

On the Approximation of Protein Threading

Tatsuya Akutsu and Satoru Miyano

Human Genome Center, Institute of Medical Science, University of Tokyo

4-6-1 Shirokanedai, Minato-ku, Tokyo 108 Japan

e-mail: {takutsu, miyano}@ims.u-tokyo.ac.jp

Abstract

In this paper, we study the protein threading problem, which was proposed for finding a folded 3D protein structure from an amino acid sequence. Since this problem was already proved to be NP-hard by Lathrop, we study polynomial time approximation algorithms. First we show that the protein threading problem is MAX SNP-hard. Next we show that the protein threading problem can be approximated within a factor 4 for a special case in which a graph representing interaction between residues (amino acids) is planar. This case corresponds to a β -sheet substructure, which appears in most protein structures.

1 Introduction

The *protein folding* problem is, given an amino acid sequence (a string), to find its correctly folded 3D protein structure. It is one of the most important computational problems in molecular biology. Although this problem can be defined as a minimization problem, it is too hard to be solved directly.

Recently, an indirect approach called *inverse folding* was proposed [2, 4, 6, 10]. In inverse folding, given an amino acid sequence and a set of protein structures (structural patterns), a structure into which the sequence is most likely to fold is computed. To test whether or not a sequence is likely to fold into a structure, an *alignment* between spatial positions of a 3D structure and amino acids of a sequence is computed using a suitable *score function*. That is, an alignment which minimizes the total score (corresponding to potential energy) is computed. This minimization problem is called a *protein threading* problem, and an alignment between a sequence and a structure is called a *threading* (see Fig. 1) [4, 9, 10]. Note that, in Fig. 1, gaps (insertions and deletions of amino acids) are not allowed in *core regions*, but allowed only in *loop regions*, where a protein structure is partitioned into core regions and loop regions. This assumption is used in Ref. [10] and seems reasonable. Although we employ this assumption in this paper, our results may be extended for the case in which insertions and deletions of amino acids in core regions are allowed.

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.

RECOMB 97, Santa Fe New Mexico USA

©copyright 1997 ACM 0-89791-882-8/97/01 ...\$3.50

A variety of studies have been done for the protein threading problem. However, there are only a few studies that try to find an *optimal threading* (i.e., a threading with the minimum score) [4, 9, 10]. Bryant and Lawrence used exhaustive search to examine all possible threadings [4]. But, their method can only be applied to very small structures. Lathrop proved that the protein threading problem is NP-hard [9]. However, Lathrop and Smith applied the branch-and-bound search technique to the protein threading problem in a clever way and succeeded to compute optimal threadings for proteins of medium size [10].

Although Lathrop and Smith's algorithm is very nice, long time is required if it is applied to a large protein structure. Moreover, if we allow insertions and deletions of amino acids in core regions, it would take too long time. Thus, this paper studies a computational aspect of the protein threading problem. Since it was already proved to be NP-hard, we study (polynomial time) approximation algorithms. Of course, approximate solutions are different from optimal solutions corresponding to 3D shape. However, they are useful from the following reasons: the correct structure could be selected if the guaranteed error bound were sufficiently small; approximate solutions might lead to optimal solutions if it is combined with local search; approximate solutions might be useful for speeding up the branch-and-bound procedure. Note that an approximation algorithm has been already developed for the protein folding problem (not the inverse folding problem), but its model was too simple [7].

In this paper, we first show that the protein threading problem is MAX SNP-hard. Moreover, we show that approximation of the problem is at least as hard as approximation of the DENSE- k -SUBGRAPH problem, for which only an $O(n^{0.3885})$ ratio approximation algorithm is known [8]. Next we consider a special case in which a graph representing interactions between residues (amino acids) is planar. This case corresponds to most β -sheet substructures. For this case, we show a polynomial time algorithm which approximates the optimal score within a constant factor.

2 The Protein Threading Problem

As mentioned in Introduction, the protein threading problem (PROTEIN THREADING, in short) is, given a sequence and a 3D protein structure, to find an alignment (a threading) between the sequence and the structure with the minimum score. Lathrop and Smith defined this problem in a formal way [9, 10]. In this paper, we modify their definition into much simpler form without loss of generality as follows (see Fig. 1).

