunghoon Park, Young Han Kim* and Kyu Suk Hwang†

Department of Chemical Engineering, Pusan National University, Busan 609-735, Korea
*Department of Chemical Engineering, Dong-A University, Busan 604-714, Korea
(*Received 17 January 2005 • accepted 16 March 2005*)

Abstract−Multiple sequence alignment is a method for comparing two or more DNA or protein sequences. Most multiple sequence alignment methods rely on pairwise alignment and Smith-Waterman algorithm [Needleman and Wunsch, 1970; Smith and Waterman, 1981] to generate an alignment hierarchy. Therefore, as the number of sequences increases, the runtime increases exponentially. To resolve this problem, this paper presents a multiple sequence alignment method using a parallel processing suffix tree algorithm to search for common subsequences at one time without pairwise alignment. The cross-matched subsequences among the searched common subsequences may be generated and those cause inexact-matching. So the procedure of masking cross-matching pairs was suggested in this study. The proposed method, improved STC (Suffix Tree Clustering), is summarized as follows: (1) construction of suffix tree; (2) search and overlap of common subsequences; (3) grouping of subsequence pairs; (4) masking of cross-matching pairs; and (5) clustering of gene sequences. The new method was successfully evaluated with 23 genes in *Mus musculus* and 22 genes in three species, clustering nine and eight clusters, respectively.

Key words: Multiple Sequence Alignment, Clustering, Sequence, Gene

INTRODUCTION

The DNA and protein data of diverse species have been daily discovered and deposited in the public archives according to each established format [Ostell et al., 2001; Chen and Carlis, 2003]. Those databases provide not only an easy-to-use, flexible interface to the public, but also *in silico* analysis tools of unidentified sequence data [Mount, 2001; Salzberg et al., 1998].

Sequence alignment can be used to study the relationships among sequences in sets of two or more sequences. It is particularly useful when studying the relationship of similar types of gene products that are expressed by different organisms, or when studying similar, yet divergent, sequences within the same organism. Often, a primary purpose of a multiple sequence alignment is to identify, within several related sequences, the regions that are highly conserved, and therefore probably have functional and structural relatedness. The multiple sequence alignment of a set of sequences may also be viewed as an evolutionary history of the sequences [Phillips et al., 2000; Mount, 2001]. And the clusters of sequences present a basis to deposit systemically huge data.

Being based on the pairwise alignment and progressive method, most multiple alignment methods such as CLUSTAL [Higgins and Sharp, 1988; Thompson et al., 1994; Higgins et al., 1996], DIALIGN [Morgenstern et al., 1998] and SAGA [Notredame and Higgins, 1996] are not effective in comparing large number of data sets. Also, dynamic programming algorithms [Lee and Lee, 2004; Kim et al., 2004] such as the Smith-Waterman algorithm using sensitive pairwise comparison are quite exhaustive [Pearson and Miller, 1992].

The suffix tree algorithm can be appropriate to deal with huge genomic data in linear time [Ukkonen, 1995; Gusfield, 1997; Zamir and Etzioni, 1998]. The suffix tree clustering (STC) method was first introduced by Zamir et al. [1997] and presented to cluster web-documents [Zamir and Etzioni, 1998]. Zamir and Etzioni [1998] compared the execution times (Fig. 1) on the number of snippets

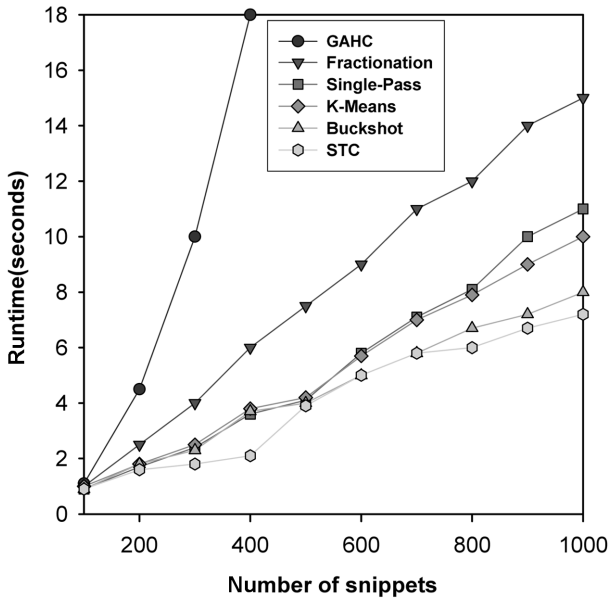

Fig. 1. The runtime comparison of the several different clustering algorithms on snippet collections as a function of each collection size.

†To whom correspondence should be addressed.
E-mail: kshwang@pusan.ac.kr
‡This paper was prepared at the 2004 Korea/Japan/Taiwan Chemical Engineering Conference held at Busan, Korea between November 3 and 4, 2004.

