Protein Engineering vol.7 no.9 pp. 1059–1068, 1994

The protein threading problem with sequence amino acid interaction preferences is NP-complete

Richard H.Lathrop

Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

In recent protein structure prediction research there has been a great deal of interest in using amino acid interaction preferences (e.g. contact potentials or potentials of mean force) to align ('thread') a protein sequence to a known structural motif. An important open question is whether a polynomial time algorithm for finding the globally optimal threading is possible. We identify the two critical conditions governing this question: (i) variable-length gaps are admitted into the alignment, and (ii) interactions between amino acids from the sequence are admitted into the score function. We prove that if both these conditions are allowed then the protein threading decision problem (does there exist a threading with a score $\leq K$?) is NP-complete (in the strong sense, i.e. is not merely a number problem) and the related problem of finding the globally optimal protein threading is NP-hard. Therefore, no polynomial time algorithm is possible (unless P = NP). This result augments existing proofs that the direct protein folding problem is NP-complete by providing the corresponding proof for the 'inverse' protein folding problem. It provides a theoretical basis for understanding algorithms currently in use and indicates that computational strategies from other NP-complete problems may be useful for predictive algorithms.

Key words: contact potentials/inverse protein folding/NP-complete/protein structure prediction/protein threading/sequence–structure alignment

Introduction

The protein folding problem is to start from a string giving the protein's amino acid sequence and compute its correctly folded 3-D protein structure. It is an important problem because proteins underlie almost all biological processes and their function follows directly from their 3-D folded shape. Its importance is escalating rapidly due to the rapid increase in the number of sequences becoming available compared with the slow growth in the number of experimentally determined 3-D protein structures. The direct approach to protein folding seeks to find the folded conformation having minimum energy. This is difficult because (i) a folded protein results from the delicate energy balance of powerful atomic forces and (ii) the vast number of possible conformations poses a formidable computational barrier. The forces involved are often poorly understood or difficult to model accurately, and include stabilizing and destabilizing terms making large contributions of opposite sign summed over a very large number of atoms (Creighton, 1983). The computational burden of the direct approach has been shown to be NP-hard (widely assumed to require an exponential amount of time) even on simplified lattice models, with respect to either (i) finding minimum energy conformations or (ii) meeting endpoint and conformation (e.g. secondary structure) constraints (Ngo and Marks, 1992; Fraenkel, 1993; Unger and Moult, 1993).

One important alternative approach is to use the known protein structures as (i) spatial folding templates, (ii) additional knowledge about protein structure and (iii) constraints on possible folds. This is a powerful strategy because folded proteins exhibit recurring patterns of organization. Chothia (1992) estimates that there are only ~1000 different protein structural families. In this approach, each known protein structure (or family) 'recognizes' the protein sequences likely to fold into a similar structure. Because it starts with structures and predicts sequences instead of starting with sequences and predicting structures, it is often referred to as the 'inverse' folding problem. In its fully general sense it includes *ab initio* design of protein sequences to achieve a target structure (Pabo, 1983), but we shall restrict attention to folding given native sequences. The known structure establishes a set of possible amino acid positions in 3-D space (perhaps the spatial locations of its main-chain α carbons). 'Recognition' is mediated by a suitable score function. An alignment between spatial positions and sequence amino acids is usually a by-product of the recognition step. The sequence is given a similar 3-D fold by placing its amino acids into their aligned spatial positions. [Further techniques are necessary to correctly place the variable loop regions (Greer, 1990; Zheng *et al.*, 1993) and position the side chains (Desmet *et al.*, 1992), but the focus of this paper is on predicting and placing the conserved fold.] The process of aligning a sequence to a structure and thereby guiding the spatial placement of sequence amino acids is referred to as 'threading' the sequence onto the structure (Bryant and Lawrence, 1993). 'A threading' means a specific alignment between sequence and structure (chosen from the large number of possible alignments). In this way 'threading' specializes the more general term 'alignment' to refer specifically to a structure (considered as a template) and a sequence (considered as being arranged on the template).

Initially, such methods employed primary sequence string similarity between the candidate sequence and the native sequence of the structure to perform the threading ('homology modeling' or 'homological extension'). Computing the sequence similarity yields a direct alignment of amino acids in the sequences of the candidate and structure (Sankof and Kruskal, 1983), and hence their spatial positions. In cases where the sequence similarity is high this is still the most successful protein structure prediction method known, but it is of limited general use because few sequences have high primary sequence similarity to another whose structure is known. Recently, however, researchers have been able to extend the match process beyond primary sequence similarity and align a sequence directly to a structure.

These new approaches exploit the fact that amino acid types have different preferences for occupying different structural environments (e.g. preferences for being in α-helices or β-sheets, or for being more or less buried in the protein interior). Additionally, some of the new approaches also exploit the fact

that there appear to be distinct preferences for side-chain contact (e.g. contact potentials; Maiorov and Crippen, 1992), or more generally for spatial proximity (e.g. potentials of mean force; Sippl, 1990, 1993), as a function of those environments. For example, a buried charged residue may be more likely to be adjacent to another buried residue of opposite charge. These interaction preferences have been quantitated statistically (e.g. Miyazawa and Jernigan, 1985). They can be used to produce a score function reflecting the extent to which amino acids from the sequence are located in preferred environments and adjacent to preferred neighbors. The known protein structures can be represented in a way that makes explicit the structural environments at each position, as well as the spatially adjacent structural positions. This done, the sequence can be threaded onto the structure by searching for a threading which optimizes the score function and hence maximizes the degree to which environment and adjacency preferences are satisfied. This has been a very active area of recent research, in part because it has been somewhat successful, and various scoring schemes and threading algorithms have been proposed.

Before proceeding it is necessary to separate carefully the different approaches to protein threading that have been explored in the literature and to state clearly the problem we address. The common theme to all threading proposals is the supposition that: (i) the known structure provides a set of positions in 3-D space; (ii) these will be filled by amino acids from a sequence of unknown structure; (iii) different candidate threadings arise from different structures, different sequences and different possible alignments of structure positions and sequence amino acids; and (iv) a score function (often statistical) can distinguish good from bad threadings.

For the purposes of this paper we will assume that the structure and the sequence are fixed in advance. We will examine in detail the role of the different possible alignments and the score function.

It will turn out that the critical conditions governing computational complexity are whether or not: (i) variable-length gaps are permitted in the alignments; and (ii) the score for placing a sequence amino acid into a given structural position depends on the specific amino acid types from the sequence being threaded that are placed into neighboring (interacting) structural positions.

The variable-length gap condition arises if one wants to reflect genetic insertions and deletions. The score function condition arises if one wishes to reflect amino acid preferences for side-chain contact or spatial proximity based on the sequence amino acids proposed to actually occupy the spatial positions.

