CLIQUE BASED ALGORITHMS FOR PROTEIN THREADING WITH PROFILES AND CONSTRAINTS

DUKKA BAHADUR K.C.

Graduate School of Information Science & Bioinformatics Center
Institute for Chemical Research, Kyoto University
Uji-city, Kyoto 611-0011, Japan
E-mail: dukka@kuicr.kyoto-u.ac.jp

ETSUJI TOMITA JUN'ICHI SUZUKI

Graduate School of Electro-Communications
The University of Electro-Communications
Chofu-city, Tokyo 182-8585, Japan
E-mail: {tomita,jsuzuki}@ice.uec.ac.jp

KATSUHISA HORIMOTO

Human Genome Center, Institute of Medical Science
The University of Tokyo
Minato-ku, Tokyo 108-8639, Japan
E-mail: khorimot@ims.u-tokyo.ac.jp

TATSUYA AKUTSU

Bioinformatics Center, Institute for Chemical Research
Kyoto University, Uji-city, Kyoto 611-0011, Japan
E-mail: takutsu@kuicr.kyoto-u.ac.jp

Protein threading with profiles in which constraints on distances between residues are given is known to be NP-hard. Moreover, a simple algorithm known as CLIQUETHREAD based on efficient reduction to maximum edge-weight clique finding problem has been known to be a practical algorithm for solving the protein threading problem with profiles and constraints. This algorithm is not efficient enough to be applicable to large scale threading prediction. Besides, the algorithm was only presented for profile threading with strict constraints. This paper presents a more efficient algorithm FTHREAD for profile threading with strict constraints which is more than 18 times faster than CLIQUETHREAD for larger proteins. Moreover, we also present a novel practical algorithm NTHREAD for profile threading with non-strict constraints. The comparison of FTHREAD with existing state-of-the-art methods shows that although our algorithm uses a simple threading function, our algorithm performs equally well as these existing methods for protein threading. Besides, our computational experiments for sequence-structure alignments for a number of proteins have shown better results for non-strict constraints threading than protein threading with strict constraints. We have also analyzed the effects of using a number of distance constraints.

1. Introduction

The computational prediction of protein structure from the sequence of amino acids is one of the most important task in the field of computational biology. In a situation like this, there are three possible approaches for the computational prediction depending upon the amino acid sequence of the newly generated protein. If the new protein is found to have high homology with a protein whose 3D structure is already known, methods based on homology modeling are very useful. In the second case, when the new protein is found to have weak sequence homology with proteins of known structure, protein threading is utilized and thirdly, when the new protein does not show any sequence similarity to the proteins previously known, ab-initio prediction is applied. It has been shown that it is possible to detect a weak homologous protein with known structure for a large percentage of proteins in a newly sequenced genome. In this regard protein threading is one of the important approaches for computational prediction of structure of a newly sequenced protein.[14]

According to Mirny and Shakhnovich,[18] there are two major factors affecting the accuracy of the threading alignment in the structure prediction by *threading*: (i) the degree of similarity between the template structure and the native structure (ii) the accuracy of the potential. One of the possible ways to overcome this fundamental problem in threading is to use some extra information about the query sequence or template structure. Hence, it is required to exploit more biological knowledge of the template or query sequence. This extra information gives rise to constraints on the alignments. In this regard, Young *et al.*[26] have developed a novel method which uses the Lys-Lys cross links determined using chemical cross-linking and time-of-flight and have shown how these cross-links can be used to identify the fold of a protein and to aid in the construction of homology modeling.

Moreover, Xu *et al.*[24] reported a method for the improvements of threading methods by incorporating partial NMR data. Also, Albrecht *et al.*[4] reported that using experimental distance constraints, an improvement in the fold recognition of protein threading can be achieved. Threading methods using additional information obtained from experimental data like distance between atoms of protein residues as measured by mass spectrometry or by NOE (Nuclear Overhauser effect) restraints of NMR spectroscopy have shown improvements in the efficiency of the folding algorithm. On the other hand, development of PSI-BLAST has significantly enhanced our ability to detect remote homologues and this in turn has helped to improve the efficiency of the protein structure prediction methods.

In this regard, we had also reported a mathematical analysis of protein threading with profiles and constraints and presented practical algorithms for protein threading with profiles and constraints.[3] We had shown that the protein threading problem with profiles and constraints is NP-hard and we had defined three types of protein threading problems on the basis of constraints viz. *Profile threading without constraints*, *Profile Threading with Strict Constraints* and *Profile Threading with Non-strict Constraints*. Using the notion of maximum edge-weight clique and dynamic programming, we had presented two algorithms called CLIQUETHREAD and BBDPTHREAD respectively for protein threading with strict constraints. However, the clique based algorithm CLIQUETHREAD reported was not very efficient especially in the case of larger proteins. For a protein pair of around

200 amino acids, the method took about an hour.