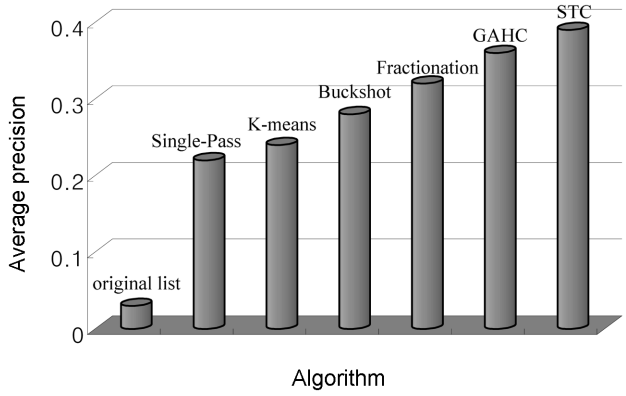

Fig. 2. The average precision comparison of the several different clustering algorithms and the original ranked list.

as a function of the collection size and the precisions (Fig. 2) in different clustering algorithms. As shown in Figs. 1 and 2, the STC (Suffix Tree Clustering) method is faster and more precise than other standard clustering methods such as Single-Pass, K-means, Buckshot, Fractionation and GAHC (Group-average Agglomerative Hierarchical Clustering). Delcher et al. [1999] made MUMmer software [Delcher et al., 1999, 2002; Hon and Sadakane, 2002] to find MUMs (Maximal Unique Matching subsequence) between two genomes of related species, and Volfovsky et al. [2001] made a new rice repeat database to rapidly search the repeats in the genome by using the suffix tree. They utilized the suffix tree algorithm to find only common subsequences. But our method not only finds common subsequences, but also clusters similar sequences. Kalyanaraman et al. [2002] have developed a parallel EST clustering program, which enables us to find the shared maximal common substrings by suffix tree algorithm and to discover the sequence pairs sharing a maximal common substring of length greater than or equal to a threshold value. But in case of gene [Shin et al., 1996] sequence, it is not appropriate to compare sequence similarity among gene sequences without cross-matching masking, because the sequences should be sequentially matched. To resolve the above problem, we presented the step of masking cross-matching fragments.

In the present paper, we have introduced an improved gene clustering method based on the suffix tree algorithm. The STC developed by Zamir et al. [1997, 1998] was used to search for effectively common subsequences and cluster genes with the suffix tree, which creates clusters based on common subsequences (strings shared among sequences).

The gene clustering program was developed with Perl (Practical Extraction and Report Language) [Randal and Christiansen, 1997; Tisdall, 2001] and used to search for common subsequences and cluster similar gene sequences. The feasibility of this program was evaluated by using twenty-three genes in *Mus musculus* species and twenty-two genes in three other species retrieved from the Homologene database of NCBI.

METHOD

The aim of this paper is to present an improved gene clustering method based on the STC, which tabulates the position information and searches for common subsequences shared in several gene sequences. The STC is an incremental linear time algorithm that constructs clusters based on common substrings, and is known to be faster than other standard clustering methods [Zamir et al., 1997; Zamir and Etzioni, 1998]. The procedure of web document clustering by Zamir and Etzioni [1998] is as follows. (i) document cleaning (the string of text representing each document is transformed); (ii) identifying base clusters (searching for sets of document sharing common phrase); and (iii) combining base clusters (merging base clusters with a high overlap).

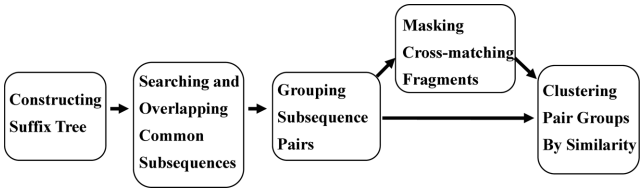

Fig. 3. The scheme of our gene clustering system.

In the present study, the original STC of Zamir et al. [1997] was modified to cluster similar genes of DNA sequences. In the case of DNA sequences, document cleaning was not required. However, it was necessary to divide the last step which combines the base clusters into two steps (grouping the common subsequence pairs and clustering the common subsequence pair groups), since the common subsequences have to be sequentially matched [McCreight, 1976] and the cross-matching subsequences of long sequences should be avoided.

The modified method of gene clustering with a suffix tree can be illustrated by a simple example using four sequences (ATGCA, ACGCA, TAATC, TACTC). Fig. 3 shows the scheme of our gene clustering tool.

1. Constructing the Suffix Tree

The suffix tree algorithm is a data structure algorithm that can be used to solve the exact matching problem in linear time. The linear-time suffix tree algorithm was first introduced by Weiner [1973], and then a different, more space-efficient algorithm to build suffix trees in linear time was given by McCreight [1976] later. Ukkonen [1995] developed a conceptually different linear-time algorithm for building suffix trees, which has all the advantages of McCreight's algorithm. We used Ukkonen's easy-to-understand, space-efficient suffix tree algorithm which is faster than Weiner's method. The suffix tree by Ukkonen has the major features as follows [Ukkonen, 1995; Gusfield, 1997; Zamir and Etzioni, 1998].

