7-2010

New Algorithms for Protein Structure Comparison and Protein Structure Prediction

Zaixin Lu
University of Texas-Pan American

NEW ALGORITHMS FOR PROTEIN STRUCTURE COMPARISON AND PROTEIN STRUCTURE PREDICTION

A Thesis

by

ZAIXIN LU

Submitted to the Graduate School of the
University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

July 2010

Major Subject: Computer Science

NEW ALGORITHMS FOR PROTEIN STRUCTURE COMPARISON AND PROTEIN STRUCTURE PREDICTION

A Thesis
by
ZAIXIN LU

COMMITTEE MEMBERS

Dr. Bin Fu
Chair of Committee

Dr. Zhixiang Chen
Committee Member

Dr. Robert Schweller
Committee Member

July 2010

ABSTRACT

Lu, Zaixin, New Algorithms for Protein Structure Comparison and Protein Structure Prediction. Master of Science (MS), July, 2010, 39 pp., 3 tables, 11 figures, 59 references.

Proteins show a great variety of 3D conformations, which can be used to infer their evolutionary relationship and to classify them into more general groups; therefore algorithms of protein structure alignment, protein similarity search and protein structure prediction are very helpful for protein biologists. We developed new algorithms for the problems in this field. The algorithms are tested with structures from the Protein Data Bank (PDB) and SCOP, a Structure Classification of Protein Database. The experimental results show that our tools are more efficient than some well known systems for finding similar protein structures and predicting protein structures.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Proteins are sequences of amino acids. As we know, there are total 20 types of amino acids, and each of them has different physio-chemical properties. The length of proteins is usually several hundred long. Therefore, proteins show a great variety of 3D conformations, which are necessary to support their diverse functional roles.

Amino Acid	3-Letter Abbreviation	1-Letter Abbreviation
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

Figure 1 20 Types of Amino Acids

See Figure 1 for the 20 types of amino acids. It is widely believed that protein

sequences and structures have close relationship with their biological functions, while protein structures reveal more evolutionary information than protein sequences do, since the structure of a protein changes more slowly in the evolution than its sequence does. Also, researchers frequently find that proteins with low sequential similarity are structurally homogenous. Therefore it is particularly important to discover the structures of proteins.

In this thesis, we attempts to address some computational problems in the areas of protein structure comparison and protein structure prediction from its sequence.

Protein 3D Structure Comparison

Protein structures can be determined *via* experimental techniques such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy, and even cryo-electron microscopy. Due to these techniques, the number of proteins discovered by biologists has increased dramatically over the last 30 years. The rapid growth of the PDB (see Figure 2a of [1] for an illustration of the PDB growth rate from 1970's to the year 2005) necessitates the development of efficient and accurate protein structure comparison and search algorithms and automatic software tools.

Proteins have three main substructures, α-helix, β-sheet and loop. Figure 2 is a protein 3D structure. In order to compare the structural similarity between proteins, current protein structure alignment algorithms (e.g. [2-19]) usually try to align the C_α atoms in protein backbones. An alignment is characterized by (1) how many atoms are matched, (2) where their positions are, and (3) how well they are matched. (1) and (2) are available once an alignment is determined. For (3), a transformation based alignment algorithm usually calculates RMSD, namely, the root mean square distance between aligned (and transformed) C_α atoms in the structures. Although it has been studied for over 30 years, the protein structure alignment problem is far from being well

resolved. New approaches and improvements to existing approaches are frequently proposed (see [20-23] for some recent works). Moreover, many questions are still under active discussions.

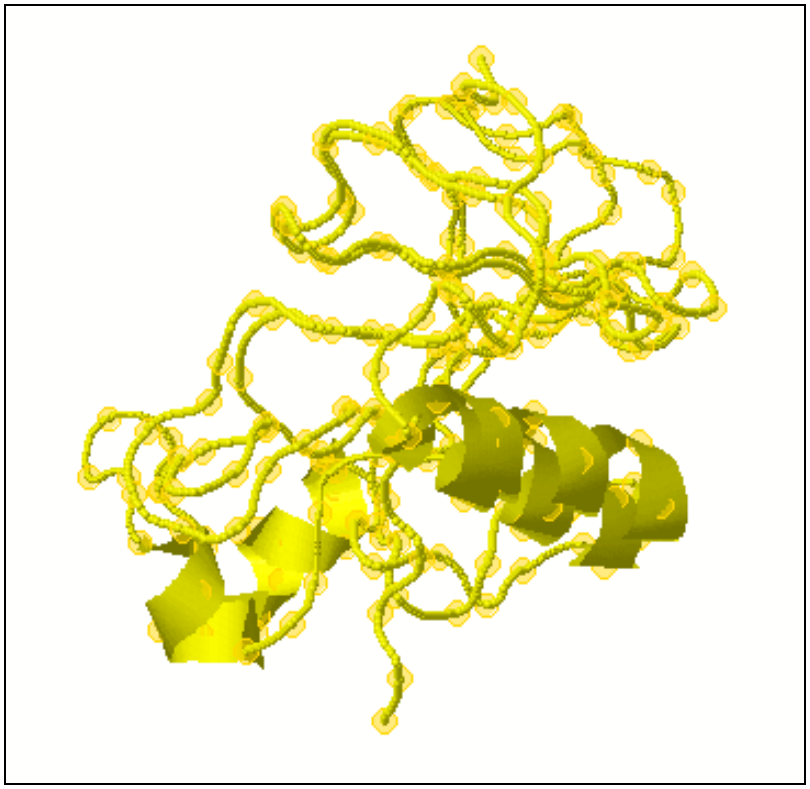

Figure 2 Protein 3D Structure

Protein structural similarity can be used to infer evolutionary relationship between proteins and to classify protein structures into more general groups; therefore a good protein structure alignment algorithm is very helpful for protein biologists. However, a good alignment algorithm itself may be insufficient for effective discovering of structural relationships among tens of thousands of proteins.

Protein Similarity Search

Protein structure query (e.g. [10,12,13,24-36]) aims to find similar structures in a protein dataset according to a given query structure. Due to the large size of protein data repositories like the PDB, protein structure query requires a very fast structure alignment tool; however, the

complexities of current alignment algorithms are usually too high to make a fully alignment-based search practical. For some proteins, it may take hours to days for protein structure search engines like CE [12] and DALI [37] to return a search result; a fast and accurate protein structure query tool which enables real-time structure searching in a large dataset is still in need. To improve the search speed, in recent years many methods have been designed to reduce the query time. Baker and Dauter (2004) developed SSM [8] which uses Secondary Structure Match for the pairwise structure comparison. In addition, various linear encoding methods have been applied to protein search systems. For instance, 3D-BLAST [33] developed by Yang and Tung (2006), can improve the comparison speed thousands of times as the speed of CE and DALI. Similar methods include ProtDex2 [35], Sarst [31], and TopScan [36]. These methods improve the time performance greatly. However, when being compared with pairwise alignment methods, they have weakness in accuracy.

Protein Structure Prediction

As mentioned before, protein structures can be determined via experimental techniques such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy, and even cryo-electron microscopy. However, these methods are relative expensive and time-consuming. Therefore, the output of protein structures is lagging far behind that of protein sequences. So it is reasonable that to solve proteins structures using the above experimental techniques for a small number of all the proteins and to apply computational techniques to predict the structures for the rest of proteins. Based on our knowledge, there are two main factors to make the protein structure prediction difficult. First the number of possible protein structures is very large. Second the protein folding process has not been fully understood by the biologists. Therefore, the protein structure prediction problem is that given a protein sequence, to first search all the possible

structures for the sequence, and to find the most likely structure based on a scoring system. In the past, many methods about this problem [48-59] have been proposed, and most of them can be categorized into three types: Homology Modeling, Threading and Ab Initio Modeling. The Homology Modeling-based methods align the sequences of new proteins whose structures are unknown to the sequences of proteins with known structures. Therefore, the accurate of this kind of method depends on how similar protein sequences can be found in the database, PDB. It is widely believed that proteins will fold into similar structures if they have similar sequences. However if the sequence identity is less than 25%, the accurate of Homology Modeling-based methods is very low. Unlike Homology Modeling, the Threading-based methods align the query protein sequence to structures of solved proteins in the PDB; the structures in the PDB are also called templates. A scoring system is needed to identify good matched templates, and they may have low sequential similarity with the query protein. The Threading-based methods are more efficient than the Homology Modeling-based methods for finding good templates at fold level. Furthermore if no good template can be found in the PDB library for a given protein sequence, the Ab Initio Modeling is using to predict its structure. This kind of approach assumes that the native structure of a protein corresponds to its global minimum energy and they construct as many structures as possible, and to identify the minimum energy structure. This is the most difficult category of protein structure prediction and now can only predict small proteins usually with less than 100 amino acids.