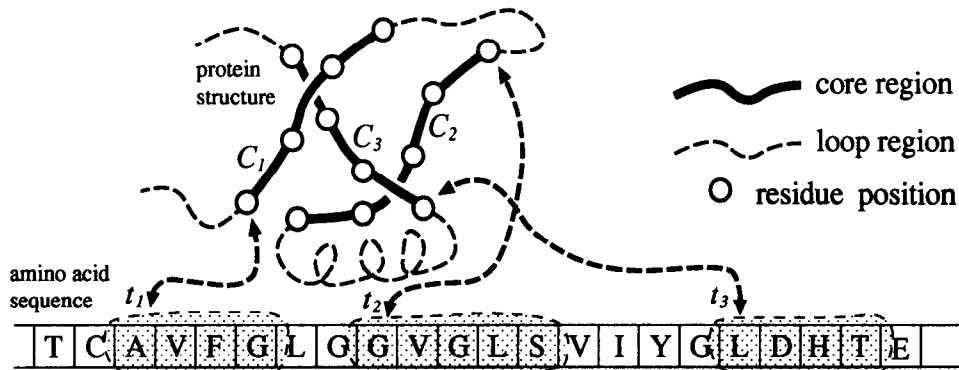


Figure 1: The protein threading problem.

Input: sequence $s = s_1 s_2 \dots s_n$ over a fixed alphabet Σ (usually $|\Sigma| = 20$),
 core lengths c_1, c_2, \dots, c_m ,
 score function $g(i, j, t_i, t_j)$ ($g(i, j, t_i, t_j) \geq 0$),

Output: m -tuple $t = (t_1, \dots, t_m)$ which maximizes a total score: $score(t) = \sum_{i < j} g(i, j, t_i, t_j)$
 under the condition that $1 \leq t_1$, $t_i + c_i \leq t_{i+1}$,
 $t_m + c_m \leq n + 1$.

This definition may look strange since it is defined as a maximization problem. However, the protein threading problem is intrinsically a maximization problem as well as the protein folding problem is. Indeed, usual score functions can take negative values and the minimum score can become negative. Thus, by inverting the sign of values of score functions and by adding a constant factor, we can treat the protein threading problem as a maximization problem. This definition seems more natural when we consider approximation algorithms for the problem.

In Lathrop and Smith's definition, two kinds of score functions $g_1(i, t_i)$ and $g_2(i, j, t_i, t_j)$ are used, while only one kind is used in the above definition. However, letting $g(i, i+1, t_i, t_{i+1}) = g_1(i, t_i) + g_2(i, i+1, t_i, t_{i+1})$, we can treat any score functions used by Lathrop and Smith. Note that we ignore the time for computing $g(i, j, t_i, t_j)$ because it can be computed in polynomial time for most score functions. Although the lengths of loop regions are included in an input in Lathrop and Smith's definition, the effect of loop regions can be taken into account in the above definition by adding a length of a loop region to a length of a core region and by modifying $g(i, j, t_i, t_j)$ suitably. Therefore, in the above definition, there is no loss of generality.

We call $t = (t_1, \dots, t_m)$ a *threading* if it satisfies the above condition ($1 \leq t_1$, $t_i + c_i \leq t_{i+1}$, $t_m + c_m \leq n + 1$). C_i denotes the i 'th core (with length c_i). For an instance PT of the problem, we associate a directed (multi) graph $G_{PT}(V_{PT}, E_{PT})$ such that $V_{PT} = \{C_1, \dots, C_m\}$, $E_{PT} = \overline{E_{PT}} \cup \{(C_i, C_{i+1}) \mid 1 \leq i < m\}$, where $\overline{E_{PT}} = \{(C_i, C_j) \mid i < j \text{ and } \exists (t_i, t_j)(g(i, j, t_i, t_j) \neq 0)\}$. Note that G_{PT} can have multi-edges, where the maximum multiplicity is at most 2.

In this paper, we consider two special cases: a case in which the maximum vertex degree of G_{PT} is bounded by a constant B , and a case in which G_{PT} is planar. The former case is called PROTEIN THREADING- B . Most protein structures correspond to this case because each core interacts with small number of other cores if we ignore weak

interactions. Indeed, most scores become very small when the distance between residues exceeds a threshold value [9].

The planar case corresponds to β -sheet substructure, which appears in most core regions and is known as a kind of *secondary structure* (see Fig. 2) [3]. β -sheet consists of multiple β -strands. To classify β -sheet structures, topology diagram has been used [3]. In topology diagram, β -strands are usually arranged parallel in a plane, and each β -strand strongly interacts with at most two neighbor β -strands. Thus, in most β -sheet substructures, G_{PT} can be considered a planar graph. Although other substructures (e.g., α -helices) can appear in core regions, good approximations might be obtained even if we only consider β -sheet substructures. At least, this special case would be useful to identify β -strands in a given amino acid sequence. Identifying β -strands is also an important problem in molecular biology.