We will show that any formulation of protein threading which allows both these conditions is NP-complete and necessarily leads to an NP-hard search for the optimal threading. It is insufficient merely to observe that the problem's search space is exponentially large (e.g. by counting the number of arrangements of an ordered set of residues with possible gaps). In their proof that the direct protein folding problem is NP-complete, Ngo and Marks (1992) address in detail and lay to rest this fallacy. Many problems which have an exponentially large search space still have polynomial time solutions, and the distinctions between polynomial time and NP-complete problems are often extremely subtle. For example, the 3SAT problem discussed here is known to be NP-complete; the closely related 2SAT problem (identical except that each clause has two literals instead of three) shares the same exponentially large space of possible solutions (the 2^N possible truth value assignments to N Boolean variables), but is easily solved in polynomial time by resolution techniques (Garey and Johnson, 1979). However, no polynomial time solution is known for any NP-complete problem.

In the remainder of this section we categorize the different threading approaches according to how they accommodate the search problem. Some disallow either variable-length gaps or interactions between amino acids from the sequence being threaded, and can therefore find the optimal threading without an NP-hard search problem. Others admit both conditions, but these must choose between potentially finding only an approximation to the optimal threading or coping with a potentially exponential search.

Some approaches do not permit variable-length gaps in the alignment at all. Instead they employ a fixed-length moving window, equal in length to either the structure or the sequence under analysis. This is used to extract from a database a large number of subsequences or substructures of the same length as the candidate. The lengths being equal, they evaluate the score function at the alignment that pairs the ith sequence amino acid with the ith structure position [Hendlich *et al.*, 1990; Sippl, 1990; Crippen, 1991; Maiorov and Crippen, 1992; Sippl and Weitckus, 1992; and the sections of Bryant and Lawrence (1993) discussing ungapped alignment]. This permits the analysis to focus closely on the score function, without being distracted by other complications. The results demonstrate clearly that score functions can be devised to identify the correct match from a large number of alternatives. However, this method is of limited use in a predictive setting. Ignoring variable-length gap regions means that the structure and a novel sequence will almost invariably be partially out of registration, perhaps grossly so. Where this occurs, the correct alignment will not be found and consequently the spatial locations predicted for sequence amino acids will be wrong.

Alternatively, some approaches do not reflect the interaction preferences between amino acids from the sequence to be threaded. In this case, the score for placing a sequence amino acid of a specific type into a specific position in the structure will be independent of sequence amino acid types placed into other (neighboring) positions of the structure. For example, the score for placing a charged residue from the sequence into a position buried in the structure core will be independent of whether a sequence residue of opposite charge is placed into a spatially adjacent position. This leads to a much simpler search for the optimal threading and often performs quite well, but at the price of giving up a potentially richer source of structural information. Such approaches include: (i) ignoring contacts altogether to consider only the local environment of an amino acid (Bowie *et al.*, 1991; Lüthy *et al.*, 1992; Johnson *et al.*, 1993); (ii) taking all contacts and interactions to be to generic bulk peptide instead of to specific amino acid types (Ouzounis *et al.*, 1993); or (iii) evaluating interaction preferences with respect to residues from the structure's original native sequence instead of from the sequence being threaded (Sippl, 1993; Wilmanns and Eisenberg, 1993). These are all fundamentally equivalent. The crucial common condition is that the score for placing any amino acid type into any structural position does not vary as different amino acid types from the sequence being threaded are placed into other positions. Although the score function may have been parameterized by considering residue interactions or contacts (as in Ouzounis *et al.*, 1993; Sippl, 1993; Wilmanns and Eisenberg,

1993), it never looks at more than one residue at a time from the sequence being threaded. This satisfies the fundamental assumption of the dynamic programming alignment method (Sankof and Kruskal, 1983) which can be used without major modification to find the optimal threading (alignment) between sequence and structure. For these approaches, the search for the optimal threading is easily accomplished in polynomial time.

For the remainder of this paper we will assume that variable-length gaps are admitted and that the score function attempts to reflect the interaction preferences between amino acid types from the sequence being threaded. This greatly complicates the search for the optimal threading, as proved below. The essential reason is that the score achieved by placing a given amino acid type into a given structural position now varies depending on what amino acid types are placed into adjacent positions, and consequently decisions are no longer purely local. Here researchers are divided on whether to give up on finding the optimal threading, or to give up on using a polynomial time algorithm.

On the one hand, researchers who do not guarantee finding the optimal threading have modified the dynamic programming alignment method to yield an approximate solution in polynomial time. Godzik *et al.* (1992) substitute the original motif residues, or previously aligned sequence residues in subsequent iterative steps, for the neighbors (their 'frozen approximation'). These researchers characterize both pairwise and triplet interactions among sequence amino acids; although we will consider only pairwise interactions in this paper, as discussed below pairwise interactions reduce to triplet interactions as a special case. Finkelstein and Reva (1991) and Goldstein *et al.* (1992) also use this iterative approach. Jones *et al.* (1992) use a modified dynamic programming routine from Taylor and Orengo (1989; see also Orengo and Taylor, 1990) that employs a secondary level of dynamic programming to fix the neighbors for the first level.

On the other hand, researchers who guarantee the globally optimal threading necessarily have used some form of search which may take exponential time in the worst case. Bryant and Lawrence (1993) used exhaustive enumeration to examine all possible threadings (their 'gapped alignment'). They set upper and lower bounds on the gap length considered, based on the observed gap lengths from aligned homologous sequences. Lathrop and Smith (1994) used a branch and bound search, which heuristically prunes unpromising regions of the search space. While it has found the globally optimal threading at speeds equivalent to as many as 10^{24} threadings per hour (most of which are pruned before they are ever examined explicitly), it still exhibits exponential growth.

The twin horns of this dilemma, whether to give up optimality or efficiency, lead to the questions which concern this paper: Is it possible to have it all? Is it possible to allow variable-length gaps, to exploit the additional structural information potentially present in pairwise amino acid interactions, and still to construct a polynomial time algorithm for finding the optimal threading of a sequence onto a structure? We will answer these questions in the negative (unless P = NP) by showing that the task is NP-hard in the strong sense.

Methods

The method of choice for answering these questions is the theory of computational complexity and NP-completeness. An exhaustive review is beyond the scope of this paper, and this section presents only enough material to motivate the results below. The interested reader is referred to Garey and Johnson (1979), Hopcroft and Ullman (1979) and Lewis and Papadimitriou (1981) for formal treatment of the subject, and to Fraenkel (1993), Ngo and Marks (1992) and Unger and Moult (1993) for discussions of its biological relevance. The reader already knowledgeable on the topic should skip to the next section.

The analysis of computational complexity is concerned with the question of how the running time of an algorithm grows as the size of its input increases. For a given algorithm this is made specific by naming some function, f, and asserting that the algorithm's running time grows with inputs of increasing size 'no faster' than f grows with arguments of increasing magnitude. Formally, let $\#(A, I)$ denote the running time (number of steps) of algorithm A when started with input I, and let $|I|$ be some reasonable measure of the size of I. Then we write $A = O(f)$, read 'A is order of f' (or 'A is O of f'), if there exists any positive constant C such that:

$$\lim_{|I| \to \infty} [\#(A, I)/f(|I|)] \leq C$$

Algorithms whose computational complexity is the order of some polynomial ('polynomial time algorithms') are considered to be formally efficient. All other algorithms have a running time that grows faster than any possible polynomial and are considered to be inefficient. It is possible, of course, that an exponential time algorithm with a tiny exponent may terminate rapidly on small and medium-sized inputs, or that a polynomial time algorithm applied to a very large input may not. Nonetheless, the distinction is valuable and important in most practical cases.