CHAPTER II

A PROTEIN ALIGNMENT ALGORITHM

In [47], we have developed an efficient protein pairwise alignment algorithm. It can align hundreds of pairs of protein structures in one second. Our experimental results show that it is as good as some well known alignment algorithms, and its speed is much faster than that of others. The algorithm has been fully implemented and is accessible online at the address http://fpsa.cs.panam.edu/.

Description of Algorithm

We give a brief description of our algorithms in this section. We have developed two protein alignment algorithms; both of them have three main stages. The first two stages are shared by them, but they have a different third stage. The third stage of the first alignment algorithm is based on DP (dynamic programming), and it preserves the order of C_α atoms in protein backbones. The second algorithm, whose third stage is based on MM (maximal matching), does not preserve the order, however it brings larger alignments than the first, while its speed is slightly slower. Our main technical contribution is a fast method used in stage two for finding a rigid body transformation to superimpose two protein structures. A process flow is shown in Figure 3.

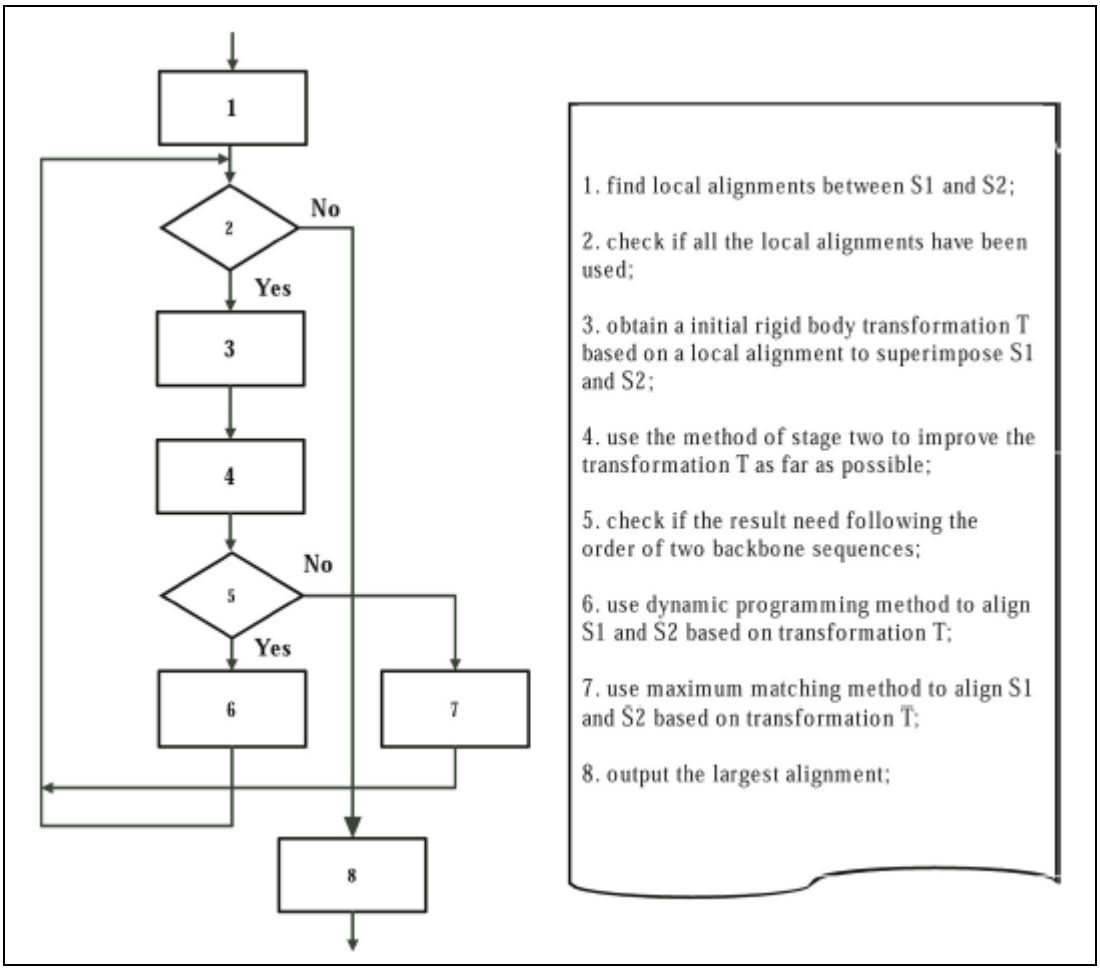

Figure 3 Flowchart of Our Alignment Algorithm

Brief-Algorithm

In the Figure 4**,** S_1 and S_2 are two 3D protein C_α backbone structures; L is a set of local alignments (a local alignment is a match of two substructures of consecutive C_α atoms from two backbones); δ is the maximum allowed distance between two matched C_α atoms; $F_G()$ is a function to calculate rigid body transformation; $T()$ is a function to translate and rotate a structure; find local alignments (stage 1);

```
for each local alignment l in L
  T = F_G(l) (beginning of stage 2);
  S2' = T(S2);
  repeat
    H = ∅;
    for every pair of points (pu, qv), where pu is in S1 and qv is in S2
      if distance(pu, qv) ≤ δ then put (pu, qv) into H;
    end for
    T = F_G(H);
    S2' = T(S2);
  until the number of pairs in H does not increase (end of stage 2)
  find aligned pairs between S1 and S2' from H (stage 3);
end for
output the largest alignment;
```

Figure 4 Brief Description of Our Alignment Algorithm

First stage

In the first stage, our algorithm searches for a set of local alignments, each consisting of a series of consecutive C_α atom pairs in the backbones of two proteins P and Q, which are represented by their C_α atoms in backbones $P = p_1p_2 \ldots p_{n1}$ and $Q = q_1q_2 \ldots q_{n2}$. We use (i, j, l) to represent a local alignment, which indicates that a gapless segment $p_ip_{i+1} \ldots p_{i+l-1}$ of the first protein backbone P starting at C_α atom i matches a gapless segment $q_jq_{j+1} \ldots q_{j+l-1}$ of second protein backbone Q starting at C_α atom j, and both segments have l atoms. We compute the distance matrices of the two backbones to match their local regions. If all the corresponding

distances in the two distance matrices have small difference, then a local alignment is found. Our algorithm to search all the local alignments runs in time $O(d_1 m_1 m_2)$, where d_1 is a constant number, m_1 is the number of C_α atoms in P and m_2 is the number of C_α atoms in Q.

Second Stage

In the second stage, each local alignment is used to find an initial rigid body transformation. There are many algorithms for finding a rigid body transformation to superimpose a set of pairs of 3D points [43]. In this paper a least square estimation method [44] is applied in our algorithm. Given a set $H = \{(a_1, b_1), (a_2, b_2),\ldots,(a_m, b_m)\}$ of point pairs in the 3D Euclidean space, $F_G(H)$ is a rigid body transformation T derived by the method in [44] to minimize the RMSD. Let H_0 be the set of all pairs (p_{i+k}, q_{j+k}) (k = 0, 1, ….., l - 1) in a local alignment (i, j, l). The rigid body transformation $T_0 = F_G(H_0)$ is derived. After obtaining T_0, we use it to superimpose the two structures and collect all the pairs (p_u, q_v), where p_u is from P and $q_v$is from Q and the distance between p_u and $T(q_v)$ is bounded by a threshold, and put them into H_1. A new rigid body transformation T_1, which is $F_G(H_1)$, is derived based on the new set H_1. By repetitively calculating a new transformation and adding new point pairs into H1, we can improve the transformation until no more pairs can be added.

A lot of matched pairs are used during the process of getting a transformation. It is necessary to mention that our algorithm selects all the matched pairs without considering conflicts, where two pairs share a same C_α atom. Obviously, this kind of conflicts is not disallowed in a global alignment. However, when calculating a transformation, there is no need to consider that, and sometimes it is hard to choose the best one among the conflicting pairs. Our method of searching a rigid body transformation with the presence of conflicting pairs not only makes the second

stage simple and fast, but also improves its accuracy.

Third Stage

In the third stage, we output an alignment that is a set of aligned pairs, where each C_α is allowed to appear in at most one pair. Two different methods, dynamic programming and maximal matching, are applied to bring the sequential alignment and non-sequential alignment, respectively.

The dynamic programming method can find an optimal solution by following the order of two backbone sequences. And the minimal length of a local alignment is set to 4. The maximal matching method returns a non-sequential alignment. Before applying the classical maximal matching algorithm to a graph of local alignments, we first delete edges in the graph so that each edge in the bipartite graph corresponds to a local alignment of length at least 4. The minimal length is used to exclude some isolated pairs, which are not biologically meaningful.

As we know, most existing protein alignment algorithms repeat finding the aligned regions between two backbones and recalculate the rigid body transformation when looking for a maximal alignment. Our algorithm does not involve the C_α atoms alignment when determining the rigid body transformation. This is why our alignment algorithm is remarkably faster than other algorithms. It can work out hundreds of pairs of protein alignments in one second. Therefore, it is very suitable for protein search. The sequential and non-sequential methods show an interesting tradeoff between speed and the number of aligned pairs.