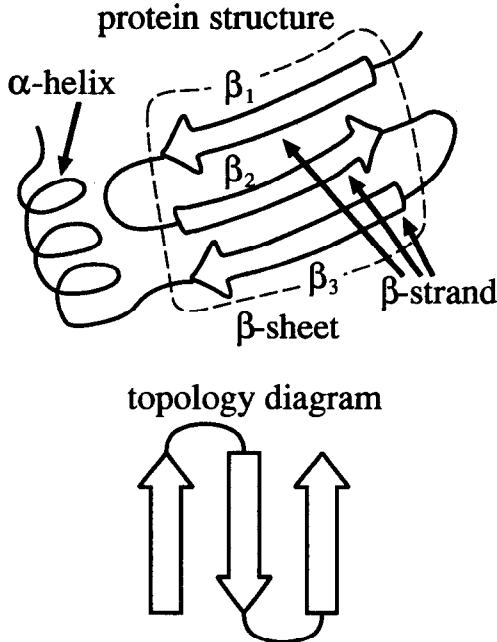


Figure 2: β -sheet and topology diagram. In this case, β_2 strongly interacts with β_1 and β_3 .

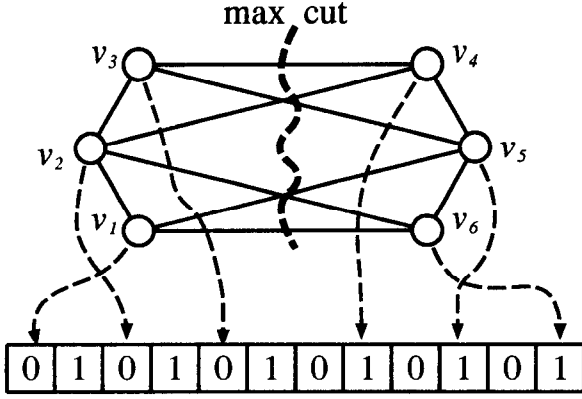


Figure 3: Correspondence between the maximum cut and the optimal threading.

In this paper, we consider polynomial time approximation algorithms. Recall that the *performance ratio* of an approximation algorithm for a maximization problem is the worst-case ratio of the size of the optimal solution to the size of the approximate solution. If there exists an approximation algorithm with performance ratio $f(n)$ for a problem X where n denotes the size of an input, we say that X can be approximated within a factor $f(n)$ [1].

3 Hardness Results

First, we show that PROTEIN THREADING is MAX SNP-hard even if the maximum vertex degree is bounded by a constant B . From this result, a constant size lower bound of performance ratio follows under the assumption of $P \neq NP$ [1, 13]. Note that the following theorem also gives a much simpler proof of NP-hardness of PROTEIN THREADING than that in [9].

Theorem 3.1 *PROTEIN THREADING-B is MAX SNP-hard.*

Proof. We use L -reduction from MAX CUT for graphs of bounded vertex degree $B - 2$, which was already proved to be MAX SNP-complete [13]. MAX CUT is, given an undirected graph $G(V, E)$, to find a subset $V' \subset V$ maximizing the cardinality of the cut (i.e., the number of edges between V' and $V - V'$). From $G(V, E)$ of bounded degree $B - 2$ where $V = \{v_1, \dots, v_n\}$, we construct an instance of PROTEIN THREADING-B in the following way:

$$s = \overbrace{0101 \dots 01}^{n-01's} \text{ over } \Sigma = \{0, 1\},$$

$$c_1 = c_2 = \dots = c_n = 1,$$

$$g(i, j, t_i, t_j) = 1 \text{ if } i < j, (v_i, v_j) \in E \text{ and } s_{t_i} \neq s_{t_j}, \text{ otherwise } g(i, j, t_i, t_j) = 0.$$

Then, each threading corresponds to a cut and the score of a threading is equal to the size of the corresponding cut (see Fig. 3). Therefore, this reduction is L -reduction and the theorem follows. \square

Next, we show a result suggesting that approximation of PROTEIN THREADING is much harder, using a reduction from the DENSE- k -SUBGRAPH problem [8]. DENSE- k -SUBGRAPH is, given an undirected graph $G(V, E)$ and an integer k , to find a subset $V' \subset V$ of cardinality k with

maximum number of edges (in an induced subgraph by V'). Although no lower bound of performance ratio has been proved, DENSE- k -SUBGRAPH is considered to be hard to approximate. Currently, an approximation algorithm by Kortsarz and Peleg achieves the best performance ratio of $O(n^{0.3885})$ [8].