Besides, in our previous work we only used the simulated distance constraints between Lys-Lys atoms. So, it is a natural second step to try to explore the efficiency of the algorithm when several other distance constraints are used. Xu et al.[24] have shown that the more the average number of NOEs used per residue, the better is the accuracy of the prediction. In this context also, one can anticipate that by using a larger number of distance constraints, the efficiency of the methods could be enhanced. However, by increasing the number of distance constraints, it is natural that there arise cases where all the constraints are not satisfied. In this scenario, the profile threading with strict constraints fail to produce efficient results as this method tries to give the best solution provided that all the constraints are satisfied. In the cases like this, profile threading with non-strict constraints comes into play. Moreover, there can be cases where threading with non-strict constraints is a feasible solution for threading. However, no algorithm was reported in our previous work for protein threading with non-strict constraints. In this regard, we have developed a more efficient algorithm FTHREAD based on maximum edge-weight clique algorithm incorporating some heuristics to achieve significant improvement in the efficiency of the protein threading algorithm with strict constraints so that the algorithm is suitable for large scale protein threading prediction. Moreover, we have also developed a practical algorithm NTHREAD for protein threading with non-strict constraints, i.e. protein threading which outputs the threading with the maximum score under the condition that the number of unsatisfied constraints is minimized.

We have also analyzed extensively the effect of distances and the type of amino acids used. To validate the efficiency of our algorithm, we have also compared our results with some of the best threading methods like COBLATH[20] and the method of Kolinski *et al.*(KRIS).[13] We have found that our algorithm FTHREAD performs equally well as these methods in terms of accuracy of alignments. There exists several related works like that of Xu *et al.*[24] and Albrecht *et al.*[4] However, our algorithms are much simpler and general than existing algorithms and can be easily modified. Besides, our approach has another merit: with a much faster clique finding algorithm, our method as a whole becomes a very efficient approach. The most important advantage of our method is that it is very general and thus can be applied to almost all types of profile-based threading algorithms. Moreover, in contrast to our previous paper which solely focused on protein threading with strict-constraints, this current paper focuses on protein threading with non-strict constraints and also on the modified algorithm which performs 18 times faster than our previous algorithm. In this way, our new FTHREAD algorithm has helped us to achieve significant improvement in the efficiency of the threading method in terms of computational time and the quality of the results.

The remainder of the paper is outlined as follows. In Sec. 2, we present the formulation of protein threading with profiles and constraints. Then, in Sec. 3, we present an efficient algorithm FTHREAD for threading with strict-constraints and another practical algorithm NTHREAD for threading with non-strict constraints. In Sec. 4, we compare the CPU times of CLIQUETHREAD , BBDPTHREAD and FTHREAD, explain the results

of NTHREAD, study the effects of combination of various parameters and then compare FTHREAD with existing methods such as COBLATH and KRIS. Finally in Sec. 5, we analyze the main contribution of this paper and some important future works.

2. Problem Formulation

This section presents a formulation of the threading problem with profiles and constraints. The profiles considered in the current study are the profiles obtained by running PSI-BLAST and the constraints are the distance constraints between two residues of the protein obtained from the PDB.[6] The basic idea of the threading with profiles and constraints is to find an alignment between a query sequence and a template structure that satisfies the constraints specified using the required profiles. For the distance constraints, it is required that the two residues related by the constraints should be aligned to the template positions with a certain tolerance.

Before we explain the algorithm, we briefly review the problem formulation as presented in our previous work.[3] Initially, the threading (without constraint) can be defined as follows. Let $s = s_1 s_2 \dots s_m$ be a query protein sequence, over an alphabet Σ, where $|\Sigma| = 20$ with s_i representing the i^{th} amino acid of the sequence $s$and $t = t_1 t_2 \dots t_n$ be a template protein structure and t_j the j^{th} amino acid in t. This t can be considered to be a sequence of C^{α} (or C^{β}) atoms of the protein. A *threading* between s and t is an alignment obtained by inserting *gap symbols* ('$-$') into or at either end of s and t such that the resulting sequences s' and t' are of same length l, where it is not allowed that both s'_i and t'_j are gap symbols.

The profile PF_t for each template structure t is defined as a function from $(\Sigma \cup \{-\}) \times \{t_1, \dots, t_n, -\}$ to the set of real numbers R. Moreover, the *score of a threading* (s', t') is defined by $\sum_{i=1}^{l} PF_t(s'_i, t'_i)$.

The constraints in this formulation are defined as follows. If s_i and t_j are aligned in the same column in a threading (s', t'), it is denoted by $\psi(s_i) = t_j$. If s_i is aligned with the gap symbol, it is denoted as $\psi(s_i) =$'$-$'.

With all these definitions in hand, we define three types of protein threading problems.

