Kim WC, Park S, Won JI. CORE: Common region extension based multiple protein structure alignment for producing multiple solution. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 28(4): 647–656 July 2013. DOI 10.1007/s11390-013-1365-x

CORE: Common Region Extension Based Multiple Protein Structure Alignment for Producing Multiple Solution

Woo-Cheol Kim[1], Sanghyun Park[2], and Jung-Im Won[3,*]

[1] *College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802, U.S.A.*

[2] *Department of Computer Science, Yonsei University, Seoul 120-749, Korea*

[3] *Research Center of Information and Electronic Engineering, Hallym University, Chuncheon, Gangwon 200-702, Korea*

E-mail: wxk11@psu.edu; sanghyun@cs.yonsei.ac.kr; jiwon@hallym.ac.kr

Received September 8, 2012; revised May 10, 2013.

Abstract Over the past several decades, biologists have conducted numerous studies examining both general and specific functions of proteins. Generally, if similarities in either the structure or sequence of amino acids exist for two proteins, then a common biological function is expected. Protein function is determined primarily based on the structure rather than the sequence of amino acids. The algorithm for protein structure alignment is an essential tool for the research. The quality of the algorithm depends on the quality of the similarity measure that is used, and the similarity measure is an objective function used to determine the best alignment. However, none of existing similarity measures became golden standard because of their individual strength and weakness. They require excessive filtering to find a single alignment. In this paper, we introduce a new strategy that finds not a single alignment, but multiple alignments with different lengths. This method has obvious benefits of high quality alignment. However, this novel method leads to a new problem that the running time for this method is considerably longer than that for methods that find only a single alignment. To address this problem, we propose algorithms that can locate a common region (CORE) of multiple alignment candidates, and can then extend the CORE into multiple alignments. Because the CORE can be defined from a final alignment, we introduce CORE* that is similar to CORE and propose an algorithm to identify the CORE*. By adopting CORE* and dynamic programming, our proposed method produces multiple alignments of various lengths with higher accuracy than previous methods. In the experiments, the alignments identified by our algorithm are longer than those obtained by TM-align by 17% and 15.48%, on average, when the comparison is conducted at the level of super-family and fold, respectively.

Keywords structure alignment, similarity search, protein structure

1 Introduction

Over the past several decades, biologists have conducted numerous studies examining both general and specific functions of proteins[1]. However, due to the time and effort required for the experimental methods employed by biologists, there are numerous limitations to these approaches in analyzing protein functions. As such, automated computer systems have recently been developed to analyze the functions of multiple proteins simultaneously[2].

Generally, if similarities in either the structure or sequence of amino acids exist for two proteins, then a common biological function is expected[3]. Protein function is determined primarily based on the structure rather than the sequence of the amino acids that compose it[4]. The fact that the sequence of amino acids can be mutated into other forms during evolutionary processes while the structure generally remains intact is evidence of this[5]. In this context[6], it has been shown that replicate proteins can be artificially constructed from existing proteins when the two proteins have similar functions and structures even with different amino acids sequences. That is, the structure of functionally related proteins provides additional insight into their functional mechanisms and this knowledge has been successfully applied to the functional annotation of proteins whose structure has been previously identified[7-8].

Regular Paper

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology of Korea under Grant No.2012R1A1A3013084.

The preliminary version of the paper was published in the Proceedings of EDB2012.

*Corresponding Author

The structure alignment problem (SAP) is the term applied to the need to find the *superposition* and the *transformation* that allow the backbones of two proteins to be aligned resulting in the highest similarity score. The superposition is a mapping that is completed between the residues of two proteins, and the transformation is the combination of the rotation and translation of the proteins. The structure superposition problem (SSP) requires that the superposition of a protein is already known, and it is easier to solve than the SAP, as the former only requires the determination of the transformation leading to an optimal alignment[9]. The SSP can be solved in linear time as a function of the number of residues in the protein[10] through a process of combining the rotation and translation of the proteins to minimize their root mean square deviation (RMSD). However, the SAP is a non-deterministic polynomial-time hard (NP-hard) problem[11], and thus, most methods used to address this problem employ a heuristic approach[12-13].

A general framework for the development of an algorithm for the SAP consists of the following four steps[9]: 1) represent the structure of the two proteins to be aligned in spatial coordinates; 2) align the two protein structures; 3) optimize the alignment; and 4) evaluate the statistical significance of the alignment using Z-scores or other methods. The quality of the algorithm depends on a quality of the similarity measure that is used, and the similarity measure is an objective function used to determine the best alignment of the two proteins[14]. Most SAP algorithms employ the cRMSD (coordinated root mean square deviation) as the similarity measure. However, when using the cRMSD, it is difficult to intuitively judge the level of similarity of the two protein structures[15]. Better alignment is expected when the alignment length is long and the cRMSD is small. However, the alignment length and the cRMSD are positively correlated, such that when the alignment length is longer, the cRMSD is larger. As such, many studies that examine the SAP attempt to balance the two parameters through the use of many heuristics or statistical methods. Although several usable SAP algorithms have been developed, none has resulted in a golden or optimal rate. In particular, it is difficult to determine which would be the more accurate alignment in the following example: (100, 3.2 Å) or (104, 3.3 Å), which means (alignment length, cRMSD).

In this paper, a new strategy is employed that finds not a single alignment, but multiple alignments with different lengths. This method has obvious benefits in that it does not require that the two parameters be balanced. However, this novel method leads to a new problem. Specifically, the running time for this method is considerably longer than that for methods that find only a single alignment. To address this problem, we propose algorithms that can locate a common region of multiple alignments, and can then extend the common region into multiple alignments.

The SAP has been established as an NP-hard task. Thus, to reduce the complexity of the SAP, the majority of researchers have simplified the SAP by adjusting it to take on the form of the SSP. In this approach, the substructures are first aligned, resulting in significantly less computational cost, and then, the aligned substructures are used as a superposition to aid in the alignment of the remainder of the structure.