Theorem 3.2 *If PROTEIN THREADING can be approximated within a factor $f(|s|)$ in polynomial time, DENSE- k -SUBGRAPH can be approximated within a factor $f(|V|^2)$ in polynomial time.*

Proof. From an instance $(G(V, E), k)$ of DENSE- k -SUBGRAPH where $V = \{v_1, \dots, v_n\}$, we construct an instance of PROTEIN THREADING in the following way.

$$s = \overbrace{00 \dots 0}^{n-1} a_1 \overbrace{00 \dots 0}^{n-1} a_2 \overbrace{00 \dots 0}^{n-1} \dots$$

$$\overbrace{00 \dots 0}^{n-1} a_k \overbrace{00 \dots 0}^{n^2 - kn} \text{ over } \Sigma = \{0, 1\}$$

where $a_i = 1$ for all i , and $c_j = 1$ for all j ,

$$g(i, j, t_i, t_j) = 1 \text{ if } i < j, (v_i, v_j) \in E \text{ and } s_{t_i} = s_{t_j} = 1, \text{ otherwise } g(i, j, t_i, t_j) = 0.$$

Then, a subset of cores $\{C_i \mid s_{t_i} = 1\}$ corresponds to a subset V' . From this observation, it is easy to see that the maximum score in PROTEIN THREADING is equal to the maximum number of edges in DENSE- k -SUBGRAPH. Moreover, given a threading t , we can obtain $V' \subset V$ of cardinality at most k having $score(t)$ edges in polynomial time. Since $|s|$ (length of s) is n^2 , the theorem holds. \square

Note that if the maximum degree of $G(V, E)$ is bounded by a constant, there is a trivial constant ratio approximation algorithm for DENSE- k -SUBGRAPH. In the above construction, an instance of PROTEIN THREADING-B is constructed from such a graph. Thus, the above theorem does not suggest the hardness of PROTEIN THREADING-B. However, we can show that approximating PROTEIN THREADING-B is at least as hard as approximating DENSE- k -SUBGRAPH for general graphs.

Theorem 3.3 *If PROTEIN THREADING-B can be approximated within a factor $f(|s|)$ in polynomial time, DENSE- k -SUBGRAPH can be approximated within a factor $f(|V|^3)$ in polynomial time.*

Proof. Modify the construction in Thm. 3.2 as follows, where we assume without loss of generality that $k < n - 1$:

$$s = \overbrace{00 \dots 0}^{n^2} b_1 \overbrace{00 \dots 0}^{n^2} b_2 \overbrace{00 \dots 0}^{n^2} \dots \overbrace{00 \dots 0}^{n^2} b_k$$

$$\overbrace{00 \dots 0}^{n^3 - kn^2 - kn} \text{ over } \Sigma = \{0, 1\}$$

where each b_i is a sequence of 1's with length n ,

$$c_1 = c_2 = \dots = c_{2n^2} = 1,$$

$$g(2n(i-1)+j, 2n(j-1)+i, t_{2n(i-1)+j}, t_{2n(j-1)+i}) = 1 \text{ if } 0 < i < j \leq n, (v_i, v_j) \in E \text{ and } s_{t_{2n(i-1)+j}} = s_{t_{2n(j-1)+i}} = 1,$$

$$\text{otherwise } g(i, j, t_i, t_j) = 0.$$

In this case, consecutive cores $C_{2n(i-1)+1} C_{2n(i-1)+2} \dots C_{2n(i-1)+n}$ correspond to a vertex v_i . Note that cores $C_{2n(i-1)+n+1} C_{2n(i-1)+n+2} \dots C_{2n(i-1)+2n}$ are introduced so that one b_i does not correspond to multiple vertices. Since

$G_{PT}(V_{PT}, E_{PT})$ is a graph of bounded degree 3, the theorem can be proved using a similar discussion as in Thm. 3.2. \square

Next, we consider the case in which $G_{PT}(V_{PT}, E_{PT})$ is planar. For this case, we will show a constant ratio approximation algorithm in the next section. But, even in this case, the problem remains NP-hard as shown below.

Theorem 3.4 *PROTEIN THREADING-B is NP-hard even if $G_{PT}(V_{PT}, E_{PT})$ is planar.*

Proof. We use a reduction from the longest common subsequence problem (LCS) over a binary alphabet, which was already shown to be NP-hard [11]. LCS is, given strings $s^1 = s_1^1 \dots s_{n_1}^1$, $s^2 = s_1^2 \dots s_{n_2}^2$, \dots , $s^k = s_1^k \dots s_{n_k}^k$ over $\Sigma' = \{0, 1\}$ and an integer L , to decide whether or not there exists a string s' of length L such that s' is a subsequence of s^i for each $i \leq k$. We assume without loss of generality that k is even. Let $(s^i)^{-1}$ denote the reversed string of s^i (i.e., $(s^i)^{-1} = s_{n_i}^i s_{n_i-1}^i \dots s_1^i$).