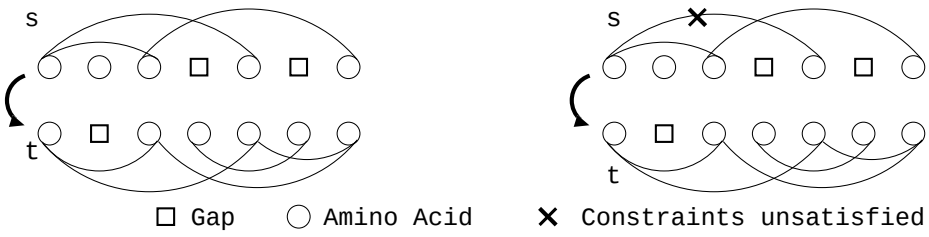

Figure 1. Threading with strict Constraints(left) and Threading with non-strict constraints(right).

Problem 1 (Profile Threading without Constraint). Given s, t and PF_t, find a threading (s', t') with the maximum score.

For a target sequence s, an *arc set* A_s is associated, which is a set of pairs of positions of s and each pair $(s_i, s_{i'}) \in A_s$ must satisfy $1 \leq i < i' \leq m$. Similarly, A_t denotes an arc set for a template structure t. In this paper, s_i appearing in A_s must not be aligned with a gap symbol at the same column. For each pairs $(s_i, s_{i'})$ and $(t_j, t_{j'})$, $IC(s_i, s_{i'}, t_j, t_{j'}) = 0$ if these pairs satisfy a constraint on $(s_i, s_{i'})$ (where a concrete definition of a constraint is to be given later and IC means *inconsistency*). If $(s_i, s_{i'}) \notin A_s$, $IC(s_i, s_{i'}, t_j, t_{j'}) = 0$. Otherwise (i.e., the pairs do not satisfy a constraint, or $(s_i, s_{i'}) \in A_s$ but $(t_j, t_{j'}) \notin A_t$), $IC(s_i, s_{i'}, t_j, t_{j'}) = 1$.

Problem 2 (Profile Threading with Strict Constraints). Given (s, A_s), (t, A_t), PF_t, and IC, find a threading (s', t') with the maximum score under the condition that $IC(s_i, s_{i'}, \psi(s_i), \psi(s_{i'})) = 0$ for all $(s_i, s_{i'}) \in A_s$.

Problem 3 (Profile Threading with Non-strict Constraints). Given (s, A_s), (t, A_t), PF_t, and IC, find a threading (s', t') with the maximum score under the condition that $\sum_{(s_i, s_{i'}) \in A_s} IC(s_i, s_{i'}, \psi(s_i), \psi(s_{i'}))$ is the minimum.

It is note-worthy that all the constraints must be satisfied in **Problem 2** whereas in **Problem 3** it is required to minimize the number of unsatisfied constraints.

In our previous work,[3] we presented a practical algorithm CLIQUETHREAD based on maximum clique finding algorithm for **Problem 2**. Although, the presented algorithm could solve most of the instances of protein threading, this algorithm is still unsuitable for the large scale threading calculation due to the fact that for larger proteins the time required is quite enormous. Moreover, no algorithm for threading with non-strict constraints (**Problem 3**) was presented in our previous paper. In this context, we have also developed an algorithm for **Problem 3**.

For our application, constraints should be defined as follows: $IC(s_i, s_{i'}, t_j, t_{j'}) = 0$ if $|dist(s_i, s_{i'}) - dist(t_j, t_{j'})|$ is less than a threshold Θ (a distance tolerance parameter) where $dist(s_i, s_{i'})$ (resp. $dist(t_j, t_{j'})$) denotes the distance between positions of C^{α} (or C^{β}) atoms associated with s_i and $s_{i'}$ (resp. t_j and $t_{j'}$).

3. Algorithms

3.1. *FTHREAD: An efficient algorithm for threading with strict constraints*

As already stated in Sec. 1, although our previous algorithm CLIQUETHREAD is able to solve the constrained threading problem, for larger proteins the computational time is extremely high.

In this regard, we have developed some heuristics which reduces the computational time significantly. We call this newer version of the algorithm as FTHREAD. This algorithm works by reducing the strict-constrained threading problem to the maximum edge weight clique finding problem, in which the total weight for edges in the clique is maximized under the condition that the number of vertices of the clique is maximum.

We construct an instance $G(V, E)$ of the clique problem in the following way. Let

$s_{i_1}, s_{i_2}, \ldots, s_{i_H}$ be residues in s appearing in A_s, where $i_1 < i_2 < \ldots < i_H$.