2 Related Work

Protein alignment algorithms can be classified into three categories according to the type of data that is used in the algorithms[16]. Specifically, the algorithms in the first, second and third categories use the C_α atom, Secondary Structure Element (SSE), and geometric hashing, respectively.

For protein alignment algorithms in the first category, the simplest method involves using the C_α atom in dynamic programming. The best-known of these methods are DALI[5] and CE[17]. Double Dynamic Programming[18], Iterative Dynamic Programming[19] and MINRMS[20] are based on multiple forms of dynamic programming.

The DALI aligns protein structures through the use of distance matrices. A distance matrix is an $n \times n$ matrix, representing the distance between each pair of residues of the underlying protein structure. Specifically, the cell at the i-th row and the j-th column of the distance matrix denotes the distance between the i-th residue and the j-th residue in the protein structure. Proteins with similar structures produce similar distance matrices. To align the structures of two proteins, DALI compares their distance matrices. To reduce the processing time for this type of alignment, DALI divides the distance matrix of each protein structure into segments and identifies segment pairs with similar distance patterns. Then, the overlapping segment pairs identified in the previous step are combined into larger segment pairs to maximize the similarity score.

Rather than using distance matrices, the CE approach divides protein structures into fragments and finds pairs of these fragments, called aligned fragment pairs (AFPs), which display high levels of similarity. Next, similar to the segment pairs found using the DALI, the AFPs are extended into a final alignment.

The computational cost of methods that are based on dynamic programming, such as DALI and CE, de-

pends upon the length of the protein structures to be aligned. Both DALI and CE have adopted a heuristic approach to reduce computational cost by treating a certain number of C_α atoms as a single processing unit. However, these methods do so without considering the similarity of the protein structures to be aligned. Therefore, even if the two protein structures being aligned have a high level of similarity, each protein structure has to be broken into multiple fragments that are subsequently compared with the fragments of the other protein. One limitation of these methods is that they require a significant amount of time even when the protein structures are analogous.

Lotan and Schwarzer proposed an algorithm[21] to minimize the alignment time by reducing the number of C_α atoms prior to the alignment process. This algorithm uses a small number of C_α atoms during the alignment process, and thus, reduces the alignment time. This method can determine the similarity of aligned protein structures. However, this method is unable to determine which of the residues are responsible for the alignment.

One method that is representative of the second category of protein alignment algorithms, called VAST, uses SSE (Secondary Structure Element)[22]. VAST is a hierarchical alignment algorithm. The algorithm initially aligns the two proteins using only their SSE, which is the higher structure data than the C_α atoms. Next, the aligned result is used as a superposition that is extended into the C_α atom level. One of the strengths of this type of algorithm is that it allows for the superposition to be more quickly identified through the use of SSEs rather than C_α atoms, thus reducing the total alignment time. However, in this process, there are still proteins whose C_α atoms are identified although the SSEs are not, and this limits the usage of this algorithm. Information about the C_α atoms can be obtained through nuclear magnetic resonance (NMR) spectroscopy and/or X-ray crystallography. Typical programs used to locate SSEs include DSSP[23] and STRIDE[24].

The third category of algorithms uses geometric hashing, which converts reference frames into hash values and stores them as a hash table. Protein structures are then aligned using these hash values as frames of reference. 3-D (3-dimensional) lookup[25] is another representative algorithm used in the alignment of protein structures, and this method employs the SSE as a reference frame. Additionally, Nussinov and Wolfson[26] have proposed a protein structure alignment algorithm that employs the C_α atom as a reference frame[27]. However, the accuracy of these methods is low as only the predefined reference frames are used during the alignment process.

Recently, Erdmann[28] proposed a protein structure alignment algorithm. To align the protein structure, this algorithm finds the helix and intersections of each protein structure by employing knot theory and geometric convolution. Although two protein structures may not be similar as a whole, this algorithm determines that they are analogous if the proteins have a similar helix or intersection.

TM-align[29] is an up-to-date algorithm that combines the use of the TM-score[30] rotation matrix and dynamic programming (DP). The major difference between the TM-align and other DP-based algorithms is that the TM-score rotation matrix, rather than the Kabsch RMSD rotation matrix, is exploited in both the heuristic DP iterations and final alignment selection. However, when using the TM-align, it is difficult to identify an alignment that has the longest evolutionary distance among multiple results which can be found by the TM-align because the TM-score weights the close matches stronger than distant matches.

3 Problem Definition

In this section, the SAP is defined as a numerical formula. The protein structure, S, is given with the complete set of x, y, z coordinates for all atoms. This representation can be further reduced to the α-carbon backbone atoms, $S = \langle r_i \rangle_{i=1}^{n}$, where n is the number of amino acids. The amino acids are ordered according to the preexisting order on the protein chain. The substructure of protein, $u = \langle q_i \rangle_{i=1}^{k}$, $u \subset S$, is the structure that is a subset of atoms in S. u_k and P_k denote the substructure and the set of substructures with the same length respectively, where k is the length of the substructure. Fig.1(a) and Fig.1 (b) show an example of S and u respectively.

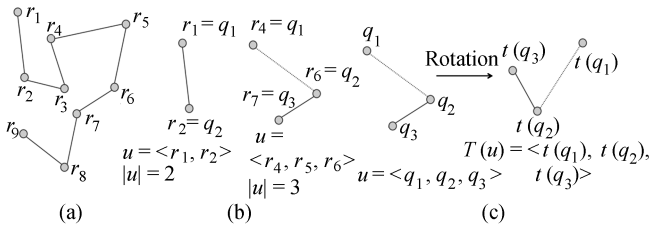

Fig.1. (a) Protein structure, S. (b) Sub-structure, u. (c) Transformation, T.