From an instance of LCS, we construct an instance of PROTEIN THREADING as follows.

$$s = b s^1 a a (s^2)^{-1} a a s^3 a a (s^4)^{-1} a a \dots s^{k-1} a a (s^k)^{-1} b$$

over $\Sigma = \{0, 1, a, b\}$,

$$c_1 = c_2 = \dots = c_{k(L+2)} = 1,$$

$$g(i, j, t_i, t_j) =$$

$$\begin{cases} 1, & \text{if } (1 \leq \exists p \leq k-1)(1 \leq \exists q \leq L)(i = (L+2)p - q \\ & \text{and } j = (L+2)p + 1 + q \text{ and } s_{t_i} = s_{t_j} \in \{0, 1\}), \\ \alpha, & \text{if } (1 \leq \exists p \leq k-1)(i = (L+2)p \text{ and} \\ & j = (L+2)p + 1 \text{ and } s_{t_i} = s_{t_j} = a), \\ 0, & \text{otherwise.} \end{cases}$$

Note that $G_{PT}(V_{PT}, E_{PT})$ becomes a planar graph in this case (see Fig. 4). We let $\alpha \gg kL$. Then, it is easy to see that there exists a common subsequence s' of length L if and only if there exists a threading t such that $\text{score}(t) \geq (k-1)(L + \alpha)$. \square

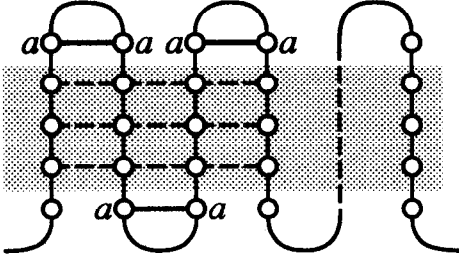


Figure 4: $G_{PT}(V_{PT}, E_{PT})$ appeared in the proof of Thm. 3.4. Shaded part corresponds to LCS.

4 An Approximation Algorithm for a Planar Case

In this section, we show a constant ratio approximation algorithm for PROTEIN THREADING in a special case in which an associated graph $G_{PT}(V_{PT}, E_{PT})$ is planar.

To develop an approximation algorithm, we partition a set of edges $\overline{E_{PT}}$ into three subsets (see Fig. 5): E_u (a set of upper edges), E_l (a set of lower edges), E_o (a set of loop edges). First we show that an optimal threading can be obtained using a simple dynamic programming algorithm if

$E_l = E_o = \emptyset$ or $E_u = E_o = \emptyset$. Next we show an approximation algorithm with performance ratio 2 for a case of $E_l = E_u = \emptyset$. Combining those, we obtain an approximation algorithm with performance ratio 4.

Although we only describe a method to compute scores of approximate threadings, it can be modified for computing such threadings.

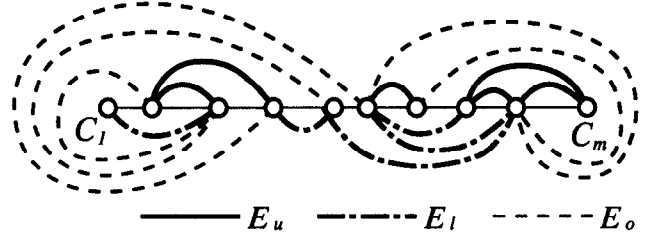


Figure 5: Partition of $\overline{E_{PT}}$ into E_u , E_l and E_o . In this figure, cores are arranged on a horizontal line where each core is denoted by a circle.

4.1 An Algorithm for Upper Edges

In this subsection, we assume that all edges in $\overline{E_{PT}}$ are upper edges (i.e., $E_l = E_o = \emptyset$). Obviously, a case of $E_u = E_o = \emptyset$ can be treated in an analogous way. First, we can see the following property from the fact that any two upper edges do not cross.

Observation. Assume that $(C_i, C_j) \in E_u$ holds and t_i, t_j are fixed. Then, the values of $t_{i+1}, t_{i+2}, \dots, t_{j-1}$ do not affect the scores of edges that do not contain any of C_{i+1}, \dots, C_{j-1} .

From this observation, we can develop a simple dynamic programming algorithm as follows (see Fig. 6). We define score $w(i, j, x, y)$ by $w(i, j, x, y) = \max_t \{ \sum_{i < h < k < j} g(h, k, t_h, t_k) \}$, where the maximum is taken from all threadings t such that $t_i = x$ and $t_j = y$ ($x < y$).

