アルゴリズム 51－3
（1996. 5. 29）

タンパク質スレッディング問題の近似について

阿久津 達也
東京大学 医科学研究所 ヒトゲノム解析センター

アミノ酸配列からのタンパク質立体構造の推定は遺伝情報解析における中心的問題の一つである。タンパク質スレッディングはそのための有効な手法の一つであり、ある種の最適化問題として定式化できる。しかしながら既にNP困難であることが証明されているので、多項式時間近似アルゴリズムについて考察する。まずMAX SNP困難であることを示し、さらにDENSE-k-SUBGRAH問題以上に近似困難であることを示す。そしてアミノ酸相互作用の影響範囲を示すグラフが平面的である場合について、最適解の4倍以内の近似解を出力するアルゴリズムを示す。この結果はβシートと呼ばれる領域の推定に応用できる可能性がある。

On the Approximation of Protein Threading

Tatsuya Akutsu
Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108 Japan
e-mail: takutsu@ims.u-tokyo.ac.jp

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 this 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 [3, 5, 7, 11]. 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) [5, 10, 11]. 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. [11] and seems reasonable.

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 minimum score)

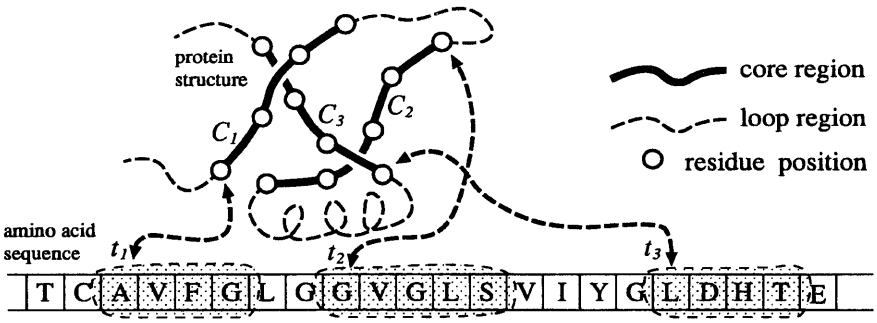

Figure 1: The protein threading problem.

directly [5, 10, 11]. Bryant and Lawrence used exhaustive enumeration to examine all possible threadings [5]. But, their method can only be applied to very small structures. Unfortunately, Lathrop proved that the protein threading problem is NP-hard [10]. 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 [11].

Although Lathrop and Smith's algorithm is very nice, long time is required if it is applied to a large protein structure. 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 can be useful from the following reasons: the correct structure could be selected if the guaranteed error bound were sufficiently small; combining with local search, approximate solutions might be lead to optimal solutions; approximate solutions might be useful for speeding up the branch-and-bound procedure. Note that an approximation algorithm has already been developed for the protein folding problem (not the inverse folding problem) although too simplified model is used [8].

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 [9]. Next we consider a special case in which a graph representing interactions between residues (amino acids) is planar. This case corresponds to most of *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 a sequence and a structure with minimum score. Lathrop and Smith defined this problem in a formal way [10, 11]. In this paper, we modify their definition into much simpler form as follows without loss of generality (see Fig. 1).

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

Output: m-tuple $\boldsymbol{t} = (t_1, \cdots, t_m)$ which maximizes a total score: $score(\boldsymbol{t}) = \sum_{i<j} g(i, j, t_i, t_j)$ 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 be felt 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, inverting the sign of values of score functions and 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 modifying $g(i,j,t_i,t_j)$ suitably. Therefore, in the above definition, there is no loss of generality.

We call $\boldsymbol{t} = (t_1,\cdots,t_m)$ a *threading* if it satisfies the above condition ($1 \leq t_1, t_i + c_i \leq t_{i+1}, t_m + c_m \leq n+1$). i'th core (length c_i) is denoted by C_i. For an instance PT of the problem, we associate a directed (multi) graph $G_{PT}(V_{PT}, E_{PT})$ such that $V_{PT} = \{C_1,\cdots,C_m\}$, $E_{PT} = \overline{E_{PT}} \cup \{(C_i, C_{i+1}) \mid 1 \leq i < m\}$, 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 where the maximum vertex degree of G_{PT} is bounded by a constant B, and a case where 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, score becomes very small when the distance between residues exceeds a threshold value [10].

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) [4]. β-sheet consists of multiple β-strands. To classify β-sheet structures, topology diagram has been used [4]. In topology diagram, β-starnds 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 as 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.

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].

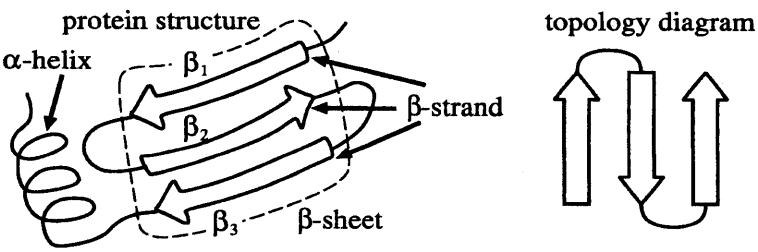

Figure 2: β-sheet and topology diagram. In this case, β_2 strongly interacts with β_1 and β_3.