(1) The suffix tree is a rooted, directed tree with all suffixes of a set of strings.
(2) Each internal node has at least two children nodes.
(3) For the sequence of m length, the tree has 1~m branches.
(4) Subsequences are labeled on the branches of a suffix tree.
(5) The labels on branches are the suffixes of sequences.
(6) The branches from the same node have not the same label.

Fig. 4 shows a branching configuration of the suffix tree for the four sequences mentioned before. To construct the suffix tree for more than two sequences, terminal symbol $ was added to the end of each sequence, and sequences were concatenated. The suffix tree made in this way is called generalized suffix tree [Gusfield, 1997].

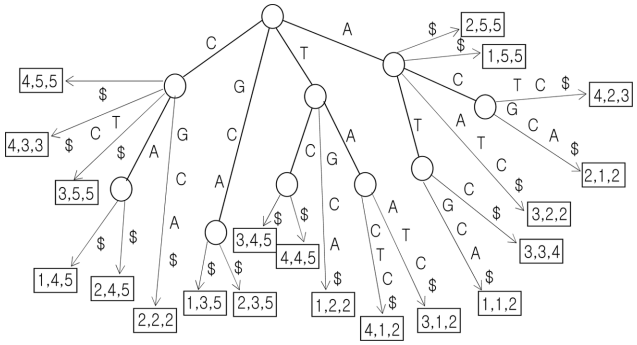

Fig. 4. The suffix tree for sequences (1. ATGCA; 2. ACGCA; 3. TAATC; 4. TACTC).

In Fig. 4, the suffix tree consists of 10 nodes including root node. The numbers in the tetragon of the end of the suffix tree represent a sequence number including strings between two nodes, a starting position of the string, and an ending position of the string on the sequence in due order.

2. Searching the Common Subsequences

The label, except for terminal symbol $, between nodes in the suffix tree represents the common subsequence shared in more than two sequences. Based on such common subsequences, the similarity of sequences was compared and similar sequences were clustered. To reduce the inefficiency caused by unrelated and useless fragments, our method selects the minimum block size (system-acceptable the minimum length of subsequence) according to total sequence size. Here, the block size was set to 2 due to short sequence length. Table 1 shows the position information of common subsequences using block size 2 which have been reduced by eliminating the position information with the same starting and ending number.

3. Overlapping the Common Subsequences

As the suffix tree algorithm finds all the common subsequences in its sequences, the overlapped subsequences including other subsequences can be generated. The overlapped subsequences are inefficient in the viewpoint of runtime, so the subsequences which are found to be overlapped should be merged into the larger subsequence, based on the position information. The rule to check the overlap of the common subsequences is as follows.

Table 3. The pair groups for the common subsequences in Table 2

Pair groups	Position information
Sequence 1 and sequence 2	{(1,3,5), (2,3,5)}
Sequence 3 and sequence 4	{(3,1,2), (4,1,2)}, {(3,4,5), (4,4,5)}

Position information; (n_1, a_1, b_1), (n_2, a_2, b_2)
If n_1 == n_2 and {(a_1 <= a_2 and b_1>=b_2) or (a_1 >= a_2 and b_1<=b_2)} **then**
Overlapping

※ n_1, n_2 ; sequence number
a_1, a_2 ; starting position
b_1, b_2 ; ending position

Table 2 shows the position information of the subsequences that exist in the each sequence before and after overlapping the subsequences, based on Table 1.

4. Grouping the Common Subsequence Pairs and Masking Fragments

The common subsequence pairs which exist commonly in different sequences of Table 2 were grouped and two groups are shown in Table 3. The rule grouping the common subsequence pairs is as follows.

Position information; (n_1, a_1, b_1), (n_2, a_2, b_2)
If n_1≠n_2 and {{string(n_1, a_1, b_1)⊃string(n_2, a_2, b_2)} or {string(n_1, a_1, b_1)⊂string(n_2, a_2, b_2)}} **then**
Grouping common subsequence pairs

Once the number of the common subsequences in the pair groups increase, the cross-matching subsequences that cause inexact-matching may be generated. Hence, based on the long length subsequence, the cross-matching fragments were masked to match common subsequences sequentially [Delcher et al., 1999]. Fig. 5 shows the diagram of this filtering procedure. For the example sequences, due to short length the cross-matching fragments were not presented and this procedure was not applied.