For each pair $(C_i, C_j) \in E_u$, we compute $w(i, j, x, y)$ in the following way. For each C_k , $(C_i, C_j) \in E_u$ is denoted by $\text{parent}(C_k)$ if $i < k < j$ holds and there exists no $(C_p, C_q) \in E_u$ such that $i < p < k < q \leq j$ or $i \leq p < k < q < j$. Let $(C_i = C_{r_1}, C_{r_2}, \dots, C_{r_h} = C_j)$ be a sequence of cores such that $\text{parent}(C_{r_k}) = (C_i, C_j)$ for all $1 < k < h$, and $i = r_1 < r_2 < \dots < r_h = j$. Then, $w(r_1, r_h, t_{r_1}, t_{r_h})$ is computed by:

$$w(r_1, r_h, t_{r_1}, t_{r_h}) = \begin{cases} g(r_1, r_h, t_{r_1}, t_{r_h}), & \text{if } r_h = r_1 + 1, \\ g(r_1, r_h, t_{r_1}, t_{r_h}) \\ + w_1(r_h, t_{r_h}), & \text{otherwise,} \end{cases}$$

$$\text{where } w_1(r_1, x) = \begin{cases} 0, & \text{if } x = t_{r_1}, \\ -\infty, & \text{otherwise,} \end{cases}$$

$$w_1(r_{i+1}, y) = \max_{x \leq y - c_{r_i}} \{ w(r_i, r_{i+1}, x, y) + w_1(r_i, x) \}.$$

It is easy to see that the score of an optimal threading is given by $\max_{x, y} w(1, m, x, y)$.

Here we analyze the time complexity of this procedure. First note that there are $O(m)$ edges in E_{PT} because G_{PT} is planar (with multiplicity at most 2) and $|V_{PT}| = m$. For all $(C_i, C_j) \in E_u$ and x, y ($x < y$), we must compute $w(i, j, x, y)$. Therefore, the number of $w(i, j, x, y)$'s to

be computed is $O(|E_u| + m)n^2 = O(mn^2)$. To compute each $w(i, j, x, y)$, $O(h_{i,j}n^2)$ time is required, where $h_{i,j}$ is the number such that $C_i = C_{r_1}$ and $C_j = C_{r_{h_{i,j}}}$. Since $\sum_{(C_i, C_j) \in E_u} h_{i,j}$ is $O(m)$, the total computation time is $O(mn^4)$.

However, $w_1(\dots)$'s used to compute $w(i, j, x, y)$ can also be used to compute $w(i, j, x, y')$ for $y' > y$. Using this fact, the time complexity is reduced to $O(mn^3)$.

Lemma 4.1 *An optimal threading can be computed in $O(mn^3)$ time if $E_l = E_o = \emptyset$ or $E_u = E_o = \emptyset$.*

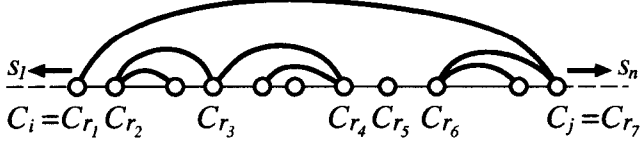


Figure 6: Computation of an optimal threading for upper edges.

4.2 An Approximation Algorithm for Loop Edges

In this subsection, we assume that all edges in $\overline{E_{PT}}$ are loop edges (i.e., $E_u = E_l = \emptyset$) until Thm. 4.4. Such a case is called a *loop case*. Note that, in a loop case, the following property holds: for any two edges $(C_i, C_j), (C_{i'}, C_{j'}) \in E_o$, $j \leq j'$ if $i \leq i'$.

First we begin with a very simple case (see Fig. 7) in which the following condition holds: there exists a number b ($1 < b \leq m$) such that every edge $(C_i, C_j) \in E_o$ must satisfy $i < b \leq j$. For each i, j such that $1 \leq i < b$ and $b \leq j \leq m$, let $w_b(i, j, x, y, z)$ be the maximum score ($\max_t \sum_{1 \leq h \leq i, b \leq k \leq j} g(h, k, t_h, t_k)$) under the condition that $t_h + c_h \leq z$ and $t_k \geq z$. Then $w_b(i, j, x, y, z)$ is determined by the following recurrence (#) (see Fig. 7):

$$w_b(1, b, x, y, z) = \begin{cases} g(1, b, x, y), & \text{if } x + c_1 \leq z \text{ and } y \geq z, \\ -\infty, & \text{otherwise,} \end{cases}$$

$$w_b(i, j, x, y, z) = g(i, j, x, y) + \begin{cases} \max_{y' \leq y - c_{j-1}} w_b(i, j-1, x, y', z), & \text{if } (C_i, C_{j-1}) \in E_o, \\ \max_{x' \leq x - c_{i-1}} w_b(i-1, j, x', y, z), & \text{otherwise.} \end{cases}$$

Computing $\max_{x, z, y} w_b(1, m, x, y, z)$ where $x < z \leq y$, we can obtain the maximum score in this very simple case.