In this paper, structure alignment refers not to the global alignment but to the local alignment. Therefore, a similarity measure for substructures is needed. It is denoted by $M(u^a, u^b|F)$, where F is a similarity function that expresses the similarity between u^a and u^b. Here a larger value of F implies that the pair is more similar. Although cRMSD and dRMSD (distance root mean square deviation) are distance functions, they are

adequate representations of F because they have been converted to a similarity function through a conversion formula, $sim = 1/(1 + dist)$, where sim denotes similarity and $dist$ denotes distance. Specifically, our proposed method uses cRMSD as the F, and M is represented as follows:

$$\begin{aligned} M(u^a, u^b|F) =& 1/(1 + cRMSD(u^a, u^b)) \\ =& 1\Big/\Big(1 + \sqrt{\frac{\sum_{i=1}^{k} d_i}{k}}\Big), \\ k = |u^a| = |u^b|, d_i =& |q_i^a - q_i^b|. \end{aligned}$$

Based on this definition, the SAP is defined as follows:

$$\begin{aligned} SAP(S^a, S^b) &= (u^a, u^b) \\ &= \arg\max_{u^a, u^b} M(u^a, u^b|F), \\ u^a \subset S^a, u^b \subset S^b, |u^a| &= |u^b|. \end{aligned}$$

The formulas outlined above are explained through the following steps. First, all substructure pairs (u^a, u^b) on S^a and S^b are populated. Second, M for every pair is computed, and the pair (u^{*a}, u^{*b}) for which M is the highest is chosen as the final result.

However, this formula needs to be modified since it is possible for S to be rotated (*rotation*) or translated (*translation*) in 3-D space, and these actions are called *transformation*. Therefore, even when the same pair is aligned, its M may have a very different value as a result of transformation. For this reason, the SAP is redefined with the following formula including transformation, T, shown in Fig.1(c).

$$\begin{aligned} SAP(S^a, S^b) &= (u^{*a}, u^{*b}, T^*) \\ &= \arg\max_{u^a, u^b, T} M(T(u^a), u^b|F), \\ u^a \subset S^a, u^b \subset S^b, |u^a| &= |u^b|, \end{aligned}$$

where (u^{*a}, u^{*b}) and T^* indicate a superposition and a transformation, respectively.

4 Similarity Measure

The cRMSD has a critical weakness as a similarity function as it does not reflect the alignment length. For example, when there are two alignments, $|u^{a'}| = |u^{b'}| = 21$ with cRMSD $= 0.2$ Å and $|u^{a''}| = |u^{b''}| = 36$, with cRMSD $= 0.3$ Å, the former is the better alignment according to cRMSD. However, in this example, the latter is clearly the better alignment if the alignment length is considered in addition to cRMSD.

Due to this problem, the SAP algorithms using cRMSD have proposed similarity measures that combine cRMSD and alignment length. To find an optimum or golden rule for the combination of cRMSD and alignment length, many researchers have applied heuristics that have been identified based on the experience of these researchers and therefore reflect the subjective views of the particular researcher. Thus, a particular user is unlikely to identify the best alignment by using another researcher's algorithm if their perspectives differ.

In order to address the problem of the cRMSD-based SAP, a straightforward solution is proposed that finds not a single alignment but multiple alignments, each of which is the best alignment for each individual alignment length. Fig.2 shows an example of difference between a single alignment and multiple alignments.

$$\begin{aligned} SAP(S^a, S^b) &= \bigcup_{k=1}^{N} SAP(P_k^a, P_k^b), \\ N &= \min(|S^a|, |S^b|) \\ &= (u_1^{a*}, u_1^{b*}, T_1^*) \cup \cdots \cup (u_i^{a*}, u_i^{b*}, T_i^*) \\ &\quad \cup \cdots \cup (u_N^{a*}, u_N^{b*}, T_N^*). \end{aligned}$$

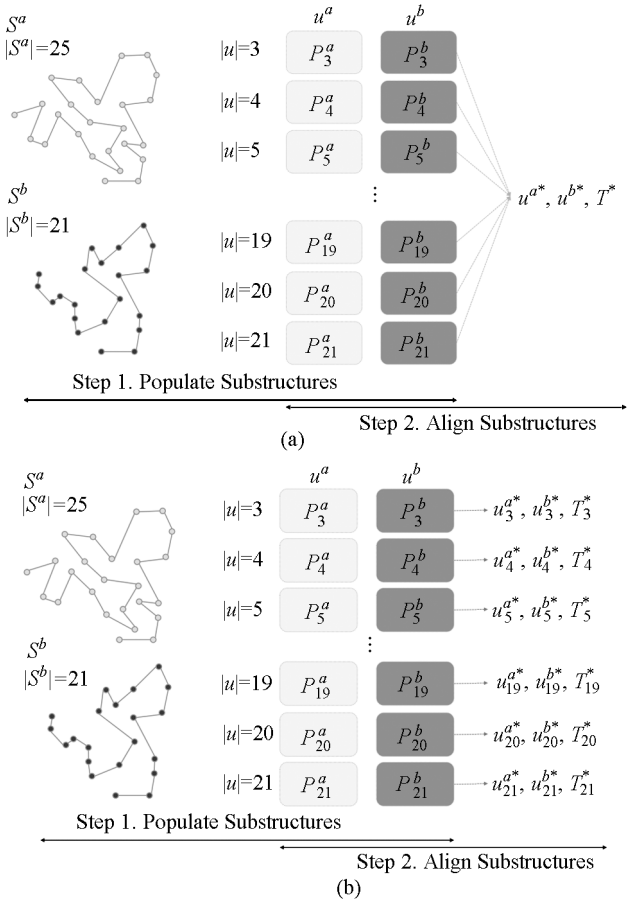

Fig.2. Single alignment versus multiple alignments with different lengths. (a) Single alignment. (b) Multiple alignments with various lengths.

The method for finding multiple alignments is a simple and objective way to address the aforementioned problem. However, the amount of time required for this technique will be N- times longer than that of the methods that find a single alignment if a naïve algorithm is used to find the multiple alignments (where N is the number of alignments). Therefore, in order to reduce the time-complexity, the CORE-based alignment is introduced in the next section.