Methods of speeding up the alignment and search

In order to speed up the computation of our basic algorithm, we propose some strategies that improve the time performance.

Finding a good rigid body transformation between two protein structures is often time

consuming. This is why most alignment algorithms are relatively slow. In order to develop an efficient protein alignment algorithm for protein search, we reduce computational time for finding the rigid body transformation while maintaining sufficient alignment quality.

First of all, the size of local alignments directly determines the time performance. A large number of local alignments will result in slow global alignments. Before finding the local alignments, we first filter out the entire short α helices which have length less than 8, because this kind of local structures are very common in proteins and may prevent the program from finding significant local alignments. Moreover, the length of a local alignment should be reasonably large to make sense, so we only consider local alignments with length greater than a threshold. In our experiments these two filters excluded about 85% unnecessary ones from all the possible local alignments.

Furthermore, each local alignment is used to calculate an initial rigid body transformation. However, it is possible that the final global alignments derived from different local alignments are the same or highly similar. Obviously, reducing these redundant local alignments can improve the time performance greatly. For each local alignment, we first apply every previously obtained rigid body transformation to it and calculate a corresponding RMSD value. A small RMSD by some transformation means that a final global alignment based on the current local alignment will be similar to that based on an old local, therefore the current one can be skipped. This technique efficiently reduces the unnecessary calculation of transformations. Supported by these effective filters, our alignment algorithm compares hundreds of pairs of proteins in one second, remarkably faster than other well known alignment algorithms.

Experimental Results

In this section, we show the experimental results for an implementation of our algorithm and

its comparisons with other well known similar systems which are accessible online. The quality evaluation of protein alignment algorithm is based on its alignment length and RMSD value. The quality evaluation of protein search tool is according to its precision, recall, and query speed.

Evaluation of the alignment results

There is no general standard for analyzing and comparing the results of different alignment algorithms, because each method uses different alignment measures. Besides the two basic measures: alignment length and RMSD value, some methods also calculate a native score for their alignment results. For instances, CE and Dali use different kinds of Z-scores as their native score, SSM has the Q-score which considers both alignment length and RMSD value when measuring the alignment results. Subbiah proposed a geometric match measure, SAS_k , in [38]. The SAS_k also considers both alignment length and RMSD value. Lower SAS_k means better alignment result and k is the degree to which the score favors. A smaller k can be used when longer alignment length is preferred and a bigger k is for smaller RMSD value. Here we have collected 224 alignment cases and used Q-score and SAS_k to test the performance of our algorithm.

The test cases were originally proposed by various papers for various testing purposes. A list file available on our website shows all the 224 cases. They include No. 1 - No. 20 (see Table III in [12]), No. 21 - No. 88 (see Table I in [24]), No. 89 (see Tables I and II in [12]), No. 90 - No. 92 (supplement to Table III in [12]), No. 93 (see Figure 5 in [12]), No. 94 - No. 101 (see Table IV in [12]), No. 102 - No. 111 (see Table V in [12]), No. 112 - No. 120 (supplement to Table V in [12]), No. 121 - No. 124 (see Table VII in [12]), No. 125 - No. 143 (see Table 1 in [11]), No. 144 - No. 183 (see Table 1 in [17]) and No. 184 - No. 224 (see Table 2 in [17]). We compare our alignment results with Dali, CE and SSM. In each test case, different alignment algorithms have

different results. CE and Dali always get more aligned pairs than those of our algorithm and SSM, but their accuracy is relatively lower (having larger RMSD value). So using merely aligned pairs or RMSD value as the criterion to measure the performance of alignment algorithm makes no sense. Therefore, we calculate Q-score and SAS_k for the alignment results of all the methods and compare the alignment results in terms of *Q*-score Difference and Average SAS_k. The *Q-score Difference* is calculated by (Q_{score_Ours} - Q_{score_X}) where *X* is Dali, CE or SSM. And we use k = 1, 2 and 3 for the Average SAS_k to analyze the quality of all the four methods. It should be mentioned that our method can output sequential alignments and non-sequential alignments, thus we compare both of them with other methods. We call the sequential method SPSA and the non-sequential method NPSA for short.

Table 1 Results of Multiple Alignment Methods

	Dali	**CE**	**SSM**	**SPSA**	**NPSA**
Average alignment length	130.43	132.82	117.78	119.20	122.65
Average RMSD	2.78	2.83	2.37	2.23	2.30
Average SAS_1	2.96	3.08	2.76	2.62	2.48
Average SAS_2	3.89	3.60	3.92	3.43	3.25
Average SAS_3	6.67	5.69	6.84	5.60	5.19

Discussion on the alignment results

Table 1 show some statistical data based on the experimental results. Compared with Dali, CE and SSM, our algorithm has smaller average RMSD, and its average alignment length is longer than that of SSM, but shorter than that of CE and Dali. Its average SAS_k is always smaller than other three algorithms, no matter k = 1, 2 or 3. A lower SAS_k score indicates a better alignment. To further test our algorithm, we compare it with others by Q-score. Figure 5 reflects the *Q*-score Difference between our algorithm and others respectively. A black area below X-axis indicates that the Q-score of our algorithm is lower than that of the compared method. Since in each graph the upper part of the whole black area is always larger than or equal to the lower part, it is clear that our alignment algorithm is comparable to other well known algorithms. It is worth mentioning that in most test cases, SSM and our method always give alignment results with small RMSD value and shorter alignment length, while CE and Dali always find more matched pairs but with larger RMSD value. So if more matched pairs are desirable, Dali and CE are good alignment tools; on the other hand, if shorter but more accurate alignments are preferred, SSM and our method are better.

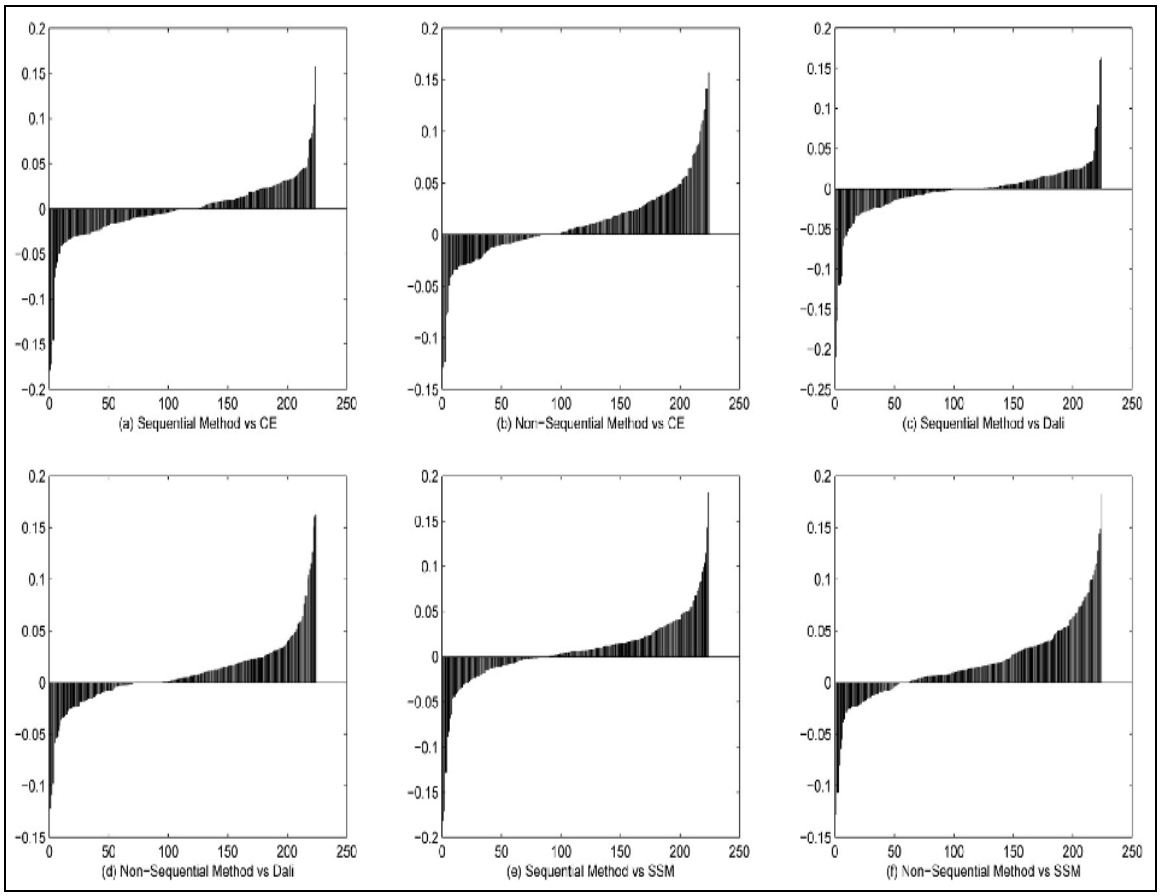

Figure 5 Q-score Difference Plot

CHAPTER III

A PROTEIN SEARCH ALGORITHM

In [34] we have developed a protein structure query algorithm and tool to find similar protein structures in the PDB for any given structure. With a combination of geometric filter and 3D structure alignment, given a query protein, the algorithm can find proteins whose structures are overall similar with the query structure in the PDB in a few minutes. The geometric filter can exclude dissimilar proteins efficiently, and reduce a lot the number of times of pairwise alignments. On the negative side, it misses some significant proteins whose structures are partially similar with the given protein. In [46], to further improve the accuracy and efficiency our protein search algorithm, we developed new algorithm and web tool to search similar protein structures in the PDB (Protein Data Bank). The algorithm is a combination of a series of methods including protein classification, geometric feature extraction, sequence alignment, and 3D structure alignment. Given a protein structure, the tool can efficiently discover similar structures from hundreds of thousands of structures stored in the PDB. Our experimental results show that it is more accurate than other well-known protein search systems in finding proteins that are structurally similar to the query protein, and its speed is also competitive with those systems. The algorithm has been fully implemented and is accessible online at the address http://fpsa.cs.panam.edu/, which is supported by a cluster of computers.