Next we consider a case in which there exists no three edges $(C_i, C_j), (C_{i'}, C_{j'}), (C_{i''}, C_{j''})$ in E_o such that $i', i'', j' \leq i$ and $i < j''$ (see Fig. 8). In this case, E_o can be partitioned into disjoint sets (blocks) B_1, B_2, \dots, B_h where each block B_i corresponds to a very simple case (see Fig. 8). This case is called a *simple loop case*, and we say that E_o is *simple*. Let $l(B_k) = \min\{i \mid (C_i, C_j) \in B_k\}$, $m(B_k) = \min\{j \mid (C_i, C_j) \in B_k\}$, and $r(B_k) = \max\{j \mid (C_i, C_j) \in B_k\}$. For convenience, we define $l(B_{h+1}) = r(B_h)$ and $m(B_{h+1}) = r(B_{h+1}) = m$.

Lemma 4.2 *An optimal threading can be computed in $O(mn^4)$ time if E_o is simple.*

Proof. Partition E_o into B_1, B_2, \dots, B_h . For any B_i and for any x, y ($1 \leq x \leq y \leq n$), an optimal score under the condition that $t_{l(B_i)} = x$ and $t_{r(B_i)} = y$ can be computed using the recurrence (#). Since $r(B_i) \leq l(B_{i+1})$ holds, an

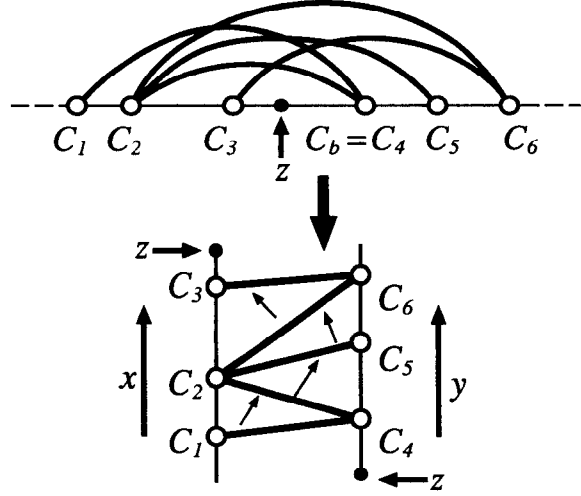


Figure 7: Example of a very simple loop case, where loop edges are described in upper half plane. In this case, an optimal threading can be computed using dynamic programming.

optimal score for E_o can be computed using a simple dynamic programming algorithm. The total computation time is $O(mn^4)$, where details are omitted here. \square

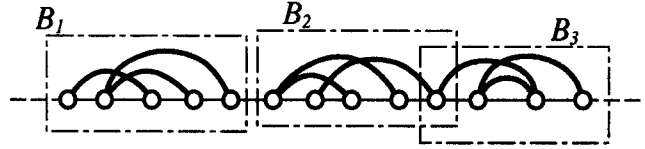


Figure 8: Example of a simple loop case. E_o can be partitioned into blocks.

Next we show that a loop case problem can be reduced to two simple loop case problems (see Fig. 9).

Lemma 4.3 *E_o can be partitioned in $O(m^2)$ time into two disjoint sets E_1 and E_2 each of which corresponds to a simple loop case.*

Proof. We use the following simple algorithm:

```

begin
  E1 := ∅;
  for i := 1 to m - 1 do for j := i + 1 to m do
    if (Ci, Cj) ∈ Eo and E1 ∪ {(Ci, Cj)} is simple
      then E1 := E1 ∪ {(Ci, Cj)};
  E2 := Eo - E1
end

```

Obviously, E_1 obtained by this algorithm is simple. Here, we consider a partition of E_1 into B_1, B_2, \dots, B_h . Then, each edge $(C_i, C_j) \in E_2$ satisfies the following condition: there exists k such that $m(B_k) \leq i \leq r(B_k)$ and $r(B_k) \leq j \leq m(B_{k+1})$. Therefore, E_2 also becomes simple. Moreover, this algorithm can be implemented so that it works in $O(m^2)$ time. \square

Combining Lemma 4.2 and Lemma 4.3, we can see that PROTEIN THREADING can be approximated within a factor 2 in $O(mn^4)$ time if $E_u = E_l = \emptyset$.

Finally, we obtain the following theorem.

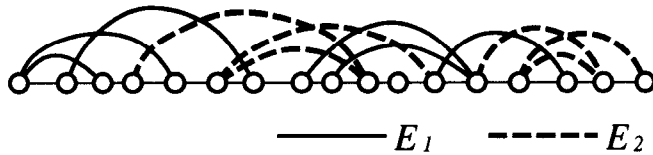


Figure 9: Partition of E_o into E_1 and E_2 .