If the size of the longest common subsequence in a pair group excesses the threshold size, the pair group is considered to be similar and is directly transferred to the step of clustering the common sub-

Table 1. The common subsequences at block size of 2 in the suffix tree of Fig. 4

Common subsequences	Common subsequences at block size 2
(4,5,5) (4,3,3) (3,5,5) (1,4,5) (2,4,5)	(~~4,5,5~~) (~~4,3,3~~) (~~3,5,5~~) (1,4,5) (2,4,5)
(2,2,2) (1,3,5) (2,3,5) (3,4,5) (4,4,5)	(~~2,2,2~~) (1,3,5) (2,3,5) (3,4,5) (4,4,5)
(1,2,2) (4,1,2) (3,1,2) (1,1,2) (3,3,4)	(~~1,2,2~~) (4,1,2) (3,1,2) (1,1,2) (3,3,4)
(3,2,2) (2,1,2) (4,2,3) (1,5,5) (2,5,5)	(~~3,2,2~~) (2,1,2) (4,2,3) (~~1,5,5~~) (~~2,5,5~~)

Table 2. The common subsequences before and after the overlapping

Number of sequences	Before overlapping	After overlapping
Sequence 1	(1,1,2), (1,3,5), (1,4,5)	(1,1,2), (1,3,5), (~~1,4,5~~)
Sequence 2	(2,1,2), (2,3,5), (2,4,5)	(2,1,2), (2,3,5), (~~2,4,5~~)
Sequence 3	(3,1,2), (3,3,4), (3,4,5)	(3,1,2), (3,3,4), (3,4,5)
Sequence 4	(4,1,2), (4,2,3), (4,4,5)	(4,1,2), (4,2,3), (4,4,5)

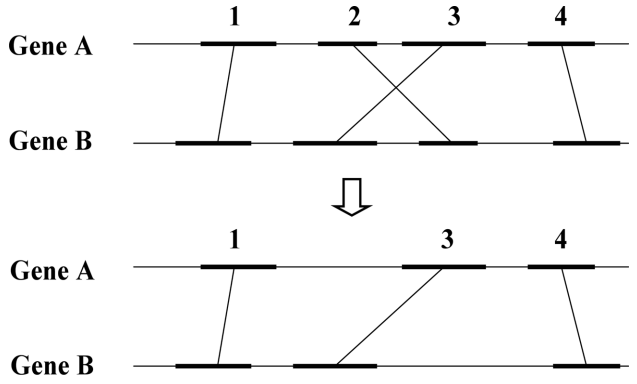

Fig. 5. Aligning Gene A and Gene B after eliminating cross-matching subsequences.

sequence pair group without masking the cross-matching subsequences. Here, the threshold size was set to 40 as an optional value based on the size and type of the sequence. The rule of masking cross-matching subsequence is as follows.

Position information; pair 1 {(n_1, a_1, b_1), (n_2, a_2, b_2)}
pair 2 {(n_1, a_3, b_3), (n_2, a_4, b_4)}
…
in two sequences (n_1 and n_2)
If the size of pair 1 is the longest, the pair 1 is the basic pair.
Using the basic pair (pair 1)

If ((a_1>=b_3 and a_4>=b_2) or (a_3>=b_1 and a_2>=b_4)) **then**
Masking cross-matching pair

---

5. Clustering the Common Subsequence Pair Groups by Similarity

Next, for each common subsequence pair group, the pair groups having a similarity under a setting value are considered unrelated sequences to be eliminated. Here, the similarity is the proportion of the common subsequences existing on the sequences in study. If the setting value is too small or large, the sequences can be clustered into false groups or not be clustered. Since the lengths of the example sequences (ATGCA, ACGCA, TAATC, TACTC) are short, the data are clustered by similarity criteria with 50%.

After removing the pair groups with no threshold similarity, in order to collect similar sequence pair groups, the pair groups were compared to each other and clustered when the sequences in any

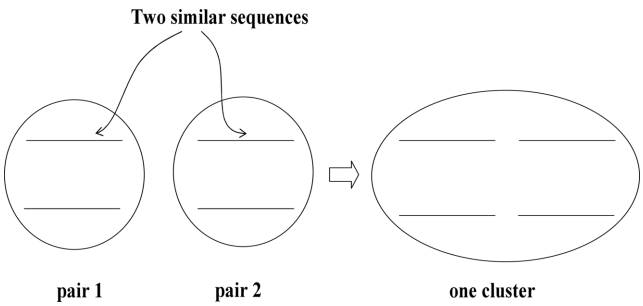

Fig. 6. The simplified diagrammatic illustration of joining pair groups.

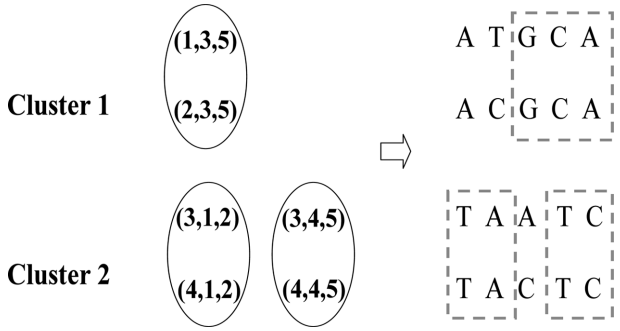

Fig. 7. The two clusters for the sequences (ATGCA, ACGCA, TAATC, TACTC).

pair group are similar to the sequences in another pair group. This rule showed suitable results for real DNA sequences. For the example sequences, the sequence pairs did not join to each other. Fig. 6 shows the way how to join the pair groups. The rule for clustering the common subsequence pair groups is as follows.