A 'problem' is a class of computational tasks defined in terms of a set of parameters (e.g. SEQUENCE is a parameter of the protein threading problem). A problem 'instance' results from replacing the parameters by actual values (e.g. replacing SEQUENCE by a specific string). An algorithm solves a problem if it terminates correctly on every instance of the problem. It is customary to identify the computational complexity of a problem with that of the most efficient algorithm that solves it. The class of problems that can be solved by a polynomial time algorithm is named 'P.' Problems which belong to this class are formally tractable. The class of problems whose solution, once found or guessed, may be verified in polynomial time is named 'NP.' Note that it may not be possible to actually find a solution in polynomial time; the condition refers only to the verification of a putative solution. Problems in NP are solvable in 'non-deterministic polynomial time', meaning that if one could somehow non-deterministically guess and check all possible solutions simultaneously, the solution would be obtained in polynomial time. The practical importance of this theoretical condition is that the search for a solution, not the verification step, determines whether a polynomial time solution is possible or not. Clearly, P is a subset of NP; but it is unknown whether P = NP.

There is a curious class of problems which belong to NP and have the property that an algorithm solving any problem in the class can be transformed in polynomial time to solve any other problem in NP. Therefore, a polynomial time algorithm solving any problem in this class would yield immediately a polynomial time solution for every problem in NP. These problems are the hardest in NP and are known as the 'NP-complete' problems. It is not known whether or not they have a polynomial time solution (if so, then P = NP because polynomials are closed under composition). They

include many problems deeply central to computer science, and so a great deal of effort by a great many talented people has been expended searching for a polynomial time solution to any one of them. Because so many people have failed, it is widely accepted that no polynomial time algorithm is likely to be found.

In some cases it is possible to prove directly from first principles that a problem at hand is NP-complete, but this is usually quite difficult. Most proofs proceed by constructing a polynomial time transformation of another problem, already known to be NP-complete, into an instance of the problem at hand. It follows that if the problem at hand could be solved in polynomial time, so could the other problem, and therefore by extension all of the problems in NP. Consequently, the problem at hand is NP-complete.

NP-complete problems are phrased as decision problems to which the answer is either 'yes' or 'no.' For example, the decision problem addressed by this paper is, 'Does there exist a threading of this sequence onto this structure under this score function, such that the threading score is $\leq K$?' This might be the case in which a candidate threading has already been found and one wishes simply to ask whether or not another threading with a better score exists. For many NP-complete decision problems there is an associated optimization problem for which the task is to produce an optimal solution. For example, the associated optimization problem in this paper is, 'Find the threading of this sequence onto this structure under this score function having the optimal (minimum) score.' It is easy to see that the optimization problem cannot be easier than the decision problem. This is because a polynomial time solution to the optimization problem could be transformed into a polynomial time solution to the decision problem. Thus, if the optimization problem is solvable in polynomial time, then so is the decision problem. For example, if we could find the optimal threading in polynomial time then it would be easy to compute its score (also in polynomial time). This would let us answer the decision question of whether there exists a threading with a score $\leq K$ by checking to see if the optimal score was $\leq K$. Problems bearing this relationship to an NP-complete problem are called NP-hard.

Finally, we note that problems can be NP-complete for different reasons. In some cases a polynomial time solution fails only because the numbers associated with the problem can become exponentially large in magnitude (e.g. perhaps the binary bits specifying an integer are used to encode some other non-numeric information). In most cases these numbers are integers; more complicated numbers are treated theoretically as composites of several distinct integers. If a problem is NP-complete, and remains NP-complete when restricted to problem instances for which the magnitude of the largest integer is bounded by a polynomial in the problem instance size, then the problem is called NP-complete in the strong sense.

Results

The principal technical result of this paper is a formal proof that the protein threading problem is NP-complete if one admits both variable-length gaps and pairwise interactions between amino acids from the sequence being threaded (we name this problem 'PRO-THREAD'). This section summarizes the principal intuitions underlying the problem statement and proof. Full mathematical details are given in Appendix. The reader interested only in the practical implications of the proof should skip to the next section.

We will state the protein (or motif) threading problem in sufficient generality to cover a wide range of cases. The problem and formalisms are equally applicable to threading protein core motifs or super-secondary structure motifs. Our general definition is similar to that of Bryant and Lawrence (1993). Details are contained in Lathrop and Smith (1994), from which many of the definitions here are drawn. A probability analysis appears in White *et al.* (1994). Although for generality the problem is stated in terms of core segments of contiguous amino acids, the proof actually uses segments of length exactly 1. Consequently, it applies equally well to formulations that admit gaps between any pair of amino acids (e.g. Godzik *et al.*, 1992; Jones *et al.*, 1992), provided that they model explicit sequence amino acid interactions.

Informal problem motivation

All current threading proposals replace the 3-D coordinates of the known structure by an abstract description in terms of core elements and segments, neighbor relationships, distances, environments and the like. This avoids the computational cost of full-atom molecular mechanics or dynamics (Weiner *et al.*, 1984; Brooks *et al.*, 1990) in favor of a much less detailed, and hence much faster, discrete alignment between sequence and structure. However, important aspects of protein structure (such as intercalated side-chain packing, excluded volume, constraints linking different environments, higher-order interactions and so forth) may also be abstracted away. This depends on the theory employed. The threading research challenge, only partially met at present, is to devise a theory of protein structure sufficiently information-rich to allow accurate prediction yet sufficiently concise to retain the simplicity of discrete alignment.

Spatial core 'positions' (implying a specific 3-D location) are abstracted to become spatially neutral core 'elements' (implying only a discrete place-holder that may be occupied by a residue from the sequence). The remaining abstractions are driven by the requirements of the score function (which in turn is driven by the underlying theory of protein structure). We need record only information that the score function will later use to assign scores to candidate threadings [e.g. Bryant and Lawrence (1993) and Sippl (1990) both record discrete distances]. Such information comprises the abstract 'structural environment' as seen by the threaded residues. If the score function quantitates individual residue preferences (e.g. for being more or less buried, or for certain secondary structures), we must record the (singleton) structural environment at each element. This is easily carried out by labeling the element. Pairs of elements are 'neighbors' if they interact for the given score function. If the score function quantitates neighbor preferences (e.g. for being a certain distance apart), we must record the (pairwise) structural environment between neighbors. Several equivalent data structures have been used; the common theme is that certain pairs of elements are distinguished as neighbors, and the pair is labeled in some manner. For pairwise interactions this is fully general because each label could (in principle) specify a different 20×20 table with an arbitrary score for any possible pair. One common data structure constructs a matrix with one row and one column for each element. Each cell contains the corresponding pairwise environment [e.g. a distance matrix (Sippl, 1990) results when all elements are neighbors and the pairwise environment is Euclidean distance]. An equivalent approach, the adjacency graph used in this paper, constructs a graph with one vertex for each element. Neighbor elements are connected by a

(directed) edge, and the edge is labeled with the pairwise environment. The edge in the graph corresponds to the cell in the matrix, and the edge label corresponds to the label contained in the cell. Related representations include adjacency matrix, contact graph, and so on.