Theorem 4.4 *PROTEIN THREADING* can be approximated within a factor 4 in $O(mn^4)$ time if $G_{PT}(V_{PT}, E_{PT})$ is planar.

Proof. We use the following simple algorithm. First compute a planar embedding of $G_{PT}(V_{PT}, E_{PT})$ and a partition of $\overline{E_{PT}}$ into E_u, E_l, E_1, E_2 . Next compute an optimal threading t_u by letting $E_l = E_1 = E_2 = \emptyset$. t_l, t_1, t_2 are computed in a similar way. Finally select the one (t_{max}) having the maximum score from t_u, t_l, t_1, t_2 .

It is easy to see that $score(t_{max}) \geq (1/4)score(t_{opt})$ holds since

$$score(t_{opt}) \leq score(t_u) + score(t_l) + score(t_1) + score(t_2),$$

where t_{opt} denotes an optimal threading. Therefore, the performance ratio of this simple algorithm is at least 4.

Since a planar embedding of $G_{PT}(V_{PT}, E_{PT})$ can be computed in $O(|V_{PT}|) = O(m)$ time [12], we can obtain a partition of $\overline{E_{PT}}$ into E_u, E_l, E_1, E_2 in $O(m^2)$ time. Therefore, the total computation time is $O(mn^4)$ from Lemma 4.1, Lemma 4.2 and $m \leq n$. \square

5 Concluding Remarks

In this paper, we have shown that approximation of the protein threading problem is hard in a general case but there exists a polynomial time approximation algorithm for a special case. However, there is still a big gap between the negative results and the positive result presented in this paper. Thus, we are currently working for shortening the gap. Such studies may lead to practical algorithms for protein threading.

Although we have not discussed details about score functions, using a good score function is also important in protein threading [10]. However, most of known score functions were not satisfactory from the viewpoint of predictive accuracy if they were applied to the threading problem. Thus, we are currently studying a method for deriving a good score function from known three-dimensional protein structures [14]. Details about the method and the experimental results will be reported elsewhere.

Acknowledgment

This work was partially supported by the Grant-in-Aid for Scientific Research on Priority Areas, "Genome Science", of the Ministry of Education, Science, Sports and Culture in Japan.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof verification and hardness of approximation algorithms," *Proc. 33rd IEEE Symp. Foundations of Computer Science*, pp. 14–23, 1992.

[2] J. U. Bowie, R. Lüthy and D. Eisenberg, "A method to identify protein sequences that fold into a known three-dimensional structures," *Science*, Vol. 253, pp. 164–170, 1991.

[3] C. Branden and J. Tooze, *Introduction to Protein Structure*, Garland Publishing, 1991.

[4] S. H. Bryant and C. E. Lawrence, "An empirical energy function for threading protein sequence through the folding motif," *PROTEINS: Structure, Function, and Genetics*, Vol. 16, pp. 92–112, 1993.

[5] C. Chothia, "One thousand families for the molecular biologist," *Nature*, Vol. 357, pp. 543–544, 1992.

[6] A. Godzik and J. Skolnick, "Sequence-structure matching in globular proteins: application to supersecondary and tertiary structure determination," *Proc. National Academy of Science USA*, Vol. 89, pp. 12098–12102, 1992.

[7] W. E. Hart and S. Istrail, "Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal," *Proc. 27th ACM Symp. Theory of Computing*, pp. 157–168, 1995.

[8] G. Kortsarz and D. Peleg, "On choosing a dense subgraph," *Proc. 34th IEEE Symp. Foundations of Computer Science*, pp. 692–701, 1993.

[9] R. H. Lathrop, "The protein threading problem with sequence amino acid interaction preferences is NP-complete," *Protein Engineering*, Vol. 7, pp. 1059–1068, 1994.

[10] R. H. Lathrop and T. F. Smith, "A branch-and-bound algorithm for optimal protein threading with pairwise (contact potential) amino acid interactions," *Proc. 27th Annual Hawaii International Conference on System Sciences*, Vol. 5, pp. 365–374, 1994.

[11] D. Maier, "The complexity of some problems on subsequences and supersequences," *J. ACM*, Vol. 25, pp. 322–336, 1978.

[12] T. Nishizeki and N. Chiba, *Planar Graphs: Theory and Algorithms*, Elsevier Science, 1988.

[13] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *J. Computer and System Sciences*, Vol. 43, pp. 425–440, 1991.

[14] H. Tashimo and T. Akutsu, "Learning score function for protein threading using linear programming," Technical Report (in Japanese), MPS-8-4, Information Processing Society of Japan, 1996.