Outline of Our Approach

Our method is a combination of sequence alignment and geometric alignment. Its speed is improved by grouping proteins with similar structures and using one representative structure for each group. A brief overview about our method is as follows:

(1) An offline phase partitions the protein 3D structures in the database into groups so that each group contains those proteins with similar structures. One representative structure is selected from each group.

(2) When an input protein is given, use the BLAST algorithm to search proteins with similar amino acid sequences and put them into list L1.

(3) Use several layers of geometric filters to check all the representative proteins among the classified protein database and exclude dissimilar proteins. Put all those similar representative proteins into list L2.

(4) Put all the representative proteins whose groups are related to proteins in L1 into L3, and let list L4 = L3 − L2 (which are the proteins in L3, but not in L2).

(5) Use a simplified version of our pair-wise 3D structure alignment algorithm to check structural similarities between the input protein 3D structure and each structure in L2 and L4. Output those groups whose representatives are structurally similar to the input protein. This is the most time-consuming part; therefore, it is implemented in a cluster of computers.

The first filter, called “sequence filter”, is based on the sequence alignment of BLAST, and the second filter, called “geometric filter”, is based on some simple geometric comparisons. A system process flow is shown in Figure 6.

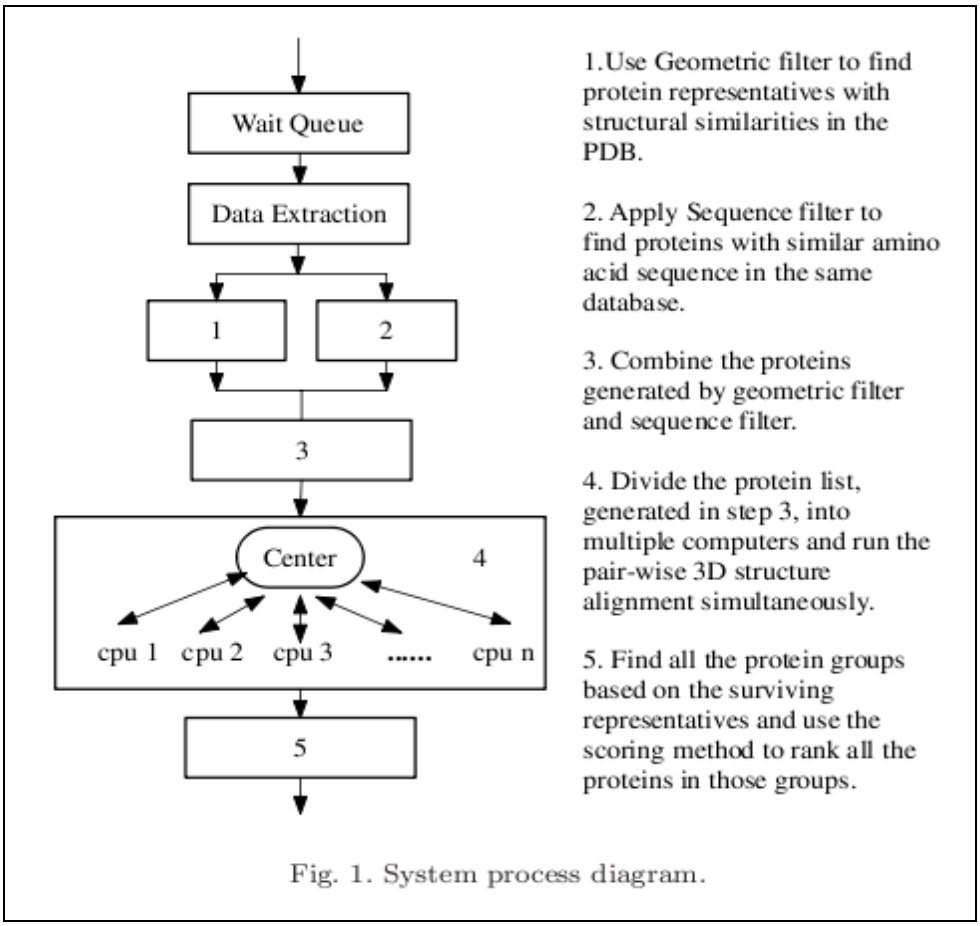

Figure 6 System Flowchart

An Offline classification

Our search system has an offline classification for all the protein structures in the PDB. The purpose of this classification is to improve the speed of protein structure query. It has the following steps:

1. Partition the PDB into groups. Use our pairwise alignment algorithm to check the structural similarity among the protein chains in the PDB. The Q-score, proposed in [8] is a non-linear score for measuring such similarity. It has been found that different protein structure query servers agree reasonably well on this score. Here we support that two proteins with alignment Q-score higher than 0.5 are similar. Therefore, after the partition every two proteins in the same group have a pairwise alignment Q-score of at least 0.5.

2. Select representatives. For each group, one protein chain is selected as a representative. In order to get it, we align each protein with all others in the same group and calculate the sum of Q-scores. A protein chain with the highest sum of Q-scores is selected as the representative. The classification is based on all-against-all pairwise alignment. It synchronizes itself with the PDB and all the protein chains in the PDB are partitioned into about 14,000 groups. Searching the entire PDB with 130,000 protein chains is reduced to doing so in our classified database with about 14,000 representatives. Thus, it improves the speed to a great extent.

Distributed Computing

In the last step of implementation we assign the surviving proteins to multiple computers to perform the pair-wise structure alignments simultaneously. However, if not scheduled properly, a distributed system can decrease the overall reliability of computations because the unavailability of a node can result in disruption of other nodes. Instead of just evenly assigning proteins to the nodes of our cluster, at the beginning, the front node assigns a small number of proteins to each available machine of the cluster, then whenever a machine is free (i.e. it has completed its task), the front node will calculate the speed of that machine and send a certain number of proteins to it. This procedure is repeated until all the computation tasks have been completed.

Experiment Results

In this section, we show the experimental results for an implementation of our algorithm and its comparisons with other well known similar systems which are accessible online. The quality evaluation of protein alignment algorithm is based on its alignment length and RMSD value. The quality evaluation of protein search tool is according to its precision, recall, and query speed.

Evaluation of the query accuracy

When applying alignment algorithms into protein search systems, the algorithms which can

output best alignment results might not have the best performance of search accuracy.

The Structural Classification of Protein database (SCOP [39-42]), manually constructed by human experts, is believed to contain accurate structural classifications. The SCOP hierarchy has the following levels: domain, family, superfamily, fold, and class.

For a database search tool, the recall rate and precision are two commonly used parameters for assessing its query quality. Precision is defined as n/N and recall rate is n/T, where n is the number of true proteins (from the same family of the query protein) in the result list, N is the total number of proteins in the result list, and T is the total number of proteins in the same family of the query protein in the database. Therefore, the precision is between 0 and 1, and the quality of a ranked output list is directly based on it. The recall rate is also between 0 and 1, and the missing problem of a search engine is in relation to it. In 2004, Aung and Tan [35] collected 34,055 proteins form the ASTRAL SCOP 1.59 to form a large target database, from which 108 proteins were selected as the query proteins. These query proteins are from four main classes (All-α, All-β, α/β and $\alpha + \beta$) of ASTRAL SCOP 1.59 and their average family size is around 80. We use these different categories of proteins to do the queries on our search engine, use Q-score as the criterion to rank the output proteins, and compare the result with that of CE [12], MAMMOTH [11], 3D-BLAST [33], PSI-BLAST [29], ProtDex2 [35], and TopScan [36]. Results for all the methods except ours are taken from [33]. Since our sequential method is much faster than its non-sequential counterpart, we just use the sequential method in our experiments. In our search system we do support both sequential and non-sequential methods.