Sequence pairs; pair 1 (n_1, n_2), pair 2 (n_3, n_4)
…
If {n_1 == (n_3 or n_4)} or {n_2 == (n_3 or n_4)} **then**
Clustering pair 1 and pair 2

Based on the position information such as {(1,3,5), (2,3,5)} and {(3,1,2), (4,1,2), (3,4,5), (4,4,5)}, two clusters, (sequence 1 and sequence 2) and (sequence 3 and sequence 4), were formed, as shown in Fig. 7.

Fig. 7 shows that the sequences are clustered by numerical information only, without relying on character data. The unmatched strings between common subsequences represent insertion or deletion, and the matched strings represent the significant conserved regions.

RESULTS AND DISCUSSION

The suffix tree algorithm proposed in this study was applied to two cases: (i) twenty three gene sequences of *Mus musculus* species, and (ii) twenty two gene sequences of different species. The data were obtained from the Homologene database of NCBI. A PC with Intel's Pentium 2.4 GHz processor, 1 GB RAM, and Linux OS was used for analysis.

1. Case Study 1: Twenty Three Gene Sequences of *Mus musculus* Species

Table 4 shows the 23 genes in the *Mus musculus* (house mouse)

Table 4. The 23 Genes of the *Mus musculus* species

Species	Accession number
Mus musculus	XM_204449, XM_289927, NM_027609, XM_355690, XM_203409, XM_356889, XM_357648, XM_357087, XM_356880, NM_021300, XM_356386, XM_109566, NM_013548, NM_175653, NM_178215, XM_358107, XM_358117, NM_027650, NM_173069, XM_355572, XM_357595, XM_142567, XM_355567

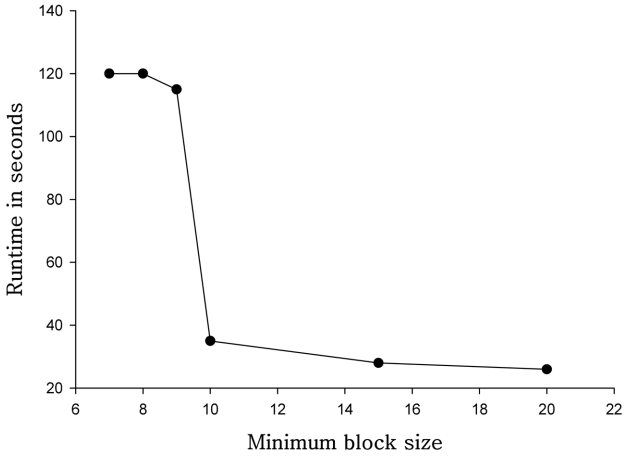

Fig. 8. The relationship between runtime and minimum block size in Case Study 1.

species retrieved from the Homologene database of NCBI.

The gene data have a long length of 700-2,000 bp. The minimum block size for searching common subsequences was set to 10-20 bp and the similarity for clustering common subsequence pair groups was set to 20%. Each gene sequence was concatenated to a long sequence by adding the terminal symbol $ to the end of each sequence, and then the suffix tree was constructed. The runtime comparison on the variation of minimum block size is shown in Fig. 8.

As the block size decreases, the runtime increases exponentially. On the other hand, when the block size is large, sequences having short length common subsequences are sometimes not included into clusters. At the minimum block size of 10, most genes were properly clustered as shown in Fig. 9. The gene XM_355567 was combined into the cluster 3 in the present analysis, although it was included in cluster 2 in the Homologene database. All the clusters except for the gene XM_355567 showed a good agreement with the Homologene database.

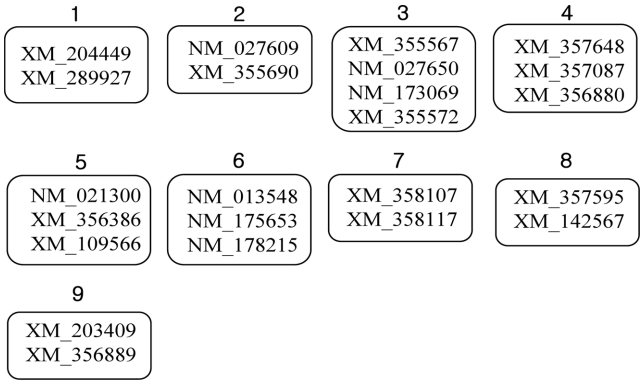

Fig. 9. The gene clusters for the minimum block size of 10 in Case Study 1.