Here, v_0 and v_e are the starting node and terminal node added to the graph. We construct an undirected graph $G(V, E)$ defined by

$$\begin{aligned} V &= \{(s_{i_h}, t_j) | 1 \leq h \leq H, 1 \leq j \leq n\} \cup \{v_0, v_e\}, \\ E &= \{\ \{(s_{i_h}, t_j), (s_{i_{h'}}, t_{j'})\} \mid\ 1 \leq h < h' \leq H, \\ &\qquad\qquad 1 \leq j < j' < n\ \} \cup \\ &\quad \{\{v_0, (s_{i_h}, t_j)\} \mid\ 1 \leq h \leq H, 1 \leq j \leq n\ \} \cup \\ &\quad \{\{(s_{i_h}, t_j), v_e\} \mid\ 1 \leq h \leq H, 1 \leq j \leq n\ \}. \end{aligned}$$

For substrings s'', t'' of s, t, let us consider that $score(s'', t'')$ denotes the score of an optimal threading without constraints (i.e., an optimal solution for **Problem 1**) between s'' and t''. Then, the weight of each edge can be defined by

$$\begin{aligned} &w(\{v_0, (s_{i_1}, t_j)\}) = \\ &\qquad score(s_1 s_2 \ldots s_{i_1 - 1}, t_1 t_2 \ldots t_{j-1}) \ +\ \alpha, \\ &w(\{(s_{i_H}, t_j), v_e\}) = \quad PF_t(s_{i_H}, t_j) + \\ &\qquad score(s_{i_H + 1} \ldots s_m, t_{j+1} \ldots t_n) \ +\ \alpha, \\ &w(\{(s_{i_h}, t_j), (s_{i_{h+1}}, t_{j'})\}) = \\ &\begin{cases} 0 \quad \text{if } IC(s_{i_h}, s_{i_{h+1}}, t_j, t_{j'}) = 1, \\ 0 \quad \text{if } IC(s_{i,h}, s_{i_{h+1}}, t_j, t_{j'}) = 0 \ and \\ \qquad score(s_{i_h + 1} \ldots s_{i_{h+1} - 1}, t_{j+1} \ldots t_{j'-1}) < \gamma \\ score(s_{i_h + 1} \ldots s_{i_{h+1} - 1}, t_{j+1} \ldots t_{j'-1}) \\ \quad + PF_t(s_{i_h}, t_j) + \alpha \quad \text{otherwise}, \end{cases} \\ &w(\{(s_{i_h}, t_j), (s_{i_{h'}}, t_{j'})\}) = \\ &\qquad \begin{cases} 0 \quad \text{if } IC(s_{i_h}, s_{i_{h+1}}, t_j, t_{j'}) = 1, \\ \alpha \quad \text{otherwise}, \end{cases} \end{aligned}$$

where α and γ are constants. The edges with weight 0 are removed from the edge set E. The introduction of the cut-off parameter γ is the core part of the heuristics.

3.2. *NTHREAD: Algorithm for non-strict Constraints*

In contrast to **Problem 2** where it is required that all the constraints be satisfied, in **Problem 3** it is required to find an optimal threading which tries to minimize the number of unsatisfied constraints.

When using a number of distance constraints and changing the value of distance tolerance parameter and position tolerance parameter, there arises cases when some constraints remain unsatified. In such a case, it is required to calculate the optimal threading by minimizing the number of unsatisfied constraints.

Profile threading with non-strict constraints can also be solved by reducing the problem to the maximum edge weight clique finding problem, in which the total weight of the clique is maximized under the condition that the number of vertices of the clique is maximum.

Let us call the algorithm for profile threading with non-strict constraints as NTHREAD. In NTHREAD, the instance $G(V, E)$ of the clique problem is constructed in the same way as in profile threading with strict constraints.

In assigning the weights to the edges, in contrast to the algorithm for strict constraints, the edges have to be weighted even if the constraints are not satisfied i.e. even if $IC(s_i, s_{i'}, \psi(s_i), \psi(s_{i'})) = 1$.

Hence, the weight of each edge is defined as:

$$
\begin{aligned}
&w(\{v_0, (s_{i_1}, t_j)\}) = \\
&\qquad score(s_1 s_2 \ldots s_{i_1 - 1}, t_1 t_2 \ldots t_{j-1}) \ + \ \alpha, \\
&w(\{(s_{i_H}, t_j), v_e\}) = \quad PF_t(s_{i_H}, t_j) + \\
&\qquad score(s_{i_H + 1} \ldots s_m, t_{j+1} \ldots t_n) \ + \ \alpha, \\
&w(\{(s_{i_h}, t_j), (s_{i_{h+1}}, t_{j'})\}) = \\
&\qquad \begin{cases} score(s_{i_h + 1} \ldots s_{i_{h+1} - 1}, t_{j+1} \ldots t_{j'-1}) \\ \quad + PF_t(s_{i_h}, t_j) + \beta \\ \quad \text{if } IC(s_{i_h}, s_{i_{h+1}}, t_j, t_{j'}) = 1, \\ score(s_{i_h + 1} \ldots s_{i_{h+1} - 1}, t_{j+1} \ldots t_{j'-1}) \\ \quad + PF_t(s_{i_h}, t_j) + \alpha \quad \text{otherwise}, \end{cases} \\
&w(\{(s_{i_h}, t_j), (s_{i_{h'}}, t_{j'})\}) = \\
&\qquad \begin{cases} \beta \quad \text{if } IC(s_{i_h}, s_{i_{h+1}}, t_j, t_{j'}) = 1, \\ \alpha \quad \text{otherwise}, \end{cases}
\end{aligned}
$$

where α and β are constants.