In this framework, a protein core folding motif, C, consists of m core segments, C_i, each representing a set of contiguous amino acid positions. Core segments are usually the pieces of secondary structure comprising the tightly packed internal protein core. As we make no restriction on core segment length, they could equally well represent only a single amino acid. The jth element of C_i is the core element $C_{i,j}$. Core segments are connected by a set of loop regions which undergo genetic insertions and deletions over evolutionary time and are not usually considered part of the conserved core motif. The loop regions might equally well be the 'gaps' (in a dynamic programming alignment sense) used by some formulations.

The adjacency graph consists of a set V of vertices and a set E of edges. Each core element $C_{i,j}$ corresponds one-to-one to a graph vertex $v \in V$. Consequently, the adjacency graph vertices merely relabel the core elements. Those pairs of vertices which interact in the score function (neighbors) are connected by a graph edge $e \in E$. Each vertex and each edge is labeled by an environment. The vertex (amino acid) environment labels, L_v, may describe local factors such as degree of solvent exposure, local secondary structure type and so forth. The edge environment labels, L_e, may encode distance or contact between amino acids, the local environments at each end of the edge, and so forth. The edges are directed because the local environments at the edge head and tail may differ.

The protein sequence, $\mathbf{a}$, consists of amino acids a_i, each from one of 20 naturally occurring amino acid types. A given threading of $\mathbf{a}$ into C associates one amino acid from $\mathbf{a}$ with each core element $C_{i,j}$, subject to the constraint that successive amino acids in the sequence necessarily fall into successive core elements in C_i and that the core segments do not overlap. A given threading of $\mathbf{a}$ into C may be described completely by the primary sequence indices of the amino acids placed in the first element of each core segment. This is compactly specified by an m-tuple $\mathbf{t} = (t_1, t_2, \ldots, t_m)$, where t_i is the primary sequence index of the amino acid that the threading places in $C_{i,1}$.

For a specific core motif C and protein sequence $\mathbf{a}$, the score of a candidate threading is defined to be the sum of the scores of the vertices and edges in the adjacency graph and the bulk composition of the loop regions. By analogy to energy minimization, lower scores are considered better. The vertex score $\sigma_v(a, d)$ depends only on the vertex environment label d and the amino acid type a threaded into the vertex. The edge score $\sigma_e(a, b, d)$ depends only on the edge environment label d and the amino acid types a and b threaded into the vertices at the head and tail of the edge. For $\tilde{\mathbf{a}}$, a subsequence of $\mathbf{a}$, the loop score $\sigma_l(\tilde{\mathbf{a}})$ depends only on the length and bulk amino acid composition of $\tilde{\mathbf{a}}$ (thus, it could represent the 'gap penalty' in a dynamic programming alignment sense). The specific numeric values depend on the particular scheme chosen to assign environments and scoring function.

Informal sketch of proof

The bulk of the proof consists of constructing an encoding from ONE-IN-THREE 3SAT into PRO-THREAD. The remainder of this section briefly and informally sketches the encoding. The reader interested in formal details should turn to Appendix.

The canonical (and first) NP-complete problem is SATISFIABILITY. A problem instance consists of a set of Boolean variables plus a set of Boolean clauses (a clause is a disjunction, i.e. logical OR, of a set of literals; a literal is either one of the variables or the negation of one of the variables). The question is whether any setting (truth-value assignment) of the variables makes all of the clauses true simultaneously. 3SAT is a well-known variant which restricts the clauses to contain exactly three literals. ONE-IN-THREE 3SAT is a further variant of 3SAT which requires that each of the clauses be made true by exactly one of the three literals. All these problems are known to be NP-complete (Schaefer, 1978; Garey and Johnson, 1979, p. 259).

The proof that PRO-THREAD is NP-complete proceeds by showing that we can encode any arbitrary instance of ONE-IN-THREE 3SAT (does there exist a setting of the Boolean variables making all the clauses simultaneously true by exactly one literal?) as an equivalent instance of PRO-THREAD. Threadings with a score of 0 encode solutions of the original ONE-IN-THREE 3SAT problem; threadings with positive scores encode failures. The equivalent encoded PRO-THREAD question is: Does there exist a threading with a score of 0 or less? The answer to this question is 'yes' exactly when a solution exists for the original ONE-IN-THREE 3SAT problem.

The essence of the proof is as follows.

(i) Amino acids from the sequence can encode whether a Boolean variable is TRUE (by T, a threonine residue) or FALSE (by F, phenylalanine); and also which literal makes a Boolean clause true (Q, glutamine encodes the first literal; R, arginine, the second literal; and S, serine, the third literal). In the encoded problem, the sequence $\mathbf{a}$ to be threaded is $\mathbf{a}$ = QRSQRSQRSQRS . . . QRSTF . . . TFTFTFTF, where we allot one 'QRS' for each clause, and one 'TF' for each Boolean variable.

(ii) By making each core segment exactly one element long, it is threaded to exactly one amino acid. Consequently, any given threading assigns every core segment to one of (Q, R, S, T, F). (As discussed below, extensions that add 'GAP' to this list are also NP-complete.)

(iii) We can use one core segment to encode each Boolean clause and choose which literal makes it true by threading it to Q (= the first literal), R (= the second literal), or S (= the third literal) in the sequence $\mathbf{a}$. Similarly, one core segment encoding each Boolean variable is threaded to T (= TRUE) or F (= FALSE), and thereby chooses truth values.

(iv) Pairs of core elements are taken as neighbors in the core (and recorded as such in the adjacency graph) exactly when the clause encoded by the first element contains a literal naming the variable encoded by the second. The edge environment label assigned is an ordered pair, $d = (i, j)$, that encodes which literal (i = 1, 2 or 3) is involved and whether the variable was negated or positive (j = N or P).

(v) An edge score $\sigma_e(a, b, d)$ can be written that is 0 when the edge label d is consistent with the literal choice encoded by amino acid a (as Q, R or S) and the truth-value encoded by amino acid b (as F or T); and is 1 otherwise.

(vi) By summing $\sigma_e(a, b, d)$ over all edges, a score function can be written that is 0 when a candidate threading encodes a truth-value and literal assignment correctly solving the original problem, and positive otherwise. The question 'Does there exist a threading with a score of 0 or less?' is now equivalent to the original ONE-IN-THREE 3SAT question.

(vii) Thus, if we could solve the general PRO-THREAD problem in polynomial time, we could solve ONE-IN-THREE 3SAT in polynomial time. PRO-THREAD is NP-complete.

In fact, PRO-THREAD is NP-complete in the strong sense (i.e. is not a number problem) because the only numbers used in the construction are 0 and 1. The optimization problem, to produce an optimal threading, is NP-hard.

Discussion

We identified the two critical conditions governing the computational complexity of protein threading as whether or not: (i) variable-length gaps are permitted in the alignments; and (ii) the score for placing a sequence amino acid into a given structure position depends on the specific amino acid types from the sequence being threaded that are placed into neighboring (interacting) structure positions.