5 CORE Algorithm

In this section, the CORE is introduced and the method for finding it and using it to find multiple alignments with various lengths is explained.

To reduce the processing time required to find multiple alignments, the ideas proposed in [31] are utilized. Heuristic-based algorithms typically find a local optimum not a global optimum. Shown in Fig.3, a common region with long length residues exists between the local optima[31]. Thus, if an algorithm is developed to locate this common region, the time-complexity for finding multiple alignments may be reduced. Therefore, the common region is defined as the CORE, and an algorithm to find the CORE and to extend the CORE into multiple alignments is proposed herein.

Finding the CORE (step 2 in Fig.4) is the most important step in the proposed algorithm. According to the analysis of alignments produced by various SAP algorithms, two characteristics of the final alignment were identified. One of these characteristics is that the number of fragment-pairs in the alignment is small. The fragment is defined as f that is a substructure and that

$$u^a = \langle\ r^a_1,\ r^a_3, r^a_4, r^a_5,\ r^a_6,\ \ r^a_9\rangle$$
$$u^b = \langle\ r^b_3,\ r^b_4, r^b_5, r^b_6,\ \ r^b_7,\ \ r^b_8\rangle$$
(a)

$$u^a = \langle\ r^a_2,\ r^a_3, r^a_4, r^a_5,\ r^a_6,\ \ r^a_9, r^a_{10}\rangle$$
$$u^b = \langle\ r^b_2,\ r^b_4, r^b_5, r^b_6,\ \ r^b_7,\ \ r^b_8, r^b_9\rangle$$
(b)

$$u^a = \langle\ r^a_2,\ r^a_3, r^a_4, r^a_5,\ r^a_6,\ \ r^a_9\rangle$$
$$u^b = \langle\ r^b_3,\ r^b_4, r^b_5, r^b_6,\ \ r^b_7,\ \ r^b_8\rangle$$
(c)

$$u^a = \langle\ r^a_3,\ r^a_4, r^a_5, r^a_6,\ r^a_{11},\ r^a_{12}\rangle$$
$$u^b = \langle\ r^b_4,\ r^b_5, r^b_6, r^b_7,\ r^b_{11},\ r^b_{12}\rangle$$
(d)

$$u^a = \langle\ r^a_3,\ r^a_4, r^a_5, r^a_6\ \rangle$$
$$u^b = \langle\ r^b_4,\ r^b_5, r^b_6, r^b_7\ \rangle$$
(e)

Fig.3. Various local optima produced by various SAP algorithms. Each alignment, (a), (b), (c) and (d), has characteristics that reflect the heuristics used in SAP algorithms. (e) is the common region of (a), (b), (c) and (d).

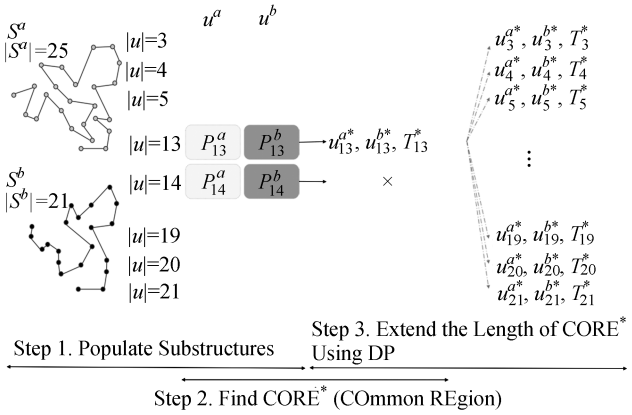

Fig.4. Proposed algorithm.

satisfies the order constraint: $q_m = r_o$, $q_{m+1} = r_{o+1}$. The fragment-pair is defined as a pair of f. The number of substructures and the number of substructure pairs can be sometime different, shown in Fig.5. For example, the number of fragment-pairs in the alignment for which the length is around 100 is approximately 12.

$$S^a = \langle r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\rangle$$
$$S^b = \langle r_1, r_2, r_3, r_4, r_5, r_6, r_7\rangle$$
(a)

$$u^a = \langle\ r_2, r_3, r_4,\ r_7, r_8\ \rangle$$
$$u^b = \langle\ r_1, r_2,\ r_5, r_6, r_7\ \rangle$$
(b)

$$u^a = \langle\ r_2, r_3,\ r_4,\ r_7, r_8\ \rangle$$
$$u^b = \langle\ r_1, r_2,\ r_5,\ r_6, r_7\ \rangle$$
(c)

Fig.5. Example of the fragmentation of a final alignment. (a) Protein structures, S^a and S^b, consist of 8 and 7 amino acids (or residues), respectively. (b) Structure alignment for S^a and S^b. Although both u^a and u^b have two fragments, their fragmented positions are not the same. As a result, like (c), the number of fragment-pairs in the alignment is 3.

The other characteristic is that the lengths of the fragment-pairs in the alignment do not follow a normal distribution. The lengths of most fragment-pairs are much smaller than the average length of the fragment-pairs, and the longest fragment-pair forms a large portion of the alignment. For example, when the average fragment-pair length is 8 (Fig.6 shows the distribution of fragment-pair length), it is common that the fragment-pair length pattern in the final alignment is 14-26-2-2-6 rather than 10-8-6-7-9. When comparing the alignments of highly homologous proteins (e.g., the *family* level of the hierarchical SCOP structural classification database[32]) the average length of an alignment and the average lengths of the longest fragment-pair are 127 and 47, respectively.

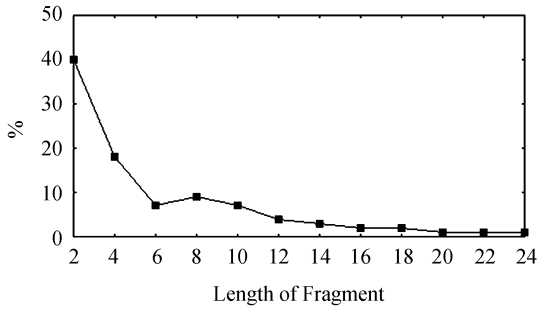

Fig.6. Graph showing the distribution of the fragment-pair lengths: the length of most fragment-pairs is smaller than the average (= 8).