In both of the algorithms, after the completion of assigning weights to the edges, a newer and more efficient version of the clique algorithm, WCQprime,[23] is utilized to the obtained graph to search for the maximum edge-weight clique.

3.3. *Efficient maximum clique finding algorithm: WCQprime*

One of the prominent advantages of the algorithm based on maximum edge-weight clique finding algorithm is that the better the clique finding algorithm becomes, the better the whole approach becomes. In this regard, in this present work we have utilized a newer version[23] of the maximum edge-weight clique finding algorithm developed by our co-authors(Suzuki & Tomita). This new algorithm is called WCQprime and this algorithm has been proved to be many times faster than the previous version of the WCQ algorithm[22,21] which is in turn much faster than the state-of-the-art clique finding algorithms that are based on the Bron & Kerbosch algorithm.[8]

The WCQprime algorithm is not described here and interested readers are requested to refer to the paper by Suzuki & Tomita.[23]

4. Computational Experiments

Each algorithm was executed using only one CPU of a PC cluster with Intel Xeon 2.8GHz CPUs under a Linux operating system using C language.

To obtain profiles to be used in the threading, PSI-BLAST[5] was used. The blastpgp command was run against the SWISSPROT database[7] using the global profile alignment algorithm with afine gap penalty (opening gap penalty = -11, gap extension penalty = -1).

In order to obtain constraints for target proteins, distances between C^{α} atoms of respective amino acids was calculated as in Young *et al.*[26] Then, amino acid pairs with the distances less than $24.0\mathring{A}$ were only taken into account as constraints based on the previous real experiments. While considering the distance constraints not only Lys-Lys pairs as in our previous paper but also aspartate, glutamate and arginine residues were taken into consideration as described in the respective computational experiments.

In addition, a position threshold cut-off P was defined. If two respective pairs of amino acid are placed within P residues in a target sequence, one of these two residues was not taken into account for generating constraints because such a pair provides little information on 3D structure. The more the value of P , the less the number of constraints. Hence, lesser the value of P, the more the number of constraints taken into consideration.

Similarly, as described in Sec. 2 a distance tolerance parameter Θ was defined as the maximum tolerable difference in the distance from the given distance value. Hence, the value of Θ decreases as the number of unsatisfied constraints increases.

4.1. *Comparison with CLIQUETHREAD*

We performed computational experiments of the newly developed FTHREAD with CLIQUETHREAD and BBDPTHREAD[3] algorithms and compared the CPU times of these algorithms for the following nine pairs of proteins. For this comparison, the value of the distance tolerance Θ was taken to be 4, the position tolerance was taken to be 6 and only the Lys-Lys pairs with the distances less than $24\mathring{A}$ were taken into account as constraints as in Akutsu *et al.*[3] For FTHREAD the value of γ was chosen to be -50. The results are summarized in Table 1. It is to be noted here that the time comparison of the three methods is for the computation which produced the same results in all of the three cases.

NA in the table shows that the computation did not terminate even after 10 hours. Particularly, in the case of protein pair (1xyz/8tim), it can be observed that the CPU time is significantly reduced from 3279 seconds to 178 seconds. Moreover, for the pair (1atn/1atr) the computational time is reduced to 4.48 hours. It can be observed from the experimental results that we have achieved significant gain in the efficiency of the clique based algorithm for profile threading with strict constraints. Although, the results of FTHREAD are still not as good as BBDPTHREAD for larger proteins, FTHREAD has many advantages over BBDPTHREAD. Some of them are: 1) FTHREAD is based on a very simple algorithm whereas BBDP is a very complicated and is not easy to modify; ii) BBDP cannot solve **Problem 3** where as slight modification of FTHREAD can solve it; iii) the time required in the case of BBDPTHREAD is not consistent; iv) and finally, for smaller proteins,

Table 1. Comparison of CPU times (sec.) of CLIQUETHREAD , BBDPTHREAD and FTHREAD.

Target	#res	Template	#res	CLIQUETHREAD	BBDP	FTHREAD
1bbn	133	1cnt1	150	1.5	8.3	0.41
1vltA	142	1nfn	132	0.27	0.11	0.05
3sdhA	145	1dlw	116	2.6	9.5	0.36
1ten	89	1ac6A	110	0.24	0.09	0.05
1bla	155	1hce	118	1.1	7.0	0.36
1a3k	137	1f5f	172	1.5	2.2	0.79
1bow	144	1d5yA2	173	0.57	0.24	0.05
1xyzA	320	8timA	247	3279	59.9	178
1atnA	372	1atr	383	NA	1101	16132

FTHREAD performs better than BBDPTHREAD.