The condition of variable-length gaps plays a different role to the condition of pairwise interactions between amino acids from the sequence being threaded. Allowing variable-length gaps ensures that the search space of possible solutions is exponentially large. This is necessary, but not sufficient, for a problem to be NP-complete. Allowing pairwise interactions between sequence amino acids, on the other hand, ensures that the solution must take non-local effects explicitly into account; decisions cannot be made merely by inspecting a single amino acid at a time, and local changes can have a non-local 'ripple' effect.

We proved that if both these conditions are allowed, then the protein threading decision problem (Does there exist a threading with a score $\leq K$?) is NP-complete in the strong sense, and the related problem of finding the globally optimal protein threading is NP-hard.

This fact imposes severe constraints on the design of protein threading algorithms. Barring a significant breakthrough in computational theory (i.e. a proof that P = NP), any protein threading algorithm must adopt (at least) one of the following four choices: (i) it must fail to admit variable-length gaps into the alignment; (ii) it must fail to admit interaction preferences between amino acids from the sequence being threaded into the score function; (iii) it must fail to find the optimal threading in some cases; or (iv) it must require an exponential amount of time in some cases.

In our categorization of current protein threading algorithms in the Introduction, we gave at least two examples from the literature for each of these strategies.

One general strategy for coping with NP-complete problems is to restrict the problem to a subclass having a polynomial time solution. For example, the NP-complete problem SATISFIABILITY has a subproblem 2SAT in which each clause is restricted to exactly two literals, and which is solvable in polynomial time. If one is fortunate, the restricted problem will still cover the problem instances one actually encounters in practice. This is more likely if one can identify and exploit particular features and constraints about the problem instances actually encountered. For example, the physical constraint of packing amino acids into space restricts adjacency graphs in native proteins to a subclass of all possible graphs. As another example, native proteins requiring exponential time to fold might be strongly selected against, even though some (possibly artificial) sequences might require it. The hydrophobic core (widely believed to drive folding) is an example of a native feature which might enforce this constraint. If true, this would result in a restricted class of sequences to consider. If discovered, such restrictions could be reflected in specialized threading and folding algorithms.

Failing this, protein threading (and folding) algorithms can draw on a wealth of computational methods that have been developed to cope with NP-complete problems. For example, the knapsack problem (how to pack objects of different sizes and values into a finite volume so as to maximize the total contained value) is known to be NP-complete, but branch-and-bound algorithms have been so successful that many consider it to be an efficiently solvable problem, even though the algorithms involved (of course) have a formal exponential time complexity (Garey and Johnson, 1979, p. 9). Packing objects into a knapsack is analogous to packing amino acids into a protein core, and a branch-and-bound algorithm has been employed successfully to find the globally optimal threading in the general protein threading problem (Lathrop and Smith, 1994). A number of other computational techniques, such as simulated annealing and genetic algorithms, can be used to find good approximate solutions to NP-complete problems. As expected, researchers are actively pursuing each of these avenues.

The basic proof can be used to prove that many threading methodology extensions and generalizations are also NP-hard. The general strategy in such cases is to first show that the extended problem remains in NP (because a putative solution can be checked in polynomial time), and then to show that the problem has not been made easier (by exhibiting some setting of the extended parameters for which the extended problem can be made to solve PRO-THREAD). Consequently, a polynomial time solution to the extended problem would imply a polynomial time solution to the simpler PRO-THREAD. Without producing formal proofs, we sketch this for three cases of interest: (i) allowing a core element to be unoccupied (threaded to a gap), as some dynamic programming methods permit; (ii) the inclusion of triplet or higher-order terms; and (iii) the presence of constraint equations on environment labels. Suppose we allow unoccupied elements. A method for solving this problem can be made to solve PRO-THREAD by using a score function that assigns any such threading a positive score. Similarly, extensions including triplet or higher-order terms can be made equivalent to PRO-THREAD by employing a score function that assigns all such terms a score of 0. Extensions which admit constraint equations on environment labels can be made to solve PRO-THREAD by adding tautologically true constraint equations to the original PRO-THREAD problem. In general, any problem which includes PRO-THREAD as a special case remains NP-hard.

Finally, one might ask whether it is necessary to find the globally optimal threading at all. Might not a good approximate solution be sufficient? In truth, it depends on one's goals. Many threading studies seek only to pair sequences with the structure into which they will fold, without attempting to actually place the sequence amino acids in space. That is, they undertake only the 'recognition' aspect of threading. For this purpose, a good approximate estimate of the true optimal score may be perfectly satisfactory, since it will only be used for comparison with other approximate scores from other structures. Alternatively, one often is interested in actually placing amino acids in space. This is, after all, the eventual goal of protein structure prediction. Each different threading corresponds to a different structure assignment. Assuming a perfect score function (a strong assumption!), only the globally optimal threading will be placed correctly in space. Every

approximate solution will correspond to a misfolded protein. Whether finding the globally optimal threading is necessary or not can only be answered relative to the goals that led one to attempt the threading.

Acknowledgements

The author thanks Barbara Bryant, Jim Cornette, Michael de la Maza, Ilya Muchnik, Kimmen Sjölander, Lisa Tucker-Kellog and the anonymous referees for comments that improved the manuscript, Jim White for discussions of the mathematical formalism, and Temple Smith for discussions of proteins. This report describes research performed at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, in consortium with the BioMolecular Engineering Research Center of Boston University, sponsored by the National Science Foundation under grant DIR-9121548. Support for the Artificial Intelligence Laboratory's research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-91-J-4038 Support for the BioMolecular Engineering Research Center's research is provided in part by the National Institutes of Health under grant RR02275-05