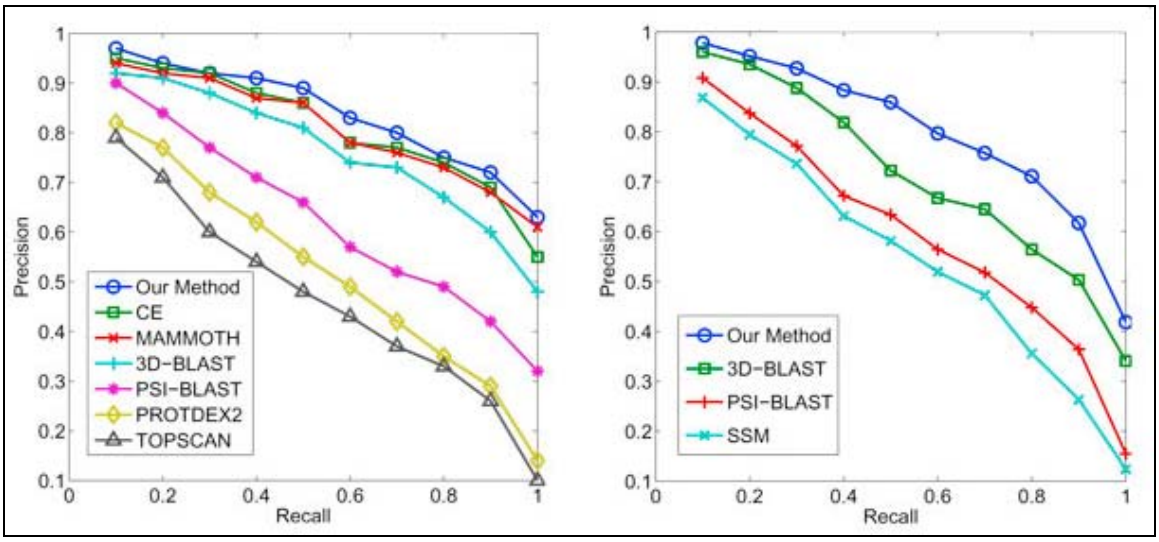

Figure 7 Precision and Recall Curves

According to the 108 query results, our method based on pairwise alignment algorithm shows better performance than others. CE and MAMMOTH are the second and third accurate methods. 3D-Blast, which is based on a linear encoding method, ranks fourth and its precision is about 5% lower than ours on average. PSI-BLAST is a classical sequence search algorithm, and its precision outperforms both TopScan and ProtDex2. However, when being compared with alignment algorithms such as CE, MAMMOTH and ours, its precision is much lower. Figure 7 (left) shows the Receiver Operating Characteristic (ROC) curves for all the methods. When the recall rate is 100%, the average precision of our method is 63%, which is the highest. In addition, all the pairwise alignment methods including CE, MAMMOTH and ours have higher precision than that of other methods, and with the increasing of recall, this trend becomes more obvious.

Performance for searching weak similarities

In the SCOP database, proteins in a same species or domain are the most similar proteins, and then are the proteins belonging to a same family. Experts also classify weakly similar proteins into a same superfamily. Therefore, to precisely assess the efficiency of searching methods challenged by searching weakly similar proteins, we use the entire ASTRAL SCOP 1.73

as the target database, and select 129 query proteins belonging to four major classes. The average superfamily size of these 129 query proteins is around 300. We use these proteins to do queries on our search engine and also on 3D-BLAST [33], PSI-BLAST [29], and SSM [8]. We are aware that ProtDex2 [35], Sarst [31], and TopScan [36] are also famous protein search systems, however they have not updated their database for a long time. Figure 7 (right) shows the experimental results of the four methods for searching similar proteins at superfamily level, according to which our method is the most accurate one. 3D-BLAST, the second most accurate method in this experiment, has its precisions about 8.1% lower than ours on average. We claim that pairwise alignment algorithms have more advantages in finding remote homologous proteins than linear encoding method or sequential search method does. Nevertheless, according to the experimental result, our method occasionally has problem detecting related proteins in the same superfamily. About 17% query results of our search engine have serious missing problems (precision lower than 50% when recall rate is 100%), while 3D-BLAST is 35% and other methods have more serious missing problems than 3D-BLAST.

Evaluation of reliability

Our search engine provides an alignment length and an RMSD value for every retrieved structure. In order to assess the reliability of our search engine, we calculate the Q-score value of each structure by its alignment length and RMSD value; and gather statistics on precision and recall rate for various Q-score values at both superfamily and family levels. According to the data in Table 2, when Q-score is higher than 0.4, the average precisions are over 90% for both levels. The recall rates are 62.05% for family level and 45.22% for superfamily level. Precision is defined as n/N and recall rate is defined as n/T, where n is the number of true proteins of Q-scores higher than the limit value in the result list. A true protein means it is from the same

family or superfamily of the query protein. N is the total number of retrieved proteins whose Q-scores are higher than the corresponding value, and T is the total number of proteins in the family or superfamily of the input protein.

Table 2 Statistics on the Reliability of Our Query Engine.

Q-Score	**0.9**	**0.8**	**0.7**	**0.6**	**0.5**	**0.4**	**0.3**	**0.2**
avg.recall(%) family	14.65	20.20	28.14	39.28	48.99	62.05	75.84	84.31
avg.precision(%) family	99.35	98.56	97.42	97.16	94.83	91.57	87.43	70.85
avg.recall(%) superfamily	9.22	12.89	18.96	27.53	34.87	45.22	56.59	68.43
avg.precision(%) superfamily	99.39	99.15	99.01	98.89	98.74	97.58	96.60	87.39

Evaluation of query speed

Results for all the methods except ours are taken from [33]. Their experiments were performed on a computer with an Intel Pentium 2.8 GHz processor and 1,024 megabytes of RAM memory. Ours were done on a computer with Intel Pentium 2.66 GHz processor and 1,024 megabytes of RAM memory. As shown in Table 3, on average, our method requires about 112.20 seconds searching the database for each query protein in a single machine. Although it is much slower than 3D-BLAST and PSI-BLAST, when being compared with pairwise alignment algorithms CE, and MAMMOTH, ours has great advantage in the time performance. In addition,

we have used our web server and the web servers of 3D-BLAST, PSI-BLAST and SSM to do the 129 queries in the entire PDB database. The PSI-BLAST, SSM, and 3D-BLAST servers have average query time of 16, 27, and 44 seconds respectively. Benefit from the distributed computing system and the offline classification, our web tool can scan the entire PDB database in 17 seconds on average, which is shorter than those of 3D-BLAST and SSM. Moreover, our results contain an alignment length and an RMSD value for every output protein. Although PSI-BLAST and 3D-BLAST do not have these data, they are the most important measures for comparing protein structural similarities. The 3D-BLAST server is the slowest one with an average query time of 44 seconds. In our knowledge, other search engines such as CE [12] and Dali [37] which are based on one-against-all pairwise alignment algorithms need hours to days to complete the queries.

Table 3 Average Search Time of Multiple Program on 108 Queries

Software	**Total search time (s)**	**Average search time per query (s)**
Our method	12,117	112.20
3D-Blast	34.35	0.318
PSI-BLAST	18.31	0.170
CE	13.5 days	3 hours
MAMMOTH	131,855	1220.88

CHAPTER IV

A PROTEIN PREDICTION ALGORITHM

It is often said that if a sequence has high percentage identity to a sequence of known structures, it is possible to build a reasonable structure by Homology-based Modeling, but at lower than 30% identity, Threading-based Modeling is better than Homology Modeling to predict a structure. The main idea of Threading-based Modeling is a one dimensional to three dimensional alignment. The query protein sequence is one dimensional and the template in the database is three dimensional. In the past many approach has been proposed for the protein threading problem. The problem can be formulated to an integer programming problem and it was converted to a linear programming problem by Xu and Li in 2003 [48].

Linear programming (LP) is a mathematical method to achieve the best outcome in a given optimizing problem for some list of requirements represented as linear equality and linear inequality constraints. The protein threading problem can be converted to a LP problem by a constraint system and a score system. The constraint system is to ensure that two or more elements in the query sequence cannot share a same position in the template. In addition, the alignment is sequential, which means that the matching preserves the order of both the sequence of query protein and the atoms in template. The score system includes many different score functions. When a element of the query sequence aligns to a position in the template, it obtains a score according to the score functions.

The linear programming is a technique for the optimization of the sum of scores, subject to linear equality and linear inequality constraints in the constraint system.

Our Contribution

The Protein threading is one of the most challenging problems in the field of computational biology today. It has a large number of literatures available now; in the previous works the problem always has the following definition: The protein model is a linear sequence of secondary structure segments, called conserved cores. Core segments have fixed length and they are connected by loop regions. When doing the sequence-structure alignment, the score system considers the element in the core segments and the loop regions are only considered for gap penalty. In addition there are two main types of threading models:

1. The score functions do not include the pairwise interaction score. This kind of threading problems can be solved simply by applying a dynamic programming algorithm.
2. The score functions include the pairwise interaction score. This kind of problems is NP hard when variable gaps are allowed into the alignment in the loop regions (Lathrop, 1994).