4.2. *Experiments with non-strict Constraints*

As already explained in Sec. 1, it is a natural second step to try to explore the efficiency of the algorithm by using more distance constraints. To show the usefulness of the algorithm with non-strict constraints, we did some computational experiments. In order to increase the number of unsatisfied constraints, the conditions were made stricter which resulted in some unsatisfied constraints for each pair of proteins presented. Initially, the threading was computed with the FTHREAD algorithm which computes threading under strict constraints and then for the same protein pairs, the non-strict version of the algorithm NTHREAD was utilized. The results of each algorithm for each pair of proteins are given with the number of unsatisfied constraints, the number of aligned residues and the corresponding RMSDs are shown in Table 2.

For these experiments not only Lys-Lys pairs but also Glu-Glu pairs were taken into account as constraints.The value of Θ is taken to be 0.5, the value of β for NTHREAD is taken to be 1, the value of P is taken to be 4 and the value of γ for FTHREAD is taken to be -50.

Table 2. Comparison of results for strict constrained and non-strict constrained algorithm

Query	Template	#Unsatisfied	FTHREAD	#Unsatisfied	NTHREAD
1fxi	1ubq	5	69/11.17	1	62/10.47
1hip	2hip	11	69/4.05	3	65/3.88
2sar	9rnt	6	76/12.89	1	70/10.09
5fd1	2fxb	9	67/7.9	2	67/6.9
1isu	2hip	0	60/3.60	0	60/3.60

It can be seen that using the non-strict version of the algorithm, improvement in the number of aligned residues and RMSDs can be obtained. It is also noteworthy that the strict version of the algorithm and non-strict version of the algorithm produce similar results if there are no unsatisfied constraints.

4.3. *Threading accuracy VS number of constraints*

In order to know the relationship of threading accuracy to the number of constraints and different distance parameters, we also performed some computational experiments for FTHREAD. The distances between either of the lysine residues, arginine residues, aspartate residues, glutamate residues or the combination of them is considered as shown in Table 3. The distance tolerance parameter is varied so as to know the effects of changing distance tolerance. The position tolerance parameter is not varied keeping in mind that varying the position tolerance parameter P directly results in the increasing of unsatisfied constraints.

Table 3. Effects of number of constraints and distance tolerance on threading accuracy. KEDR represents the respective amino acids Lysine, Glutamic acid, Asparatic acid and Arginine. Thus KEDR represents that all the four amino acids are used and KED represents that the amino acids Lysine, Glutamic acid and Asparatic acid are used and so on. The value x/y represents that x is the number of aligned residues and y is the RMSD

Pair	Amino Acids	P=6 Θ= 4	P = 6 Θ = 3.5	P = 6 Θ = 3.0	P = 6 Θ = 2.5
1cauB 1cauA	KEDR	167/5.61	167/5.61	166/5.53	165/5.55
	KED	167/5.61	167/5.61	167/5.61	163/6.15
	KDR	167/5.61	167/5.61	166/5.53	165/5.55
	KE	167/5.61	167/5.61	166/3.94	165/5.55
	KD	165/3.94	165/3.94	163/3.94	164/3.94
	K	165/3.94	165/3.94	165/3.92	164/3.94
1isuA 2hipA	KEDR	60/3.45	60/3.43	60/3.43	60/3.43
	KED	60/3.46	60/3.43	60/3.43	60/3.34
	KDR	60/3.46	60/3.43	60/3.39	60/3.39
	KE	60/3.46	60/3.39	60/3.39	60/3.39
	KD	60/3.46	60/3.43	60/3.43	60/3.43
	K	60/3.46	60/3.39	60/3.39	60/3.39
1mup 1rbp	KEDR	131/7.92	119/8.23	144/7.02	127/12.22
	KED	148/5.62	143/6.23	140/8.34	136/6.36
	KDR	143/7.32	144/7.08	143/7.04	139/7.23
	KE	154/8.93	147/7.12	147/7.12	148/7.09
	KD	140.6.83	136/7.01	122/7.19	118/8.55
	K	150/7.64	147/7.73	144/7.59	133/9.30

From the observations of Table 3, the following general conclusions can be derived. Increasing the number of amino acids in generating constraints increases the efficiency of the method in general but at the same time results in the increase of the number of unsatisfied constraints such that there is a trade-off between the number of amino acids that has to be considered. From our experiments, it can be seen that considering Lysine and Glutamic Acid produces better results than other combinations. Similarly, it can also be observed that decreasing the distance tolerance parameter increases the efficiency of the method in general, but if the distance tolerance parameter is decreased below a certain value then again the number of unsatisfied constraints increases, resulting in the loss of efficiency. Although, the results for the various parameter settings shown in Table 3 shows similar results, the combination of Lysine and Glutamic acid with $P = 6$ and $\Theta = 3$ produces slightly better results. Hence, for the comparison of our method FTHREAD with other

existing methods, we use this set of parameters.