References

Bowie,F.U., Lüthy,R. and Eisenberg,D. (1991) *Science*, **253**, 164–170.
Brooks,C.L., Karplus,M and Pettitt,B.M. (1990) *Proteins: A Theoretical Perspective of Dynamics, Structure and Thermodynamics.* John Wiley and Sons, New York.
Bryant,S.H and Lawrence,C.E. (1993) *Proteins: Struct. Funct Genet.*, **16**, 92–112.
Chothia,C. (1992) *Nature*, **357**, 543–544
Cook,S.A. (1971) In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing.* Association for Computing Machinery, New York, pp. 151–158.
Creighton,T.E. (1983) *Biopolymers*, **22**, 49.
Crippen,G.M. (1991) *Biochemistry*, **30**, 4232–4237.
Desmet,J., De Maeyer,M., Hazes,B. and Lasters,I (1992) *Nature*, **356**, 539–542.
Finkelstein,A V and Reva,B. (1991) *Nature*, **351**, 497–499
Fraenkel,A.S. (1993) *Bull. Math. Biol.*, **55**, 1199–1210
Garey,M.R. and Johnson,D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H.Freeman and Company, New York.
Godzik,A., Kolinski,A. and Skolnick,J. (1992) *J. Mol. Biol.*, **227**, 227–238.
Goldstein,R A , Luthey-Schulten,Z.A. and Wolynes,P.G. (1992) *Proc Natl Acad. Sci. USA*, **89**, 9029–9033.
Greer,J. (1990) *Proteins: Struct. Funct Genet.*, **7**, 317–333.
Hendlich,M , Lackner,P., Weitckus,S., Floeckner,H , Froschauer,R., Gottsbacher,K., Casari,G. and Sippl,M.J. (1990) *J. Mol. Biol.*, **216**, 167–180.
Hopcroft,J.E and Ullman,J.D. (1979) *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA.
Johnson,M.S., Overington,J.P and Blundell,T.L (1993) *J. Mol. Biol.*, **231**, 735–752
Jones,D.T., Taylor,W.R. and Thornton,J.M. (1992) *Nature*, **358**, 86–89.
Lathrop,R.H. and Smith,T.F. (1994) *Proceedings of the 27th Hawaii International Conference on System Sciences.* IEEE Computer Society Press, Los Alamitos, CA, pp. 365–374.
Lewis,H.R and Papadimitriou,C.H. (1981) *Elements of the Theory of Computation.* Prentice-Hall, Englewood Cliffs, NJ.
Lüthy,R , Bowie,J.U. and Eisenberg,D. (1992) *Nature*, **356**, 83–85
Maiorov,V.N. and Crippen,G.M. (1992) *J. Mol. Biol.*, **227**, 876–888.
Miyazawa,S. and Jernigan,R.L. (1985) *Macromolecules*, **18**, 534–552.
Ngo,J.T. and Marks,J. (1992) *Protein Engng*, **5**, 313–321.
Orengo,C.A. and Taylor,W.R. (1990) *J. Theor. Biol.*, **147**, 517–551
Ouzounis,C., Sander,C., Scharf,M. and Schneider,R. (1993) *J. Mol. Biol.*, **232**, 805–825.
Pabo,C. (1983) *Nature*, **301**, 200.
Sankof,D. and Kruskal,J.B. (1983) *Time Warps, String Edits and Macromolecules.* Addison-Wesley, Reading, MA.
Schaefer,T.J. (1978) *Proceedings of the 10th Annual ACM Symposium on Theory of Computing.* Association for Computing Machinery, New York, pp. 216–226.
Sippl,M.J. (1990) *J. Mol. Biol.*, **213**, 859–883.
Sippl,M.J. (1993) *J. Comput.-Aided Mol. Design*, **7**, 473–501.
Sippl,M.J. and Weitckus,S. (1992) *Proteins: Struct. Funct. Genet.*, **13**, 258–271.
Taylor,W.R. and Orengo,C.A. (1989) *J. Mol. Biol.*, **208**, 1–22.
Unger,R. and Moult,J. (1993) *Bull. Math. Biol.*, **55**, 1183–1198.
Weiner,S.J., Kollman,P.A., Case,D.A., Singh,U.C., Ghio,C., Alagona,G., Profeta,S. and Weiner,P. (1984) *J. Am. Chem. Soc.*, **106**, 765–784.
White,J., Muchnik,I. and Smith,T.F. (1994) *Math Biosci.*, in press.
Wilmanns,M. and Eisenberg,D. (1993) *Proc. Natl Acad. Sci. USA*, **90**, 1379–1383.
Zheng,Q., Rosenfeld,R., Vajda,S. and DeLisi,C. (1993) *Protein Sci.*, **2**, 1242–1248.

Received January 3, 1994; revised March 28, 1994; accepted May 16, 1994

Appendix

A formal problem statement and proof

Here we give a formal statement of the PRO-THREAD problem and state a formal proof that it is NP-complete.

A formal problem statement (PRO-THREAD)

Table I summarizes the notational usage of this paper.

Let G be a labeled directed graph having a set of vertices $V = (v_1, v_2, \ldots, v_p)$, a set of edges $E = (e_1, e_2, \ldots, e_q)$, a set of vertex labels L_v and a set of edge labels L_e. Let s map vertices to vertex labels and edges to edge labels. Let $C = (C_1, C_2, \ldots, C_m)$ be a partition of V. Let $C_{i,j}$ denote the jth

Table I. Notational usage of this paper

Notation	Usage
a	a sequence of characters drawn from AA; the concatenation of $\mathbf{a}_B$ and $\mathbf{a}_U$
$\tilde{\mathbf{a}}$	a subsequence of **a**
AA	an alphabet of 20 characters (amino acids)
$\mathbf{a}_B$	g instances of the sequence (Q, R, S), concatenated together
$\mathbf{a}_U$	h instances of the sequence (T, F), concatenated together
B	a set of Boolean clauses (disjunctions); b_k is the kth clause
C	a set of core segments; a partition of V
C_i	a core segment; the ith element of C; a subset of V
$C_{i,j}$	a core element; the jth element of C_i; another indexing of a vertex of V
c_i	$\lvert C_i\rvert$, the length of the ith core segment
E	the set of edges of the graph G; e_i is the ith edge
E_k	a subset of E; those edges whose tail is v_k
F	the amino acid phenylalanine
f	a function mapping an m-tuple of integers to a number
f_e	a function mapping an edge and an m-tuple of integers to a number
f_I	a function mapping an integer and an m-tuple of integers to a number
f_v	a function mapping a vertex and an m-tuple of integers to a number
G	a labeled, directed graph
g	$\lvert B\rvert$, the number of Boolean clauses
h	$\lvert U\rvert$, the number of Boolean variables
K	a fixed number
L_e	the set of edge (environment) labels of the graph G
L_v	the set of vertex (environment) labels of the graph G
m	$\lvert C\rvert$, the number of core segments
n	$\lvert \mathbf{a}\rvert$, the length of the sequence **a**
Q, R, S, T	the amino acids glutamine, arginine, serine, threonine
s	a function mapping vertices to vertex labels and edges to edge labels
t	a threading; an m-tuple of integers; t_i is the ith coordinate
U	a set of Boolean variables; u_i is the ith variable
V	the set of vertices of the graph G; v_i is the ith vertex
z	a literal from a clause of B
α	α_1 = Q, α_2 = R, α_3 = S
π	a function mapping a vertex and an m-tuple of integers to a character
σ_e	a function mapping two characters and an edge label to a number
σ_I	a function mapping a subsequence of **a** to a number
σ_v	a function mapping a character and a vertex label to a number

element of C_i (yielding a second indexing of V), and let $c_i = |C_i|$. For $e \in E$ let *head* (e) [respectively *tail* (e)] return the vertex at the head (respectively tail) of e.

Let AA be an alphabet of 20 characters (amino acids) and let $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ be a sequence of n characters drawn from AA.

Let $\mathbf{t} = (t_1, t_2, \ldots, t_m)$ be an m-tuple such that $1 \leq t_1$, that $t_i + c_i \leq t_{i+1}$ for $1 \leq i < m$, and that $t_m + c_m \leq n + 1$. The m-tuple $\mathbf{t}$ specifies a threading.

Let $\pi(v, \mathbf{t})$ be a function that maps vertices to characters in $\mathbf{a}$ according to $\mathbf{t}$, defined for $v = C_{i,j}$ as $\pi(v, \mathbf{t}) = a_{t_i + j - 1}$. The function π yields the sequence character (amino acid) threaded into vertex v by $\mathbf{t}$.