Therefore, any method must require an exponential time to find an optimal solution when using the second model, and doing exhaustive search is impractical for large proteins.

Xu and Li converted the threading problem to a linear programming problem in 2003, and they show that it has good accurate performance comparing with other famous protein structure prediction servers. As we investigated, when applying the linear programming to a protein threading problem, it may has numerous variables in the constraints that make the computation very slow. In order to further improve the efficiency and accuracy of the linear programming algorithm, we proposed a heuristic algorithm to reduce variables unnecessary in the linear

constraints.

Our approach first checks all the possible alignments for each core, and selects the alignment pairs which have good solvent, profile, and secondary structure match. After that, we just select one alignment as a center each time, and filter out all the alignments uncooperative. An alignment is uncooperative if it has large gap with the center. Then we do linear programming to find a optimal solution with the rest alignments. This method reduces the number of variables greatly, and improves the time performance. In addition, our experimental results show that, comparing with the original method, it always finds the same optimal solutions.

We also developed a dynamic programming method for the protein threading problem. It can find an optimal solution by following the order of both the query protein sequence and the atoms in a template if variable length gaps are allowed in the loops and pairwise interactions are considered only for the elements between neighboring cores.

Outline of Our Method for Linear Programming

Score functions

There are total four kinds of score functions and a gap penalty function used in this paper:

$$S_{total} = W_m S_m + W_{ss} S_{ss} + W_{so} S_{so} + W_p S_p + W_g S_g$$

The first term in the equation is a sequence profile matching score. The profile of the query sequence is obtained by using the classical sequence search program PSI-BLAST to search against the non redundant database NR. Given a query protein of length n, it is a $[n, 20]$ matrix includes the frequency $P(i,k)$, which is the probability of the kth amino acid at the ith position of the query protein. If the ith position of query sequence is aligned to the jth position of a template, their sequence profiles matching score is $\sum P(i,k)M(j,k)$, where $k = 1...20$, and M is PAM250 matrix.

The second term is to compare the different between the predicted secondary structure of a query sequence and real secondary structure of a template. We use PSI-PRED to predict the secondary structure for the query protein and use DSSP to obtain that of the template.

The third term is similar with the second one. It calculates the solvent accessibility match between a query protein and a template. The solvent accessibility for the query is predicted by the program SABLE and the real solvent accessibility of the template is obtained by DSSP.

The fourth term is a pairwise interaction score. The score is follows Xu and Li [48], and it is to compute the potential between two amino acids, which has short distance.

The last term in the equation is the gap penalty, and it includes a gap open penalty and a gap extension penalty.

Improve the time performance

The variables and constraints of linear programming used in this paper follow Xu and Li [48]. There are two types of variables. Let $x_{i,l}$ denotes a Boolean variable and $x_{i,l} = 1$ if the *i*th core of the template is aligned to lth position of the query sequence, otherwise it is 0. The other kind of Boolean variable $y_{i,l,j,k}$. $y_{i,l,j,k} = 1$ if both $x_{i,j} = 1$ *and* $x_{j,k} = 1$, otherwise it is 0. It is obvious that the variable $y_{i,l,j,k}$ is for the pairwise interaction score and the $x_{i,l}$ is for other single scores.

In this paper, our work focuses on reducing the number of variables in the linear programming, while avoid missing good templates for the query protein. Given a query protein sequence, we first find all the valid query sequence positions that each core of the template could align to, and select the core *C* that has the minimum number of possible positions. The valid positions for each core are selected according to their solvent, profile, and secondary structure matching scores. Let *P* denotes the set of all the positions for core *c.* we align core *C* to each

position in *P*, and consider this alignment as a center to filter out all the positions for other cores uncooperative. A position is uncooperative if its gap with the center is larger than the maximum gap defined by us. We do linear programming with the rest of valid positions each time and select the final alignment which has the best score. This method reduces the number of variables a lot, and improves the time performance greatly.

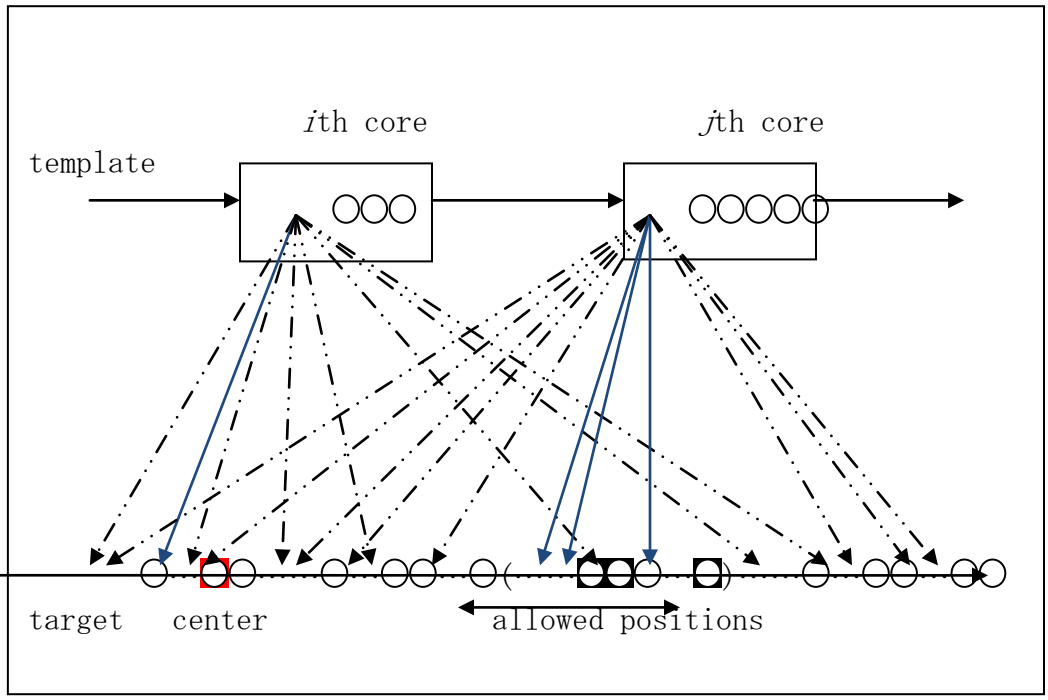

Figure 8 Example of Our Filtering Method

Figure 8 shows an example of our filtering process. At the beginning, all the lines are the possible alignment positions for the two cores. As well as we align the *i*th core to some position, in this example it is the red position, all the dashed lines from the *j*th core can be ignored, because they have big gap with the center. In addition all the dashed lines from the *i*th core can be deleted, because the *i*th core is aligned to the read position fixedly.

Suppose the two cores has pairwise interaction, and the number of all the possible query sequence position for the *i*th core is *m*, the number of that for the *j*th core is *n*. Thus the number of the *x* type variables of the *i*th core and *j*th core are *m* and *n* respectively. The number of the *y*

type variables for these two cores is $O(mn)$. Therefore, our method not only reduces the x type variables, but also reduces y type variables significantly.

Our Dynamic Programming Algorithm

As we know if the scoring system does not include the pairwise interaction score, the threading problem can be solved by a simple dynamic programming algorithm. However if pairwise interaction is considered, dynamic programming cannot be applied to optimize the score. In this paper, we developed a dynamic programming method for finding the optimal solution subject to (1) variable length gaps are allowed in the loops and (2) pairwise interactions are considered only for the elements between neighboring cores. Figure 9 briefly describes our algorithm.

The number of possible query sequence position for each core is less than the length of the sequence. Therefore, the complexity of our dynamic programming is $O(n^2m)$, where n is the length of query sequence and m is the number of cores in the template.

The Input is a template *Ta* and a query sequence *S*. The output is the optimum matching score for *Ta* and *S*.

Let *M* denotes the score matrix and *n* denotes the number of the cores in *Ta*. *Fs()* is the function to calculate the profile, secondary structure and solvent scores and *Fp()* is to compute the pairwise score.

1. For *i = 1* to *n*

2. Let *m* denotes the number of valid query sequence position for *i*th core, *m'* denotes the number of that for *(i-1)*th core, and *D[i]* denotes the corresponding query sequence position set;

3. For *j = 1* to *m*

4. If *i* is *0*

5. *M[i][j] = Fs(Ta[i],S[D[i][j]]);*

6. Else

7. *M[i][j] = Max(Fs(Ta[i],S[D[i][j]])+M[i-1][k]+Fp(S[D[i][j]],S[D[i-1][k]]))*, where *k = 1, 2, ..., m'*;

8. Output *Max(M[n][l])*, where *l* is from *1* until *u*, where *u* is the number of positions last core could align to;

Figure 9 Our Dynamic Programming Algorithm

Experimental Result

In this section, we show the experimental results for an implementation of our algorithm and some examples of its prediction result.