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 [1, 13]. Note that the following theorem also gives much simpler proof of NP-hardness of PROTEIN THREADING than that in [10].

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 $V=\{v_1,\cdots,v_n\}$, we construct an instance of PROTEIN THREADING-B in the following way:

$$s = \overbrace{01\,01\cdots01}^{n\text{-}01\text{'s}} \text{ over } \Sigma=\{0,1\},\ c_1=c_2=\cdots=c_n=1,$$
$$g(i,j,t_i,t_j)=1 \text{ if } i<j,\ (v_i,v_j)\in E \text{ and } s_{t_i}\neq s_{t_j},\quad \text{otherwise } g(i,j,t_i,t_j)=0.$$

Then, each threading corresponds to a cut and the score of a threading is equal to the size of the corresponding cut. Therefore, this reduction is L-reduction and the theorem follows. □

Next, we show a result suggesting that approximation of PROTEIN THREADING is much harder, using a reduction from the DENSE-k-SUBGRAPH problem [2, 9]. 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 $O(n^{0.3885})$ [9].

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 $V=\{v_1,\cdots,v_n\}$, we construct an instance of PROTEIN THREADING in the following way.

$$s = \overbrace{00\cdots0}^{n-1}\,a_1\,\overbrace{00\cdots0}^{n-1}\,a_2\,\overbrace{00\cdots0}^{n-1}\cdots\overbrace{00\cdots0}^{n-1}\,a_k\,\overbrace{00\cdots0}^{n^2-kn} \text{ over } \Sigma=\{0,1\}$$
$$\text{where } a_i=1 \text{ for all } i,\ c_1=c_2=\cdots=c_n=1,$$
$$g(i,j,t_i,t_j)=1 \text{ if } i<j,\ (v_i,v_j)\in E \text{ and } s_{t_i}=s_{t_j}=1,\quad \text{otherwise } g(i,j,t_i,t_j)=0.$$

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 $\boldsymbol{t}$, we can obtain $V' \subset V$ of cardinality at most k having $score(\boldsymbol{t})$ edges in polynomial time. Since $|s|$ (length of s) is n^2, the theorem holds. □

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, where we omit the proof here.

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.*

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 a planar graph. Although we do not yet know the answer, it seems that this restricted case remains in NP-hard.

To develop an approximation algorithm, we partition a set of edges $\overline{E_{PT}}$ into three subsets (see Fig. 3): 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.

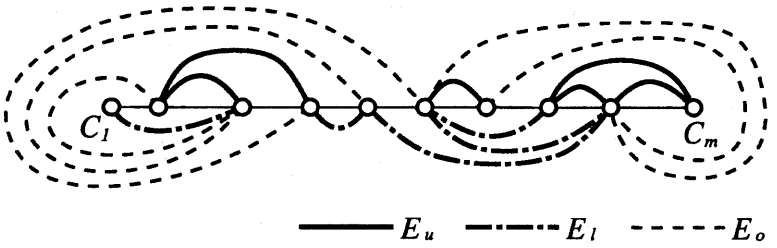

Figure 3: Partition of $\overline{E_{PT}}$ into E_u, E_l and E_o. In this figure, cores are arranged on a horizontal line where each core is denoted by a circle.

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* $t_{i+1}, t_{i+2}, \cdots, t_{j-1}$ *do not affect the scores of edges that do not contain any of* $C_{i+1}, \cdots, C_{j-1}$.

From this observation, we can develop a simple dynamic programming algorithm as follows. We define score $w(i, j, x, y)$ by $w(i, j, x, y) = \max_{\boldsymbol{t}}\{\sum_{i \le h < k \le j} g(h, k, t_h, t_k)\}$. where maximum is taken from all threadings $\boldsymbol{t}$ 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 $parent(C_k)$ if $i < k < j$ holds and there exists no $(C_p, C_q) \in E_u$

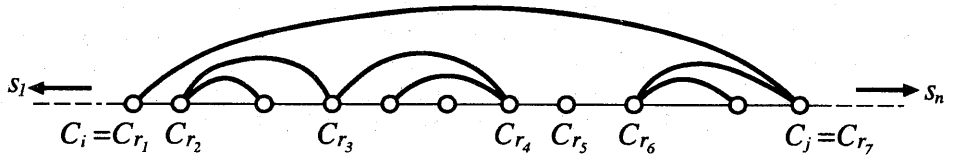

Figure 4: Computation of an optimal threading for upper edges.