Let $\sigma_v(a, d)$ map the character $a \in$ AA and the vertex label $d \in L_v$ to a number. Let $\sigma_e(a, b, d)$ map the characters $a, b \in$ AA and the edge label $d \in L_e$ to a number. If $\tilde{\mathbf{a}}$ is any contiguous subsequence of $\mathbf{a}$, let $\sigma_f(\tilde{\mathbf{a}})$ map $\tilde{\mathbf{a}}$ to a number. These are score functions for vertices (amino acids), edges (interacting amino acid pairs) and loop regions (gaps), respectively.

Extend these functions to a score function f mapping $\mathbf{t}$ to a number as follows. Let:

$$f_v(v, \mathbf{t}) = \sigma_v(\pi(v, \mathbf{t}), s(v)), \tag{1}$$

$$f_e(e, \mathbf{t}) = \sigma_e(\pi(head\ (e), \mathbf{t}), \pi(tail\ (e), \mathbf{t}), s(e)), \tag{2}$$

$$f_f(i, \mathbf{t}) = \begin{cases} \sigma_f((a_1, a_2, \ldots, a_{t_1 - 1})), & \text{if } i = 0, \\ \sigma_f((a_{t_i + c_i}, \ldots, a_{t_{(i+1)} - 1})), & \text{if } 1 \leq i < m, \\ \sigma_f((a_{t_m + c_m}, \ldots, a_n)), & \text{if } i = m. \end{cases} \tag{3}$$

Finally,

$$f(\mathbf{t}) = \Sigma_{v \in V} f_v(v, \mathbf{t}) + \Sigma_{e \in E} f_e(e, \mathbf{t}) + \Sigma_{i=0}^{m} f_f(i, \mathbf{t}). \tag{4}$$

The decision problem is: For given G, s, C, $\mathbf{a}$ and f, and a fixed number K, does there exist a $\mathbf{t}$ such that $f(\mathbf{t}) \leq K$?

The ONE-IN-THREE 3SAT problem

SATISFIABILITY, which asks whether there exists a truth-value assignment which satisfies an arbitrary Boolean expression, was the first problem shown to be NP-complete (Cook, 1971). The proof that SATISFIABILITY is NP-complete is restated in Garey and Johnson (1979), along with its transformations into 3SAT and ONE-IN-THREE 3SAT. The bulk of this section will show that ONE-IN-THREE 3SAT (Schaefer, 1978) can be transformed into the protein threading decision problem (PRO-THREAD). The transformation of PRO-THREAD into the optimal protein threading problem, the final result of this section, is then immediate. This sequence is illustrated in Figure 1.

The well-known 3SAT problem is a variant of SATISFIABILITY in which each clause in the Boolean expression is constrained to have exactly three literals. The ONE-IN-THREE 3SAT problem is a variant of 3SAT, differing only in that each Boolean clause is required to be satisfied by exactly one of the three literals. It is known to be NP-complete (Schaefer, 1978; Garey and Johnson, 1979, p. 259).

Let $B = \{b_1, b_2, \ldots, b_g\}$ be a collection (a conjunction, i.e. logical AND) of Boolean clauses (disjunctions, i.e. logical ORs) with $|b_i| = 3$, $1 \leq i \leq g$, and let $U = \{u_1, u_2, \ldots, u_h\}$ be the set of Boolean variables mentioned in B.

The decision question is: Does there exist a truth-value assignment for U that satisfies all the clauses in B (i.e. makes the conjunction of all the disjunctions true), and such that each clause in B has exactly one true literal?

PRO-THREAD is NP-complete

Theorem. PRO-THREAD is NP-complete.

Proof. It is easy to verify that PRO-THREAD is in NP since a non-deterministic algorithm need only guess a particular threading and check in polynomial time whether its score is K or less.

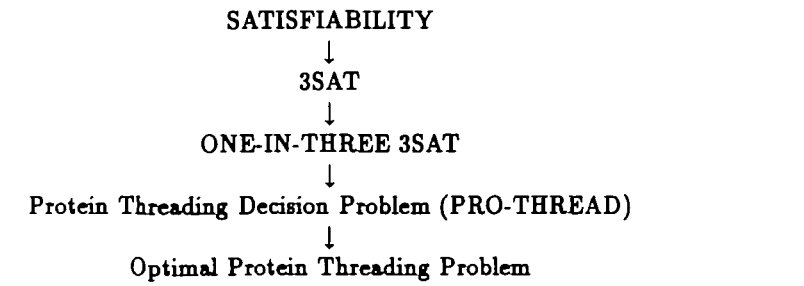


Fig. 1. The sequence of problem transformations we employ.

$$U = \{u_1, u_2, u_3, u_4\}$$
$$B = \{(u_1, \bar{u}_2, u_3), (\bar{u}_2, u_3, \bar{u}_4), (u_1, \bar{u}_3, u_4)\}$$

Fig. 2. Example of the ONE-IN-THREE 3SAT problem.

We transform ONE-IN-THREE 3SAT to PRO-THREAD. Figures 2–5 illustrate the transformation for a simple example.

Let $B = \{b_1, b_2, \ldots, b_g\}$ be a collection of clauses with $|b_i| = 3$, $1 \leq i \leq g$, and let $U = \{u_1, u_2, \ldots, u_h\}$ be the set of Boolean variables mentioned in B. These make up an arbitrary instance of ONE-IN-THREE 3SAT.

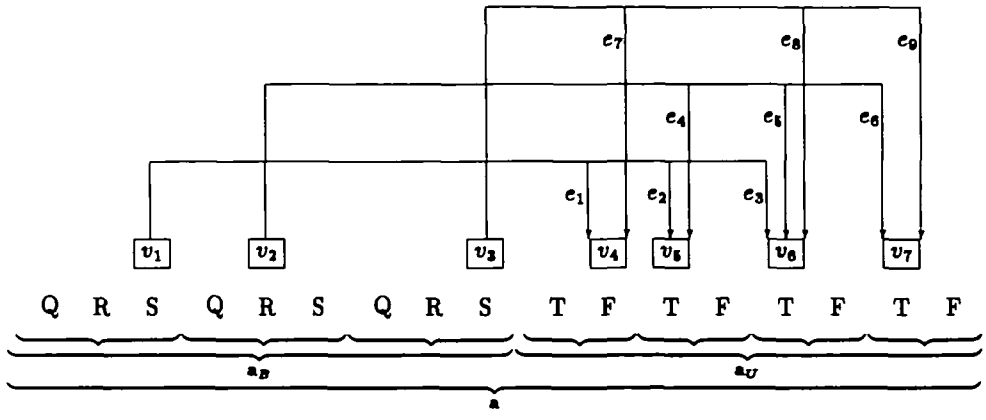
Let $\mathbf{a}$ be an amino acid sequence with $|\mathbf{a}| = 3g + 2h$. Let $\mathbf{a}$ consist of two concatenated subsequences: $\mathbf{a}_B$, with $|\mathbf{a}_B| = 3g$, which corresponds to the clauses and literals of B; and $\mathbf{a}_U$, with $|\mathbf{a}_U| = 2h$, which corresponds to the truth-value assignments of U. $\mathbf{a}_B$ is composed of g repetitions of the subsequence of three amino acids (Q, R, S). Let $\alpha_1 =$ Q, $\alpha_2 =$ R and $\alpha_3 =$ S. Each Q (respectively each R and each S) corresponds to some b_i being satisfied by the first (respectively the second and the third) literal in b_i, $1 \leq i \leq g$. $\mathbf{a}_U$ is composed of h repetitions of the subsequence of two amino acids (T, F). Each T (respectively each F) corresponds to a truth-value assignment of TRUE (respectively of FALSE) to some variable u_i, $1 \leq i \leq h$. Although we have provided one subsequence (Q, R, S) for each clause in B and one subsequence (T, F) for each variable in U, this is only to ensure that there are sufficient amino acids for any possible ONE-IN-THREE 3SAT instance. Nowhere do we restrict particular clauses or variables to particular subsequences, and consequently there generally will be several admissible threadings solving any given ONE-IN-THREE 3SAT instance.