4.4. *Comparison with other methods*

For the efficiency of our FTHREAD algorithm, we compared our methods with the methods of Kolinski *et al.*[13](KRIS) and Shan *et al.*[20] (COBLATH). The method of Kolinski uses a high-coordination lattice approximation of the query protein fold and monte Carlo simulated annealing to improve the alignment accuracy of threading. Similarly,the method of Shan *et al.*(COBLATH) utilizes PSI-blast and a sophisticated scoring function for threading. These methods are compared using a 12 query template pair first utilized by Kolinski *et al.*[13] The RMSDs of the alignments by KRIS, COBLATH and our method are compared in Table 4.

The value of Θ is chosen to be 3, the value of the position tolerance parameter P is set to be 6, and the Lysine residues and Glutamic acid residues were taken into consideration while generating constraints as obtained from the analysis of different sets of parameters.

Table 4. Comparison with KRIS and COBLATH

Query	Template	KRIS	COBLATH	FTHREAD
1aba	1ego	4.86	3.38	3.20
1bbhA	2ccyA	6.82	3.51	2.97
1cewI	1molA	14.38	13.29	9.08
1hom	1lfb	3.70	4.80	5.73
1stfI	1molA	5.95	12.98	8.53
1tlk	2rhe	4.17	5.04	7.52
256bA	1bbh	4.26	3.92	6.18
2azaA	1paz	10.77	3.82	5.04
2pcy	2azaA	4.41	5.65	5.58
2sarA	9rnt	7.83	3.80	7.52
3cd4	2rhe	6.39	8.50	9.05
5fd1	2fxb	12.40	9.61	8.37

It can be seen that although our method utilizes only a simple threading algorithm, our method produced lower RMSDs for six proteins compared to COBLATH and higher RMSDs for six proteins than COBLATH. In comparison to KRIS, our method produced lower RMSDs for six proteins and higher RMSDs for six proteins. The algebraic mean of the 12 RMSDs is $7.2\mathring{A}$ for the method of KRIS, $6.4\mathring{A}$ for COBLATH and $6.56\mathring{A}$ for our method. Hence, it can be observed that although our method uses only a simple threading function, our method produces results similar to some of the sophisticated methods.

5. Conclusion and Discussion

The main contributions of this paper are the FTHREAD algorithm for threading with strict constratins and a practical NTHREAD algorithm for threading with non-strict constraints. In the case of FTHREAD, we were able to achieve a significant gain in the computational time for larger proteins than its predecessor, CLIQUETHREAD.

We also presented a novel algorithm NTHREAD for threading with non-strict constraints presented some results to show that this threading helps to attain a better prediction especially when there are a number of unsatisfied constraints. It can also be observed from the computational experiments that threading with constraints produces better results than the threading with no constraints. Moreover, in most of the cases the lesser the number of unsatisfied constraints, the better is the RMSD of the predicted structure.

Moreover, adding more constraints also results in the increase of number of unsatisfied constraints. In this scenario, the NTHREAD algorithm developed for non-strict constraints is much more useful than the FTHREAD algorithm developed for strict constraints. However, the current version of NTHREAD algorithm is not as fast as the FTHREAD algorithm. Hence, one of the major future works is to work on the improvement of efficiency of the NTHREAD algorithm.

About the practical usage of the NTHREAD algorithm for non-strict constraints, it seems to be very useful especially when there are a number of unsatisfied constraints. Especially, when a number of constraints like simulated distance constraints, NOE distances constraints or distances between disulfide bonds are used as constraints, it is likely that the number of unsatisfied constraints in this case will be higher. Hence, the algorithm for non-strict constraints can be expected to be very useful in such a scenario.

We showed that a small number of experimental distance constraints already suffice to improve the query sequence template structure alignment. Other additional constraints like disulfide bridges, NOE restraints could also be used to improve the accuracy of the prediction. Although, there exists similar methods like this, our method is simple and general and hence, can be applied to any type of profile-based threading algorithms and modified easily.

We have utilized profiles of template structures in order to improve the quality of threading algorithms. Sadreyev *et al.*[19] have utilized profile-profile alignment to protein threading. In this context, our method can also be applied to profile-profile alignment approach for protein threading. Hence, another important future work is to explore the possibilities of our approach for possible application in profile-profile alignment for protein threading.

Even though we did not perform any experiments for fold recognition, we expect to obtain better results in case of fold recognition also. The major difference between fold recognition and sequence-structure alignment is the size of the search space that is needed to be searched or the number of the alternatives to choose from. Fold recognition is aimed at finding a structure in a representative fold database which contains about some thousand folds whereas threading algorithm applied to prediction of threading two proteins tries to explore the search space that is much larger compared to the fold space. In this sense, it can be inferred that our algorithm might work well for the fold recognition problem as fold recognition problem is less demanding than the sequence-structure alignment problem.