Definition 1 (CORE). *Given an alignment, (u^{a*}, u^{b*}), for two protein structures, S^a and S^b, let (u'^{a*}, u'^{b*}) be the aligned fragment-pair that satisfies the following conditions: $u'^{a*} \subset u^{a*}$, $u'^{b*} \subset u^{b*}$, and u'^{a*} and u'^{b*} are fragments. CORE denotes the longest aligned fragment-pair.*

According to Definition 1, the CORE can be found only from a final alignment. However, the CORE must be used as seed to find an alignment. As a solution to this problem, we introduce CORE* that is similar to CORE and propose an algorithm to identify the CORE*.

Definition 2 (CORE*). *Given two fragments, f^a and f^b, from structures S^a and S^b, respectively, let f^a and f^b be the aligned fragment-pair candidates that satisfy the condition that the similarity score of their alignment is larger than the user-defined threshold, T. CORE* denotes the longest aligned fragment-pair candidate.*

As discussed above, the most significant characteristic of CORE is that its length is significantly longer than the average length of the fragment-pairs. That is, if the longest fragment-pair is identified from those with a similarity level greater than that of the average pair, it is probably the same as that of CORE. This observation is more accurate when the length of the CORE is longer. In a preliminary experiment with a real dataset, 97% of CORE*s, of which length is longer than 32, are identical to CORE. The remaining 3% of the CORE* findings only have two mismatches.

A naïve procedure to find CORE* is composed of the following four steps: 1) generate all fragment-pairs from S^a and S^b; 2) compute M for all fragment-pairs; 3) remove fragment-pairs for which M is smaller than the user threshold; and 4) choose the longest fragment-pair among the survivors of the previous steps as the CORE*. However, this procedure is time-consuming as the number of generated fragment-pairs is quite large. For example, the number of fragment-pairs from S^a ($|S^a| = 120$) and $S^b(|S^b| = 100)$ is 439 350. Therefore, the following two major heuristics are adopted to increase the speed of the verification process for the fragment-pairs.

One heuristic is that the verification process proceeds from the longer fragment-pair to the shorter fragment-pair as it is desirable to identify the longest survivor. In particular, the practical maximum length of CORE*, not the theoretical maximum length, is used as the beginning length for this process. As an example, when aligning two structures, S^a and S^b, the lengths of which are 120 and 100, respectively, the practical maximum length of CORE is no longer than 43. Nonetheless, the theoretical maximum length of CORE* is 100. This implies that it is needless to compute the M of fragment pairs over a length of 43.

The other heuristic involves changing the similarity function from cRMSD to dRMSD and L_∞, in order to increase the speed of the computation of M. CORE* is found not by the exact value of M, but by knowing whether the value of M is larger than the user-defined threshold. According to the analysis, it was determined that there is a high positive correlation between cRMSD, dRMSD, and L_∞ when the length of CORE* is long enough. The computation time for cRMSD is considerably longer than that for dRMSD and L_∞ because cRMSD requires *transformation*. This appears to be true when L_∞ is used as F, and it is faster to check whether L_∞ of fragment-pair is greater than the user-defined threshold. Theorem 1 proves this.

$$dRMSD(u^a, u^b) = \sqrt{\frac{\sum_{i=1}^{|A|}\sum_{j=1}^{|A|}(d_{ij}^a - d_{ij}^b)^2}{|A|}},$$

$$d_{ij} = |r_i - r_j|,$$

$$L_\infty(u^a, u^b) = \max_d \sum_{i=1}^{k} d_i,$$

$$k = |u^a| = |u^b|, d_i = |q_i^a - q_i^b|.$$

Theorem 1. *If a fragment-pair u^a and u^b, have two residues with the intra-residue distance of l^a and l^b, respectively, and if $|l^a - l^b|$ is greater than $2d$, none of T satisfies $L_\infty(T(u^a), u^b) < d$, $L_\infty(T(S^a), S^b) < d$.*

Proof. Let $l^b > l^a$, $L_\infty(T(u^a), u^b) < d$ means that the two points (=residues) of $T(u^a)$ have to be inside circle O_1 and O_2, respectively. To find two points meeting the condition, the distance of the two points, l^a, is longer than $l^b - 2d$, the nearest distance between O_1 and O_2. However, if $|l^a - l^b| > 2d$, none of T satisfies $L_\infty(T(u^a), u^b) < d$ because l^a is always less than $l^b - 2d$. This is proved by the inequality:

$$|l^a - l^b| > 2d \Leftrightarrow l^b - l^a > 2d \Leftrightarrow l^b - 2d > l^a. \quad \square$$

Lemma 1. *Given a fragment-pair, u^a and u^b, l_i^a*

denotes the distance between r_i^a *and* r_{i+1}^a *and* l_i^b *denotes the distance between* r_i^b *and* r_{i+1}^b. *If any* i *satisfies* $|l_i^a - l_i^b| > 2d$, *none of transformation,* T, *satisfies* $L_\infty(T(u^a), u^b) < d$.

Proof. $L_\infty(T(u^a), u^b) < d$ is representative of all of the distances between the residue pairs that are never greater than d. However, according to Theorem 1, if any i satisfies $|l_i^a - l_i^b| > 2d$, at least one of the L_∞ of a pair is greater than d. As a result, Lemma 1 is proved. Fig.7 shows a graphical representation of this proof. □

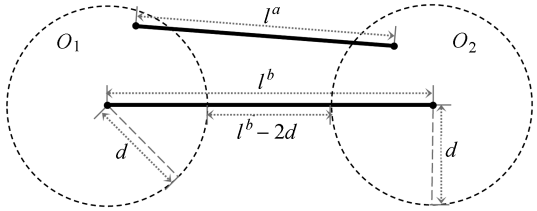

Fig.7. Distance between residues after transformation: l^a is a distance between the two contiguous residues of u^a and l^b is a distance between two contiguous residues of u^b.