The algorithm is implemented with C program, and worked on a personal machine with an Intel Pentium 1.66 GHz processor and 2,048 megabytes of RAM memory. We test the time performance of our method by using sequences from the CASP9 (Critical Assessment of Techniques for Protein Structure Prediction) competition as targets. In the CASP9 set, we have 91 target sequences, 74 from them have length between 100 than 500. For each of those targets, a suitable template, which has 15%-25% sequence similarity is identified by using PSI-BLAST to

search against the PDB. We do the target-template alignments for the entire 74 pairs by the original linear programming algorithm and our new algorithm. The two methods share the same constraint system and score system. Their average running time is 14.4 seconds and 155.8 seconds respectively, our new algorithm is about 12 times faster than the original one. Most of the pairs have the same alignment results and optimal scores for the two methods expect one pair, the target T0526 and template 1JOVA. The optimal score obtained by the original method for this pair is 6286.47 and the optimal score by our algorithm for it is 6116.67. The slight difference is for the reason that a gap in the alignment of the original method is larger than the maximum gap allowed in our new algorithm.

We also use our dynamic programming method to do the alignment for the above test pairs. The average running time is 2.9, which is much faster than that of the approaches based on linear programming. However when using the same score system to verify its alignment result, the scores obtained by the alignments of our dynamic programming method are lower than that of our linear programming for most of the test cases. It is reasonable since our dynamic programming method is for templates that only have pairwise interactions between neighboring cores, or the pairwise interactions are only considered for the elements between neighboring cores.

We test the accuracy of our protein structure prediction algorithm by sequences from newly discovered proteins in the SCOP1.7.5. Since our algorithm is relatively fast, we construct a large non-redundant templates database which contains about 13000 proteins from SCOP1.7.3. Given a target sequence with length n, our algorithm only considers the templates whose lengths are between $2n$ and $n/.2$. We present an example of the prediction result by our algorithm.

In the experiment, we use the target sequence of a protein whose PDB code is 2F15A, it has

89 residues and is classified into All-β class in the SCOP1.75. Its most suitable template founded by our linear programming algorithm and dynamic programming algorithm in the template database has PDB code 2BHUA and 1M7XA respectively. To investigate how structurally similar they are with the target protein, we use our protein structure comparison tool to compare their structures. The target protein and the template protein 2BHUA have 69 matched pairs and the RMSD value is only 1.8, which means they are structurally similar. Figure 10 shows the comparison details.

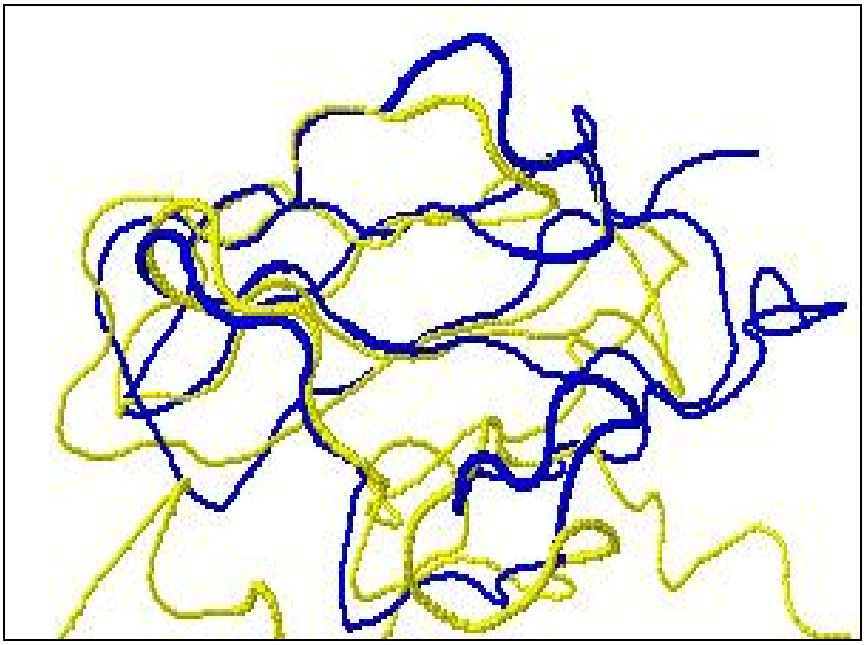

Figure 10 Structure Comparison Result for 2BHUA

Figure 11 shows the structure comparison details for the target protein and the template 1M7XA, which is the best template according to the result of our dynamic programming algorithm. They have 76 matched pairs and the RMSD value is 2.4. It is obvious that the two structures in the figure are very similar. The dynamic programming algorithm also has good performance in our experiment.

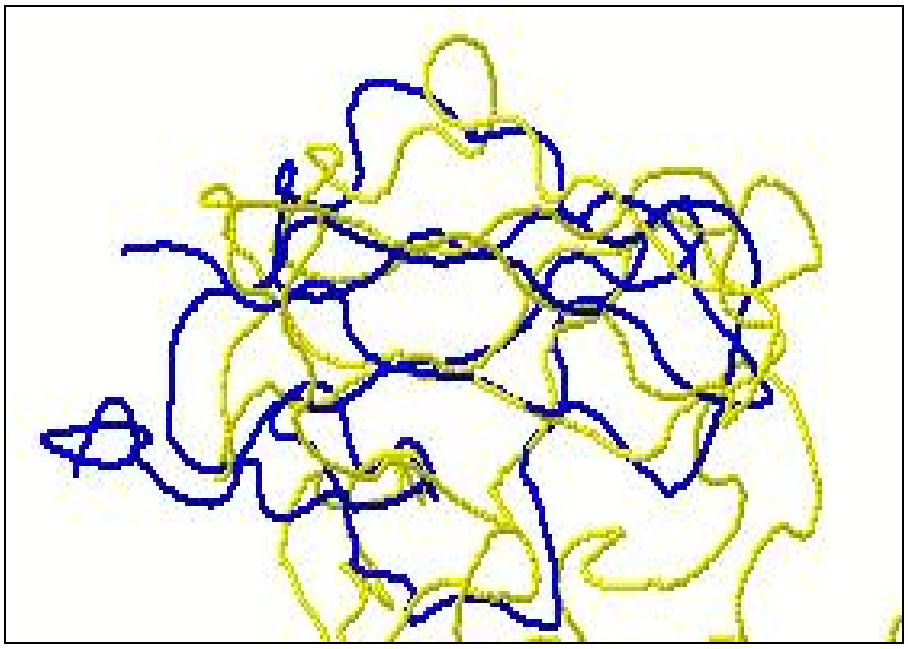

Figure 11 Structure Comparison Result for 1M7XA

Our protein threading algorithm is fast and accurate. This makes it very suitable for large scale computing. In the future we want to improve the score system of the threading approaches and apply it to predict the folds of SCOP for newly discovered proteins.

REFERENCES

[1] Levitt M: *Growth of novel protein structural data.Proceedings of the National Academy of Sciences of the United States of America* 2007, *104:*3183-3188.

[2] Chew LP, Kedem K, Huttenlocher DP, Kleinberg J: *Fast detection of geometric substructure in proteins.J of Computational Biology* 1999, *6*(3-4)*:*313-325.

[3] Falicov A, Cohen FE: *A surface of minimum area metric for the structureal comparison of protein.Journal of Mol Biol* 1996, *258:*871-892.

[4] Fischer D, Nussinov R, Wolfson H: *3D substructure matching in protein molecules.Proc 3rd Intl Symp Combinatorial Pattern Matching, Lecture Notes in Computer Science* 1992, *644:*136-150.

[5] Holm L, Sander C: *Protein structure comparison by alignment of distance matrices.J Mol Biol* 1993, *233:*123-138.

[6] Ilyin VA, Abyzov A, MLeslin C: *Structural alignment of proteins by a novel TOPOFIT method, as a superimposition of common volumes at a topomax point.Protein Science* 2004, *13:*1865-1874.

[7] Kolodny R, Linial N: *Approximate Protein Structural Alignment in Polynomial Time.Proceedings of the National Academy of Sciences of the United States of America* 2004, *101*(33)*:*12201-12206.

[8] Krissinel E, Henrick K: *Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions.Acta Crystallogr D Biol Crystallogr.* 2004, *60*(12)*:*2256-2268.

[9] Lessel U, Schomburg D: *Similarities between protein 3-D structures.Protein Engineering* 1994, *7*(10)*:*1175-87.

[10] Madej T, Gibrat JF, Bryant SH: *Threading a database of protein cores.Proteins* 1995, *23:*356-369.

[11] Ortiz A, Strauss C, Olmea O: *MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison.Protein Science* 2002, *11:*2606-2021.

[12] Shindyalov IN, Bourne PE: *Protein structure alignment by incremental combinatorial extension (CE) of the optimal path.Protein Eng* 1998, *11:*739-747.

[13] Singh AP, Brutlag DL: *Hierarchical protein superposition using both secondary structure and atomic representation.Proc Intelligent Systems for Molecular Biology* 1997, 284-293.