Table 5. The ratio of the cross-matching subsequences in the clusters of Fig. 9

The number of clusters	The number of cross-matching subsequences	The number of total subsequences	The ratio of masking (%)
Cluster 1	3	22	13.64
Cluster 2	17	40	42.50
Cluster 3	19	140	13.57
Cluster 4	5	56	8.93
Cluster 5	4	29	13.79
Cluster 6	2	25	8.00
Cluster 7	49	55	89.09
Cluster 8	5	50	10.00
Cluster 9	12	29	41.38

In the cluster groups of Fig. 9, the cross-matching common subsequences were masked since they may cause inexact-matching. Table 5 shows the ratio of masking the cross-matching subsequences in the clusters of Fig. 9.

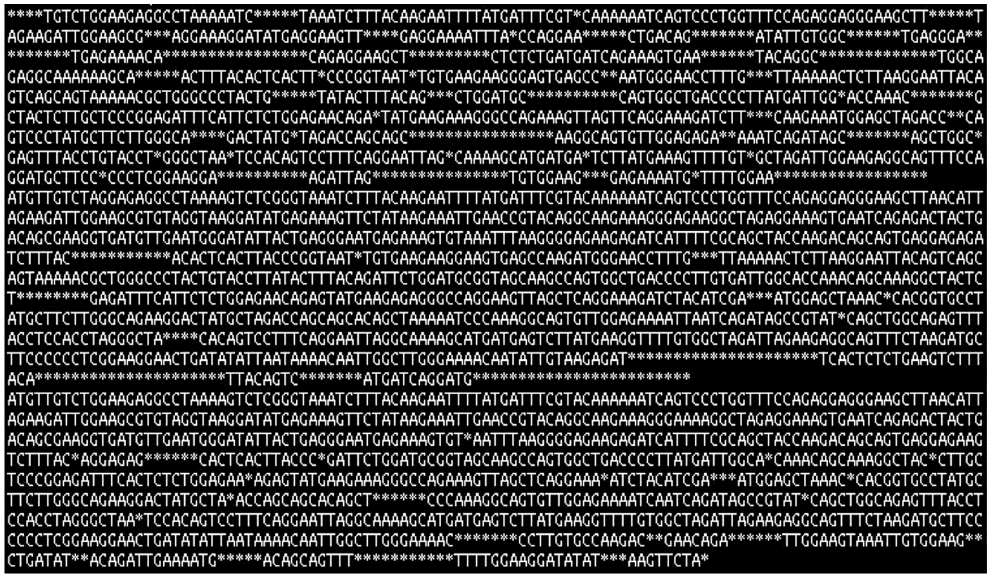

Fig. 10. The result of alignment for group 4 at the minimum block size of 7.

Fig. 11. The result of alignment for group 4 at the minimum block size of 10.

Fig. 12. The result of alignment for group 4 at the minimum block size of 15.

Also, to identify the relationship between common subsequences and minimum block size, the sequence alignments of common subsequences in cluster group 4 of Fig. 9 using minimum block size 7, 10, 15, 20 were presented in Fig. 10, 11, 12, 13 respectively. The characters except for symbol '*' represent the common subsequences. These figures show that the large number of minimum block size causes a decrease of the number of common subsequences. So a too large minimum block size results in a decrease of sequence similarity and false clusters.

2. Case Study 2: Twenty Two Gene Sequences of Different Species

The twenty two genes (Table 6) in the three species were retrieved from the Homologene database and clustered. The length of gene sequences was 700-3,700 bp, longer than Case Study 1. To reduce unnecessary common subsequences, the minimum block size was selected as 7-20. Also, the similarity threshold (a criterion for clustering sequence pairs) was set to 20%.

The clustering procedure of Case Study 2 is the same as Case Study 1. Fig. 14 shows the relationship between runtime and minimum block size. Since a shorter block size results in more common subsequences to be handled, the runtime increases. Due to a longer length and increased number of fragments, a longer runtime was observed in Case Study 2.

Fig. 15 shows that the gene sequences were properly clustered at the minimum block size of 10. At the minimum block size of 15 and 20, NM_001547 in the cluster 1 and NM_133492 in the cluster 3 were not clustered. Also, NM_173565 in the cluster 8 was not in the cluster in case of the minimum block size of 20. NM_003810 and NM_006926 that were not included at the minimum block size of 10 were clustered, respectively, into cluster 2 and cluster 7 at a minimum block size of 7. Most clusters in Fig. 15 accorded with the clusters in the Homologene database. It is noted that as the minimum block size decreases, the number of matching common subsequences per sequence pair increases. The cross-matching subsequences in the clusters of the Fig. 15 were eliminated and the ratio of masking the cross-matching subsequences was presented in Table 7.

Fig. 13. The result of alignment for group 4 at the minimum block size of 20.