6 Extended CORE* Algorithm

The above-obtained CORE* is a seed for the final alignment. However, the CORE* is too short when compared to the final alignment. Our analysis determined that its coverage is not higher than 40% of the final alignment. The DP is known to have the advantage of accuracy but also have the disadvantage of time-complexity. The length of the CORE* is too short to extend using DP. To reduce the running time of the extension algorithm, a longer seed is needed. Therefore, multiple CORE*s must be used. We call the multiple CORE* as E-CORE*.

The first issue to be addressed in composing an E-CORE* is which CORE* will be chosen. E-CORE* is an extension of CORE*. In other words, the T of the selected CORE*s must be similar to one another. If the two Ts are different, the similarity score of the merged CORE*s, E-CORE*, will likely be decreased significantly. A second criteria for selecting CORE*s is that the longer CORE* is preferred because the shorter CORE* is likely included in the other longer CORE*. Using these methods, we are able to set priority for selecting CORE*.

The second question to be addressed is how many CORE*s should be chosen. There is a trade-off between the running time and the quality of the final alignment according to the number of CORE*s in the E-CORE*. Therefore, recall and precision of E-CORE* must be examined to determine this parameter. The results are presented in Table 1.

Table 1. Recall(=Coverage) and Precision with Varied Number of CORE*

|E-CORE*|	Recall	Precision
1	0.32	0.98
2	0.46	0.94
3	0.53	0.79
4	0.57	0.68
5	0.61	0.64
6	0.65	0.58
7	0.68	0.51
8	0.72	0.43

Table 1 shows that precision decreases considerably when the number of CORE*s in the E-CORE* is greater than 3. This result indicates that the third CORE* included in the E-CORE* contains more mismatches than the first or the second. Although it is desirable to extend the E-CORE* as long as possible in this step, it is not desirable for the E-CORE* to have mismatches, as these are challenging to remove during the next post-processing step.

The precisions of E-CORE* shown in Table 1 is not 1, which means that E-CORE* contains mismatches. If the mismatches are not eliminated, the precision of the final alignments will never reach 1. In order to eliminate the mismatches, a simple heuristic that removes residue pairs for which the distance is beyond some threshold is applied to E-CORE*. The threshold is set at 6 Å based on the analysis shown in Fig.8.

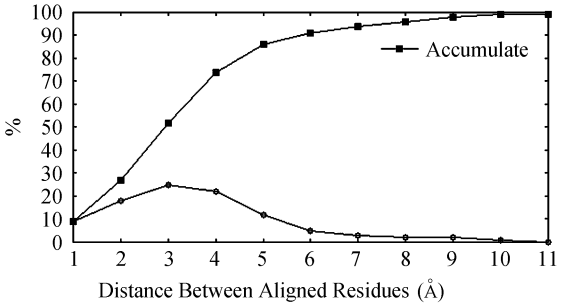

Fig.8. Distribution of the distance between the aligned residue pairs in the final alignment. The majority of the distances between the aligned residues are less than 6 Å.

The final step for this algorithm is to extend the E-CORE* into multiple alignments using DP. The element of the score matrix is assigned to M and 1 or 0, depending on whether or not the residues are aligned. When computing M, obvious differences exist between previous methods and the proposed method. The proposed method utilizes only residues of S^a and S^b for which the E-CORE* are removed, whereas previous methods make use of all the residues of S^a and

S^b. For example, given lengths of 56 and 48 for S^a and S^b, respectively, DP is performed after deploying $20(=56-36)$ residues in the x-axis and $12(=48-36)$ residues in the y-axis when the length of E-CORE* is 36. For the sake of simplicity, residues composing the E-CORE* are not utilized when the DP is formed, they are utilized when the M is computed. For example, if three matches are found in the DP process, it uses the M obtained by aligning 39 total residue pairs, including residue pairs of the three matches and 36 residue pairs of the E-CORE*.

7 Experiments

Experiments are conducted using a variety of protein structure data in Protein Data Bank (PDB)[33] to demonstrate the superiority of the proposed algorithm. First, the way that the number of CORE*s affects performance is examined by comparing the alignments according to the number of CORE*s. Second, performance when the refinement process is conducted compared to when it was not is examined to verify the necessity of the refinement process. Third, the superiority of the proposed method is verified through comparison with the TM-align algorithm. Lastly, further analysis is provided that examines the differences between the proposed method and TM-align.

Correct alignments are necessary to calculate the accuracy of the experiments. In the experiments, the results of the TM-align are used as the alignments. Incorrect alignments may be included in the TM-align results since this process is also based on heuristics. However, it is rare for family-level protein structure pairs to have a high level of similarity. Therefore, family-level and super-family-level protein structure pairs are utilized in calculating the accuracy. The ratio of the length of the common region between TM-align alignment and an alignment whose length is the same as TM-alignment among our multiple alignments is considered to be the accuracy; for example, if the length of TM-align alignment is 140 and the length of the common region is 133, then the accuracy will be 95% (133/140).

Two parameters are used in the experiments. The first is that the length of the fragment pair which is used as the start length of verification progress when finding CORE*. It was revealed that more than 99% of the alignments have COREs with a length of less than 38. Therefore, the alignment is conducted on the fragment pairs that have a length of 38. The second parameter used is the similarity score. The thresholds for dRMSD and L_∞ are used to decide whether or not the fragment-pairs could be the CORE*. After analyzing the CORE of alignments, the thresholds are decided to 9.41 Å and 4.05 Å respectively.

7.1 Processing Time and Accuracy According to the Number of Appended COREs*

In the process of finding seeds, the more CORE*s are used, the longer the average length of the seeds is. Consequently, this affects the accuracy of the alignment and processing time. As the accuracy decreases, the processing time is reduced according to the increase in the number of COREs used, as depicted in Fig.9, where the number following "L" means the number of CORE* which is used to build E-CORE*.