[14] Taylor WR, Orengo C: *Protein structure alignment.J Mol Biology* 1989, *208:*1-22.

[15] Taylor WR: *Protein structure comparison using iterated double dynamic programming.Protein Science* 1999, *9:*654-665.

[16] Ye Y, Godzik A: *Database searching by flexible protein structure alignment.Protein Science* 2004, *13*(7)*:*1841-1850.

[17] Ye J, Janardan R, Liu S: *Pairwise protein structure alignment based on an orientation-independent backbone representation.Journal of Bioinformatincs and Computational Biology* 2005, *4*(2)*:*699-717.

[18] Yona G, Kedem K: *The URMS-RMS hybrid algorithm for fast and sensitive local protein structure alignment.Journal of Computational Biology* 2005, *12:*12-32.

[19] Zhang Y, Skolnick J: *TM-align: a protein structure alignment algorithm based on the TM-score.Nucleic Acids Research* 2005, *33*(7)*:*2302-2309.

[20] Zhao Z, Fu B: *A Flexible Algorithm for Pairwise Protein Structure Alignment.Proceedings International Conference on Bioinformatics and Computational Biology 2007* 2007.

[21] Zhao Z, Fu B, Alanis FJ, Summa CM: *Feedback Algorithm and Web-Server for Protein Structure Alignment.Journal of Computational Biology* 2008, *15:*505-524.

[22] Salem S, Zaki MJ: *Iterative Non-Sequential Protein Structural Alignment.Proceedings of the 7th Annual International Conference on Computational Systems Bioinformatics (CSB'08)* 2008.

[23] Jiang M, Xu Y, Zhu B: *Protein Structure Structure Alignment With Discrete Frechet Distance.Journal of Bioinformatics and Computational Biology* 2008, *6:*51-64.

[24] Alexandrov NN, Fischer D: *Analysis of topological and montopological structural similarities in the PDB: new examples from old structures.Proteins* 1996, *25:*354-365.

[25] Koch I, Lengauer T, Wanke E: *An algorithm for finding maximal common subtopolgies in a set of protein structures.Journal of Computational Biology* 1996, *3-2:*289-306.

[26] Mizguchi K, Go N: *Comparison of spatial arrangements of secondary structural elements in proteins.Protein Eng* 1995, *8:*353-362.

[27] Rufino SD, Blundell TL: *Structure-based identification and clustering of protein families and superfamilies.Journal of Comput Aided Mol Dec* 1994, *233:*123-138.

[28] Camoglu O, Kahveci T, Singh AK: *PSI: Indexing protein structures for fast similarity search.Proceedings of Elventh International Conference on Intelligent Systems for Molecular Biology* 2003, 81-83.

[29] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z: *Gapped blast and psi-blast: a new generation of protein database.Nucleic Acids Research* 1997, *17:*3389-3402.

[30] Chi PH, Scott G, Shyu CR: *A Fast Protein Structure Retrieval System Using Image-Based Distance Matrices and Multidimensional Index.Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering* 2004, 522-532.

[31] Lo WC, Huang PJ, Chang CH, Lyu PC: *Protein structrual similarity search by ramachandran codes.BMC Bioinformatics* 2007, *8*(307)*:*1-14.

[32] Altschul S, WMEM , Gish W, Lipman D: *Basic local alignment search tool.Journal of Molecular Biology* 1990, *3:*403-410.

[33] Yang JM, Tung CH: *Protein structure database search and evolutionary classification.Nucleic Acids Research* 2006, *1:*3646-3659.

[34] Lu Z, Zhao Z, Garcia S, Fu B: *New Algorithm and Web Server for Finding Proteins with Similar 3D Structures.Proceedings of the International Conference on Bioinformatics and Computational Biology (BIOCOMP'08)* 2008.

[35] Aung Z, Tan KL: *Rapid 3D protein structure database searching using information retrieval techniques.Bioinformatics* 2004, *20*(7)*:*1045-1052.

[36] Martin ACR: *The Ups and Downs of Protein Topology; Rapid Comparison of Protein Structure.Protein Engineering* 2000, *13:*829-837.

[37] Holm L, Kaariainen S, Rosenstrom P, Schenkel A: *Protein structure database searching by Dalilite v. 3.Bioinformatics* 2008, *24*(23)*:*2780-2781.

[38] S Subbiah DL, Levitt M: *Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core.Current Biology* 1993, *3:*141-148.

[39] Murzin AG, Brenner SE, Hubbard T, Chothia C: *SCOP: a structural classification of proteins database for the investigation of sequences and structures.J Mol Biol* 1995, *247:*536-540.

[40] Conte LL, Brenner SE, Hubbard T, Chothia C, Murzin A: *SCOP database in 2002: refinements accommodate structural genomics.Nucl Acid Res* 2002, *30:*264-267.

[41] Andreeva A, Howorth D, Brenner SE, Hubbard TJP, Chothia C, Murzin AG: *SCOP database in 2004: refinements integrate structure and sequence family data.Nucl Acid Res* 2004, *32:*D226-D229.

[42] Andreeva A, Howorth D, Chandonia JM, Brenner SE, Hubbard TJP, Chothia C, Murzin AG: *Data growth and its impact on the SCOP database: new developments.Nucl Acid Res* 2008, *36:*D419-D425.

[43] Eggert D, A Lorusso RF: *A comparison of four algorithms for estimating 3-d rigid transformations.British Machine Vision Conference* 1995, 237-246.

[44] Umeyama S: *Least-squares estimation of transformation parameters between two point patterns.IEEE Transactions on Pattern Analysis and Machine Intelligence* 1991, *13*(4):376-380.

[45] Lu Z, Zhao Z, Fu B [http://fpsa.cs.panam.edu/]

[46] Zaixin Lu, Zhiyu Zhao, Sergio Garcia, Krishnakumar Krishnaswamy, and Bin Fu: *Search Similar Protein Structures with Classfication, Sequence and 3-D Alignments. Journal of Bioinformatics and Computational Biology* 2009, 7(5):755-71.

[47] Zaixin Lu, Zhiyu Zhao and Bin Fu: *Efficient Protein Alignment Algorithm for Protein Search. BMC Bioinformatics* 2010, 11(Suppl 1):S34.

[48] Xu J, Li M, Kim D, Xu Y: *Optimal Protein Threading by Linear Programming.Journal of Bioinformatics and Computational Biology* 2003, 1(1):95-117.

[49] Sali, A. and T.L. Blundell: *Comparative protein modelling by satisfaction of spatial restraints. J Mol Biol* 1993, 234(3):779-815.

[50] Fiser, A., R.K.G. Do, and A. Sali: *Modeling of loops in protein structures. Protein Science* 2000, 9(9):1753-1773.

[51] Bowie, J.U., R. Luthy, and D. Eisenberg: *A method to identify protein sequences that fold into a known three-dimensional structure. Science* 1991, 253(5016):164-70.

[52] Wu, S. and Y. Zhang, MUSTER: *Improving protein sequence profile-profile alignments by using multiple sources of structure information. Proteins* 2008, 72(2):547-56.

[53] Jones, D.T., W.R. Taylor, and J.M. Thornton: *A New Approach to Protein Fold Recognition. Nature* 1992, 358(6381):86-89.

[54] Xu, Y. and D. Xu: *Protein threading using PROSPECT: design and evaluation. Proteins* 2000, 40(3):343-54.

[55] Skolnick, J., D. Kihara, and Y. Zhang: *Development and large scale benchmark testing of the PROSPECTOR_3 threading algorithm. Proteins* 2004, 56(3):502-18.

[56] Wu, S. and Y. Zhang: *LOMETS: a local meta-threading-server for protein structure prediction. Nucleic Acids Res* 2007, 35(10):3375-3382.

[57] Bradley, P., K.M.S. Misura, and D. Baker: *Toward high-resolution de novo structure prediction for small proteins. Science* 2005, 309(5742):1868-1871.

[58] Simons, K.T., C. Strauss, and D. Baker: *Prospects for ab initio protein structural genomics. Journal of Molecular Biology* 2001, 306(5):1191-1199.

[59] Kihara, D., et al.: *TOUCHSTONE: an ab initio protein structure prediction method that uses threading-based tertiary restraints. Proc Natl Acad Sci USA* 2001, 98(18):10125-30.

BIOGRAPHICAL SKETCH

ZAIXIN LU got his B.E degree in 2006 from the computer science department of Yanshan University, China. He got his M.S degree from the computer science department of the University of Texas-Pan American. His permanent mailing address is 18-3, 4-2-2, Nujiang North St, Shenyang City, Liaoning Province, China 110000.

He was doing research about algorithms and bioinformatics under Dr. Bin Fu's direction during his master; He has developed several practical algorithms and software for protein structure comparison, protein structure prediction and protein similarity search. Three of his papers have been published in this field, which are available at http://fpsa.cs.panam.edu/.