Table 6. The 22 Genes of the three species

Species	Accession number
Homo sapiens	NM_001547, NM_003810, NM_133492, NM_052890, NM_000546, NM_173565, NM_006926
Mus musculus	NM_008332, NM_009425, NM_175731, NM_011640, NM_009921, NM_023134, XM_205565
Rattus norvegicus	XM_220060, NM_145681, XM_236790, XM_234838, NM_030989, XM_236642, NM_017329, XM_213713

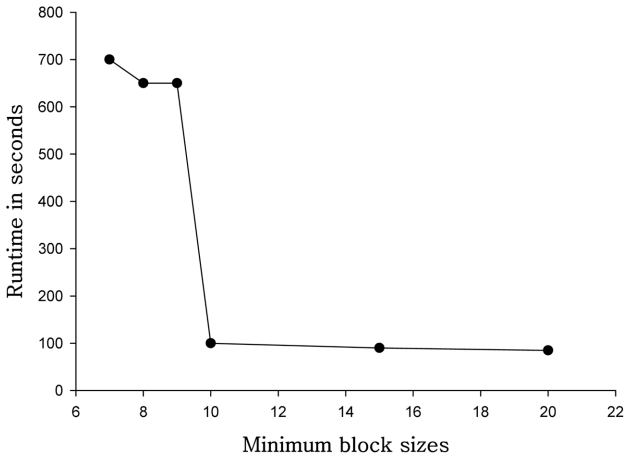

Fig. 14. The relationship between runtime and minimum block size in Case Study 2.

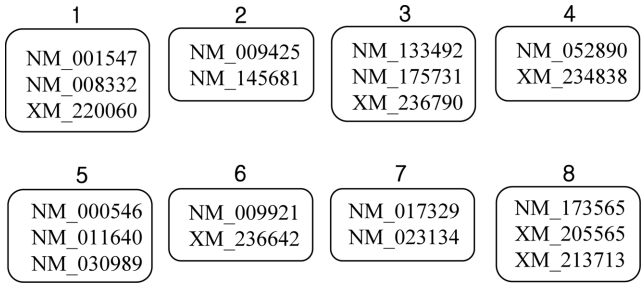

Fig. 15. The gene clusters for the minimum block size of 10 in Case Study 2.

Table 7. The ratio of the cross-matching subsequences in the clusters of Fig. 15

The number of clusters	The number of cross-matching subsequences	The number of total subsequences	The ratio of masking (%)
Cluster 1	28	123	22.76
Cluster 2	1	29	3.45
Cluster 3	7	56	12.5
Cluster 4	14	38	36.84
Cluster 5	21	141	14.89
Cluster 6	0	19	0
Cluster 7	6	46	13.04
Cluster 8	29	191	15.18

As shown in Fig. 8 and 14, a minimum block size of 10 was observed to be suitable for the algorithm in terms of the speed and precision. With a block size of 10 in the two Cases, the clusters obtained by the present method were in good agreement with those Homologene(http://www.ncbi.nlm.nih.gov/entrez/). As the clustered data in the Homologene database are gene sequences having similar functions and finally identified by experiments, the clusters with the other block size value may not be error. So the better value of the block size would be obtained from running the clustering tool with more gene data. It was also observed that the increases of the length of sequence and the number of common subsequences make the runtime longer.

Consequently, this method of performing multiple sequence alignment based on the sequence fragments was evaluated for two case studies: (i) the gene sequences in a single species; and (ii) the gene

sequences in three species from the Homologene database, and nine and eight clusters were presented.

CONCLUSIONS

The suffix tree algorithm has been used to cluster the search results of web documents [Zamir and Etzioni, 1998] and to find the conserved region in the related genomes [Delcher et al., 1999, 2002; Miller et al., 1999] or repeated region [Volfovsky et al., 2001].

The existing methods rely on pairwise comparison and progressive alignment to cluster sequences and are not effective in comparing DNA or proteins. In this paper, a modified STC (Suffix Tree Clustering) was proposed, which enables us to compare sequences at once without the pairwise alignment and Smith-Waterman algorithm.

The current method did not consider gap penalty or other refinements. In the future, these options could be included to improve specificity while keeping high sensitivity. Once it is done to implement performance-decisive parts of the algorithm in a lower level language with focus on optimization [Choi and Manousiouthakis, 2002], clustering larger gene sequences will be possible. Such clustered data can assist in studying sequences of unknown function or the evolutionary history of the sequences.

ACKNOWLEDGMENT

This work was supported by Brain Korea 21 Project in 2004.

REFERENCES

Chen, J. Y. and Carlis, J. V., "Genomic Data Modeling," *Information Systems*, **28**, 287 (2003).

Choi, S. H. and Manousiouthakis, V., "Global Optimization Methods for Chemical Process Design: Deterministic and Stochastic Approaches," *Korean J. Chem. Eng.*, **19**(2), 227 (2002).

Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O. and Salzberg, S. L., "Alignment of Whole Genomes," *Nucleic Acids Res.*, **27**(11), 2369 (1999).

Delcher, A. L., Phillippy, A., Carlton J. and Salzberg, S. L., "Fast Algorithms for Large-scale Genome Alignment and Comparison," *Nucleic Acids Res.*, **30**(11), 2478 (2002).

Gusfield, D., *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, Cambridge, London (1997).

Higgins, D. G., Thompson, J. D. and Gibson, T. J., "Using CLUSTAL for Multiple Sequence Alignments," *Methods Enzymol.*, **266**, 383 (1996).

Higgins, D. G. and Sharp, P. M., "CLUSTAL: A Package for Performing Multiple Sequence Alignment on a Microcomputer," *Gene*, **73**, 237 (1988).

Hon, W. K. and Sadakane, K., "Space-Economical Algorithms for Finding Maximal Unique Matches," *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching*, 144 (2002).

Kalyanaraman, A., Aluru, S. and Kothari, S., *Parallel EST Clustering*, HICOMB 2002, 185 (2002).

Kim, D. K., Lee, K. S. and Yang D. R., "Control of pH Neutralization Proess Using Simulation Based Dynamic Programming," *Korean J. Chem. Eng.*, **21**(5), 942 (2004).

Lee, J. M. and Lee, J. H., "Simulation-Based Learning of Cost-To-Go for Control of Nonlinear Processes," *Korean J. Chem. Eng.*, **21**(2), 338 (2004).

McCreight, E., "A Space Economical Suffix Tree Construction Algorithm," *Journal of the ACM*, **23**, 262 (1976).

Miller, R. T., Christoffels, A. G., Gopalakrishnan, C., Burke, J., Ptitsyn, A. A., Broveak, T. R. and Hide, W. A., "A Comprehensive Approach to Clustering of Expressed Human Gene Sequence: The Sequence Tag Alignment and Consensus Knowledge Base," *Genome Research*, **9**, 1143 (1999).

Morgenstern, B., Frech, K., Dress, A. and Werner, T., "DIALIGN: Finding Local Similarities by Multiple Sequence Alignment," *Bioinformatics*, **14**, 290 (1998).

Mount, D. W., *Bioinformatics : Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press (2001).

Needleman, S. B. and Wunsch, C. D., "A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins," *J. Mol. Biol.*, **48**, 443 (1970).

Notredame, C. and Higgins, D. G., "SAGA: Sequence Alignment by Genetic Algorithm," *Nucleic Acids Res.*, **24**, 1515 (1996).

Ostell, J. M., Wheelan, S. J. and Kans, J. A., "The NCBI Data Model," *Methods Biochem. Anal.*, **43**, 19 (2001).

Pearson, W. R. and Miller, W., "Dynamic Programming Algorithm for Biological Sequence Comparison," *Methods Enzymol.*, **210**, 575 (1992).

Phillips, A., Janies, D. and Wheeler, W., "Multiple Sequence Alignment in Phylogenetic Analysis," *Molecular Phylogenetics and Evolution*, **16**, 317 (2000).

Randal, L. S. and Christiansen, T., *Learning Perl*, Second Edition, O'Reilly (1997).

Salzberg, S. L., Searls, D. B. and Kasif, S., *Trends Guide to Bioinformatics*, Elsevier Science (1998).

Shin, P. K., Koo, J. H. and Lee, W. J., "Modeling of Cell Growth and phoA-Directed Expression of Cloned Genes in Recombinant *Escherichia coli*," *Korean J. Chem. Eng.*, **13**(1), 82 (1996).

Smith, T. F. and Waterman, M. S., "Identification of Common Molecular Sequences," *J. Mol. Biol.*, **197**, 723 (1981).

Thompson, J. D., Higgins, D. G. and Gibson, T. J., "CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-specific Gap Penalties and weight Matrix Choice," *Nucleic Acids Res.*, **22**, 4673 (1994).

Tisdall, J. D., *Beginning Perl for Bioinformatics*, O'REILLY (2001).

Ukkonen, E., "On-line Construction of Suffix Trees," *Algorithmica*, **14**, 249 (1995).

Volfovsky, N., Haas, B. J. and Salzberg, S. L., "A Clustering Method for Repeat Analysis in DNA Sequences," *Genome Biology*, **2**, 1 (2001).

Weiner, P., "Linear Pattern Matching Algorithms," *In Proc. of the 14th IEEE Annual Symposium on Switching and Automata Theory*, 1 (1973).

Zamir, O., Etzioni, O., Madani, O. and Karp, R. M., "Fast and Intuitive Clustering of Web Documents," *In Proc. of the 3rd International Conference on Knowledge Discovery and Data Mining*, 287 (1997).

Zamir, O. and Etzioni, O., "Web Document Clustering: A Feasibility Demonstration," *In Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 46 (1998).