such that $i < p < k < q \leq j$ or $i \leq p < k < q < j$. Let $(C_i = C_{r_1}, C_{r_2}, \cdots, C_{r_h} = C_j)$ be a sequence of cores such that $parent(C_{r_k}) = (C_i, C_j)$ for all $1 < k < h$, and $i = r_1 < r_2 < \cdots < r_h = j$. Then, $w(r_1, r_h, t_{r_1}, t_{r_h})$ is computed by:

$$
\begin{aligned}
w(r_1, r_h, t_{r_1}, t_{r_h}) &= \begin{cases} g(r_1, r_h, t_{r_1}, t_{r_h}), & \text{if } r_h = r_1 + 1, \\ g(r_1, r_h, t_{r_1}, t_{r_h}) + w_1(r_h, t_{r_h}), & \text{otherwise}, \end{cases} \\
\text{where } w_1(r_1, x) &= \begin{cases} 0, & \text{if } x = t_{r_1}, \\ -\infty, & \text{otherwise}, \end{cases} \\
w_1(r_{i+1}, y) &= \max_{x \leq y - c_{r_i}} \{w(r_i, r_{i+1}, x, y) + w_1(r_i, x)\}.
\end{aligned}
$$

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 $G_{PT}(V_{PT}, E_{PT})$ 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, $w_1(\cdots)$'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 $E_u = E_o = \emptyset$.*

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 $i \leq i'$.

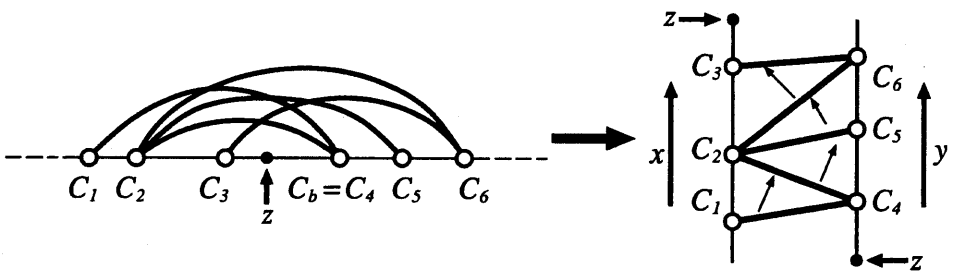

Figure 5: Example of a very simple loop case where loop edges are described in upper half plane. In this case, an optimal threading can be computed using dynamic programming.

First we begin with a very simple case (see Fig. 5) 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. 5):

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

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

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

Next we consider a case where there exists no three edges $(C_i,C_j),(C_{i'},C_{j'}),(C_{i''},C_{j''})$ in E_o such that $i',i'',j' \le i$ and $i < j''$ (see Fig. 6). In this case, E_o can be partitioned into disjoint sets (blocks) $B_1,B_2,\cdots,B_h$ where each block B_i corresponds to a very simple case (see Fig. 6). 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 $B_1,B_2,\cdots,B_h$. For any B_i and for any x,y $(1 \le x \le y \le n)$, 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 $r(B_i) \le l(B_{i+1})$ holds, an 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. □

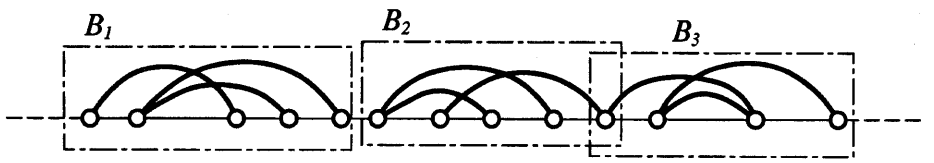

Figure 6: Example of a simple loop case. E_o can be partitioned into blocks.

Next we show that a loop case problem can be reduced to two simple loop case problems (see Fig. 7).

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
  $E_1 := \emptyset$;
  for $i := 1$ **to** $m-1$ **do** **for** $j := i+1$ **to** m **do**
    if $(C_i,C_j) \in E_o$ **and** $E_1 \cup \{(C_i,C_j)\}$ is simple **then** $E_1 := E_1 \cup \{(C_i,C_j)\}$;
  $E_2 := E_o - E_1$
end

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

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.

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.*

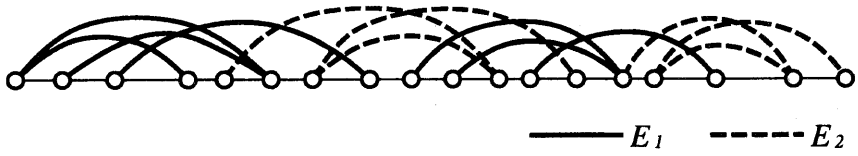

Figure 7: Partition of E_o into E_1 and E_2.

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 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.

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. Thus, the total computation time is $O(mn^4)$ from Lemma 4.1, Lemma 4.2 and $m \leq n$, and the theorem follows. □

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof verification and hardness of approximation algorithms," *Proc. FOCS*, pp. 14–23, 1992.

[2] Y. Asahiro, K. Iwama, H. Tamaki and T. Tokuyama, "Greedily finding a dense subgraph," *Manuscript*, 1996.

[3] 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.

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

[5] 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.

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

[7] 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.

[8] W. E. Hart and S. Istrail, "Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal," *Proc. STOC*, pp. 157–168, 1995.

[9] G. Kortsarz and D. Peleg, "On choosing a dense subgraph," *Proc. FOCS*, pp. 692–701, 1993.

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

[11] 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.

[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.