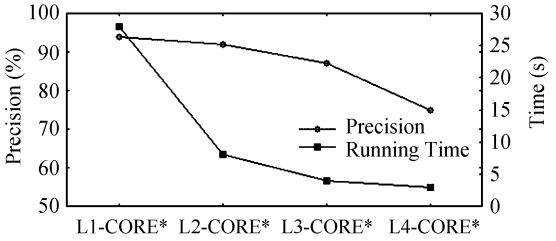

Fig.9. Processing time and accuracy according to the number of appended CORE*s used for seeds.

7.2 Accuracy After Refining the COREs

After locating seeds, the refinement process is executed. This eliminates the unnecessary mismatches of the seeds. This is particularly true as shown in the example in Fig.10.

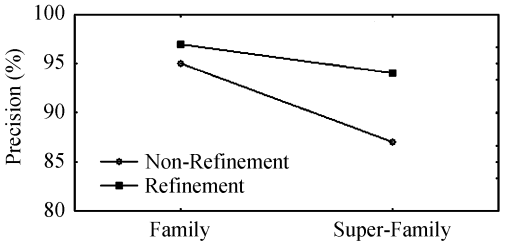

Fig.10. Accuracy with and without the refinement process of the family and super-family.

7.3 Comparison of Similarity with Other Algorithms

In order to compare the two methods, the CORE's alignments are measured. It is determined that CORE's alignments have the same length and similarity score as the TM-align's alignment. In the comparison, the CORE itself has different length from the TM-align, but has the same similarity scores. The CORE's alignments shown in Table 2 are longer than the TM-align's alignment by 0.08%, 5.17%, and 15.48%, on average, when the comparison is performed at the level of family, super-family, and fold, respectively.

Table 2. Comparison Between the Alignments Obtained by TM-Align and CORE

Class	Measure	Length	Similarity
Family	TM-align	120.80	2.35
	CORE	120.90	2.33
	Ratio (%)	0.08	0.85
Super-family	TM-align	77.40	3.21
	CORE	81.40	3.10
	Ratio (%)	5.17	3.43
Fold	TM-align	49.10	3.83
	CORE	56.70	3.51
	Ratio (%)	15.48	8.36

Additionally, the CORE's alignments have a greater similarity score than the TM-align's alignment by 0.85%, 3.43%, and 8.36% when the comparison is conducted on the level of family, super-family, and fold, respectively.

The performance gap is not significant in the case of family-level proteins, which means that there is a high level of similarity between the alignment targets. In the case of the fold-level, however, the CORE outperforms the TM-align by obtaining 12% gain in both length and similarity.

7.4 Graph-Based Representation for Multiple Alignments

Fig.11 represents alignments of the protein structure according to the length of the alignment. Since the TM-align produces only one alignment, its result is represented as a single dot. Contrarily, the CORE produces multiple alignments of various lengths; thus, the CORE alignments are represented as a line. Locating the saddle point, where the gradient radically increases, is a good way to determine the optimum alignment. Since a radically increased gradient indicates that the cRMSD value increased rapidly when a residue pair is appended to the alignment, an improved result is expected in the case of excluding that pair of residues.

Thus, it can be stated that the alignment of the CORE with a length of 123 are better than that of the TM-align with a length of 122, as shown in Fig.11(a). It can be concluded that the alignment of the CORE with a length of 90 is better than that of the TM-align with a length of 91, as shown in Fig.11(b).

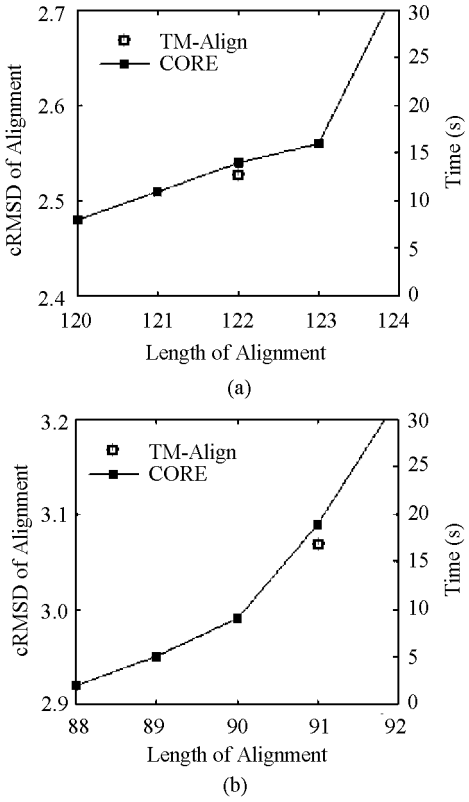

Fig.11. Graph representation of multiple alignments. (a) Alignment 1AXC with 2BEF. (b) Alignment 1CUV with 2ESY.

8 Conclusions

The most important aspect of designing algorithms for the protein structures alignment is the balancing of the two parameters, the alignment length and the similarity score (i.e., the cRMSD). In this case, a better result is expected, when the alignment length is long and cRMSD is small. However, our research has shown a positive correlation between the alignment length and the cRMSD such that increasing the alignment length is associated with an increase in the cRMSD. Therefore, the algorithm must incorporate the proper alignments that best balance the two parameters. However, this choice depends on what alignments are identified in each case.

The contributions of this paper are summarized as follows. 1) We proposed a method to produce multiple solutions of protein structures of different lengths and reduce the processing time by adopting CORE*. 2) Our algorithm makes it possible for users to visually choose the alignment they want from multiple solutions using a graph-based representation.

By adopting CORE* and DP, our proposed method produces multiple solutions of various lengths with higher accuracy than previous methods. In the experiments, the alignments identified by our algorithm are longer than those obtained by TM-align by 17% and 15.48%, on average, when the comparison was conducted at the level of super-family and folds, respectively.