6. Acknowledgments

This work was supported in part by a Grant-in-Aid for scientific Research on Priority Areas(C) for “Genome Information Science” from the Ministry of Education, Culture, Sports,

Science and Technology (MEXT) of Japan. D.B.K.C. is supported by the fellowship from International Communications Foundation(ICF). Part of the work by D.B.K.C was done while visiting structural Bioinformatics Lab at Boston University. We would also like to thank those responsible at the Institute for Chemical Research at Kyoto University and at the Human Genome Center at the University of Tokyo for providing us the resources for the computations.

References

1. T. Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Trans. on Information and Systems*, E79-D:1629–1636, 1996.
2. T. Akutsu and S. Miyano. On the approximation of protein threading. *Theoretical Computer Science*, 210:261-275, 1999. (also in Proc. RECOMB 1997, 3-8).
3. T. Akutsu, M. Hayashida, E. Tomita, J. Suzuki, and K. Horimoto. Protein threading with profiles and constraints. *Proceedings of the fourth IEEE symposium on Bioinformatics and Bioengineering(BIBE'04)*, 2004.
4. M. Albrecht, D. Hanisch, R. Zimmer, and T. Lengauer. Improving fold recognition of protein threading by experimental distance constraints. *In Silico Biology*, 2:0030, 2002.
5. S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.
6. H.M. Berman *et al.* The Protein Data Bank. *Nucleic Acids Research*, 28:235-242, 2000.
7. B. Boeckmann *et al.* The Swiss-Prot protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31:365-370, 2003.
8. C. Bron and J. Kerbosch Algorithm 457: Finding all cliques of an undirected graph. *Comm. ACM*, 16:575-577, 1973.
9. P.A. Evans. Finding common subsequences with arcs and pseudoknots. *Lecture Notes in Computer Science*, No.1645 (Proc. CPM'99), 270-280, 1999.
10. D. Goldman, S. Istrail, and C.H. Papadimitriou. Algorithmic aspects of protein structure similarity. *Proc. 40th IEEE Symp. on Foundations of Computer Science*, 512-522, 1999.
11. T. Jiang, G. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for with arc annotated sequences. *Lecture Notes in Computer Science*, No. 1848 (Proc. CPM 2000), 154-165, 2000.
12. T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9:371-388, 2002.
13. A. Kolinski, P. Rotkiewicz, B. Ilkowski, and J. Skolnick. A method for the improvement of threading-based protein models. *Proteins*, 37:592-610, 1999.
14. A. Kolinski, M.R. Betancourt, D. Kihara, P. Rotkiewicz, and J. Skolnick. Generalized Comparative Modeling (GENECOMP): A combination of sequence comparison, threading and lattice modeling for protein structure prediction and refinement. *Proteins: Structure, Function, and Genetics*, 44:133-149, 2001.
15. G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. *Proc. 5th Int. Conf. Computational Molecular Biology*, 193-202, 2001.
16. R.H. Lathrop and T.F. Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *Journal of Molecular Biology*, 255:641-665, 1996.
17. G. Lin, Z-Z. Chen, T. Jiang, and J. Wen. The longest common subsequence problem for sequences with nested arc annotations. *Journal of Computer and System Sciences*, 65:465-480, 2002.

18. L. Mirny and E.I. Shakhnovich. Protein structure prediction by threading. why it works and why it does not. *Journal of Molecular Biology*, 283:507-526, 1998.
19. RI Sadreyev, D. Baker and NV Grishin. Profile-profile comparison by COMPASS predict intricate homologies between protein families. *Protein Science*, 12(10): 2262-2272, 2003.
20. Y. Shan, G. Wang, and H. Zhou. Fold recognition and accurate query-template alignment by a combination of PSI-BLAST and threading. *PROTEINS: Structure, Function, and Genetics*, 42:23-37, 2001.
21. J. Suzuki, E. Tomita, and T. Seki. An algorithm for finding a maximum clique with maximum edge-weight and computational experiments. Technical Report MPS-42-12, 45-48, Information Processing Society of Japan, 2002.
22. E. Tomita and T. Seki. An efficient branch-and-bound algorithm for finding a maximum clique. *Lecture Notes in Computer Science*, No. 2731 (Proc. DMTCS 2003), 278-289, 2003.
23. J. Suzuki and E. Tomita. An efficient algorithm for finding a maximum clique with maximum edge-weight. Technical Report UEC-TR-CAS7, The University of Electro-Communications, 2004.
24. Y. Xu, D. Xu, O.H. Crawford, and J.R. Einstein. A computational method for NMR-constrained protein threading. *Journal of Computational Biology*, 7:449-467, 2000.
25. J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: Optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1:95-118, 2003.
26. M. M. Young *et al*. High throughput protein fold identification by using experimental constraints derived from intermolecular cross-links and mass spectrometry. *Proceedings of the National Academy of Sciences*, 97:5802-5806, 2000.