References

[1] Ginalski K, Grishin N V, Godzik A, Rychlewski L. Practi-

cal lessons from protein structure prediction. *Nucleic Acids Research*, 2005, 33(6): 1874-1891.
[2] Roytberg M, Gambin A, Noe L *et al.* On subset seeds for protein alignment. *IEEE/ACM Trans. Computational Biology and Bioinformatics*, 2009, 6(3): 483-494.
[3] Mayr G, Domingues F, Lackner P. Comparative analysis of protein structure alignments. *BMC Structural Biology*, 2007, 7: Article No.50.
[4] Zhang Y. Protein structure prediction: When is it useful? *Current Opinion in Structural Biology*, 2009, 19(2): 145-155.
[5] Holm L, Sander C. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 1993, 233(1): 123-138.
[6] Dahiyat B I, Mayo S L. De novo protein design: Fully automated sequence selection. *Science*, 1997, 278(5335): 82-87.
[7] Yakunin A F, Yee A A, Savchenko A, Edwards A M, Arrowsmith C H. Structural proteomics: A tool for genome annotation. *Current Opinion on Chemical Biology*, 2004, 8(1): 42-48.
[8] Menke M, Berger B, Cowen L. Matt: Local flexibility aids protein multiple structure alignment. *PLoS Computational Biology*, 2008, 4(1): e10.
[9] Gu J, Bourne P. Structural Bioinformatics (2nd edition). John Wiley, 2009.
[10] Arun K S, Huang T S, Blostein S D. Least-squares fitting of two 3-D point sets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1987, 9(5): 698-700.
[11] Sippl M J, Wiederstein M. A note on difficult structure alignment problems. *Bioinformatics*, 2008, 24(3): 426-427.
[12] Chen L, Zhou T, Tang Y. Protein structure alignment by deterministic annealing. *Bioinformatics*, 2005, 21: 51-62.
[13] Glasgow J, Kuo T, Davies J. Protein structure from contact maps: A case-based reasoning approach. *Information Systems Frontiers*, 2006, 8(1): 29-36.
[14] Bhattacharya S, Bhattacharyya C, Chandra N R. Comparison of protein structures by growing neighborhood alignments. *BMC Bioinformatics*, 2007, 8: Article No.77.
[15] Kolbeck B, May P, Schmidt-Goenner T, Steinke T, Knapp E W. Connectivity independent protein-structure alignment: A hierarchical approach. *BMC Bioinformatics*, 2006, 7: Article No.510.
[16] Eidhammer I, Jonassen I, Taypor W. Structure comparison and structure patterns. *Journal of Computational Biology*, 2000, 7(5): 685-716.
[17] Shindyalov I N, Bourne P E. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 1998, 11(9): 739-747.
[18] Taylor W R, Orengo C A. Protein structure alignment. *Journal of Molecular Biology*, 1989, 208(1): 1-22.
[19] Taylor W R. Protein structure comparison using iterated double dynamic programming. *Protein Science*, 1999, 8(3): 654-665.
[20] Jewett A I, Huang C C, Ferrin T E. MINRMS: An efficient algorithm for determining protein structure similarity using root-mean-squared-distance. *Bioinformatics*, 2003, 19(5): 625-634.
[21] Lotan I, Schwarzer F. Approximation of protein structure for fast similarity measures. *Journal of Computational Biology*, 2004, 11(2/3): 299-317.
[22] Gibrat J F, Madej T, Bryant S H. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 1996, 6(3): 377-385.
[23] Kabsch W, Sander C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 1983, 22(12): 2577-2637.
[24] Frishman D, Argos P. Knowledge-based protein secondary structure assignment. *Proteins-Structure Function and Genetics*, 1995, 23(4): 566-579.
[25] Holm L, Sander C. 3-D lookup: Fast protein structure database searches at 90% reliability. In *Proc. the 3rd Int. Conference on Intelligent Systems for Molecular Biology*, July 1995, Vol.3, pp.179-187.
[26] Nussinov R, Wolfson H J. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *Proc. National Academy of Sciences of USA*, 1991, 88(23): 10495-10499.
[27] Le Q, Pollastri G, Koehl P. Structural alphabets for protein structure classification: A comparison study. *Journal of Molecular Biology*, 2009, 387(2): 431-450.
[28] Erdmann M A. Protein similarity from knot theory: Geometric convolution and line weavings. *Journal of Computational Biology*, 2005, 12(6): 609-637.
[29] Zhang Y, Skolnick J. TM-align: A protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 2005, 33(7): 2302-2309.
[30] Zhang Y, Skolnick J. Scoring function for automated assessment of protein structure template quality. *Proteins*, 2004, 57(4): 702-710.
[31] Godzik A. The structural alignment between two proteins: Is there a unique answer? *Protein Science*, 1996, 5(7): 1325-1338.
[32] Murzin A G, Brenner S E, Hubbard T, Chothia C. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 1995, 247(4): 536-540.
[33] Berman H M, Westbrook J, Feng Z *et al.* The protein data bank. *Nucleic Acids Research*, 2000, 28(1): 235-242.

Woo-Cheol Kim received the B.S., M.S. and Ph.D. degrees in computer science from the Yonsei University, Korea, in 2003, 2006 and 2010, respectively. Now he is a postdoctoral researcher in the Pennsylvania State University, USA. His research focuses on similarity search including bioinformatics.

Sanghyun Park received the B.S. and M.S. degrees in computer engineering from Seoul National University in 1989 and 1991, respectively. Then he received the Ph.D. degree in computer science from University of California at Los Angeles (UCLA) in 2001. His research interest includes database, data mining and bioinformatics, and now he is a professor of Department of Computer Science, Yonsei University in Korea.

Jung-Im Won received the B.S., M.S. and Ph.D. degrees in computer engineering from the Hallym University, Korea, in 1992, 1997 and 2004, respectively. Now she is a research professor in Hallym University in Korea. Her research interest includes database system, data mining and bioinformatics.