Let $C = \{C_1, C_2, \ldots, C_{g+h}\}$ be a collection of core segments each one element long (consequently, overlapping core segments are not a problem provided they are assigned distinct indices by the threading). Core segment C_i, $1 \leq i \leq g$, corresponds to clause b_i and the choice of whether the first, second or third literal in b_i satisfies b_i. Core segment C_{g+j}, $1 \leq j \leq h$, corresponds to variable u_j and the choice of truth-value assignment to u_j.

Let $V = \{v_1, v_2, \ldots, v_{g+h}\}$ be the vertices of the adjacency graph. Identify vertex v_i with core element $C_{i,1}$, $1 \leq i \leq g + h$. Let $E = \{e_1, e_2, \ldots, e_{3g}\}$ be the set of edges in the adjacency graph. Edges will connect vertices corresponding to clauses of B (tail) to vertices corresponding to variables of U (head). Consequently the resulting graph will be bipartite. If u_i is the variable mentioned in the jth literal z of b_k, then let edge $e_{3(k-1)+j}$ connect vertices v_k (tail) and v_{g+i} (head), and let edge $e_{3(k-1)+j}$ be labeled by (j, N) if z is negated and by

$$U = \{u_1 = False, u_2 = True, u_3 = True, u_4 = True\}$$
$$B = \{(False, False, True), (False, True, False), (False, False, True)\}$$

Fig. 3. Truth-value assignments solving the example ONE-IN-THREE 3SAT problem



e	*tail* (e)	*head* (e)	s(e)
e_1	v_1	v_4	$(1, P)$
e_2	v_1	v_5	$(2, N)$
e_3	v_1	v_6	$(3, P)$
e_4	v_2	v_5	$(1, N)$
e_5	v_2	v_6	$(2, P)$
e_6	v_2	v_7	$(3, N)$
e_7	v_3	v_4	$(1, P)$
e_8	v_3	v_5	$(2, N)$
e_9	v_3	v_7	$(3, P)$

The equivalence between ONE-IN-THREE 3SAT clauses and variables and their corresponding PRO-THREAD vertices is

$$\{(b_1, v_1), (b_2, v_2), (b_3, v_3), (u_1, v_4), (u_2, v_5), (u_3, v_6), (u_4, v_7), \}.$$

Fig. 4. PRO-THREAD problem equivalent to the example ONE-IN-THREE 3SAT problem.

$$\mathbf{t} = (3, 5, 9, 11, 12, 14, 16)$$

The threading t corresponds to the placement of v_i in Figure 4. The truth-values in Figure 3, which solve the ONE-IN-THREE 3SAT problem in Figure 2, can be read directly from this threading. Note that other threadings (e.g., $(3, 8, 9, 11, 12, 14, 16)$) would yield the same truth-values.

Fig. 5. Threading solving the equivalent PRO-THREAD problem.

(j, P) if positive, $1 \leqslant j \leqslant 3$ and $1 \leqslant k \leqslant g$. Let all vertex labels be null.

Let the score functions σ_v and σ_l (and hence f_v and f_l) be identically 0. Let $\sigma_e(a, b, d)$ be defined by:

$\sigma_e(a, b, d) =$

0, if $d = (j, P)$ and $a =$ T and $b = \alpha_j$ and $1 \leqslant j \leqslant 3$,
0, if $d = (j, P)$ and $a =$ F and $b \in \{\alpha_1, \alpha_2, \alpha_3\}$ and $b \neq \alpha_j$ and $1 \leqslant j \leqslant 3$,
0, if $d = (j, N)$ and $a =$ T and $b \in \{\alpha_1, \alpha_2, \alpha_3\}$ and $b \neq \alpha_j$ and $1 \leqslant j \leqslant 3$,
0, if $d = (j, N)$ and $a =$ F and $b = \alpha_j$ and $1 \leqslant j \leqslant 3$,
1, otherwise. (5)

σ_e corresponds to the constraint that each clause in B be satisfied by exactly one literal. It is non-0 whenever $a \notin$ {T, F}, $b \notin$ {Q, R, S} or d is not a valid edge (environment) label. If $d = (j, P)$, corresponding to the jth literal being positive, then a score of 0 corresponds to either: (i) $b = \alpha_j$ (selecting the jth literal to satisfy the clause) and $a =$ T (the variable it mentions is true); or (ii) $b \neq \alpha_j$ and $a =$ F (the literal is not selected and the mentioned variable is false). Symmetric remarks apply if $d = (j, N)$, corresponding to the jth literal being negated. The 'otherwise' term ensures a non-zero score if any of these conditions are not met.

Hence the score function f is:

$$f(\mathbf{t}) = \Sigma_{e \in E} f_e(e, \mathbf{t}) \quad (6)$$
$$= \Sigma_{e \in E} \sigma_e(\pi(head\ (e), \mathbf{t}), \pi(tail\ (e), \mathbf{t}), s(e). \quad (7)$$

The decision question is: Does there exist a t such that $f(\mathbf{t}) \leqslant 0$? We show that this is equivalent to the original ONE-IN-THREE 3SAT problem.

($\Rightarrow$) By construction, if the original ONE-IN-THREE 3SAT

problem has a solution then the exhibited threading problem has a threading with a score of 0. Let the variables in U be assigned the solution truth values. We will show that $f(\mathbf{t}) = 0$ where $\mathbf{t} = (t_1, t_2, \ldots, t_{g+h})$ and:

$t_i =$

$3(i-1) + l$, if $1 \leq i \leq g$ [where $l \in \{1, 2, 3\}$ is the index of the unique literal which was satisfied in b_i], (8)

$3g + 2(i-g) - 1$, if $g < i \leq g + h$ and $u_{i-g} =$ TRUE,

$3g + 2(i-g)$, if $g < i \leq g + h$ and $u_{i-g} =$ FALSE.

Consider an arbitrary vertex, v_k, such that $1 \leq k \leq g$. By construction, this corresponds to b_k. Let $l \in \{1, 2, 3\}$ index the unique literal satisfying b_k under the truth values assigned to U. Since $t_k = 3(k-1) + l$ by assumption, $\pi(v_k, \mathbf{t}) = \alpha_l$ by the structure of $\mathbf{a}_B$.

By construction, v_k is the tail of exactly three edges. Consider an arbitrary edge e such that $v_k = tail\ (e)$. By construction, $e = e_{3(k-1)+j}$ for some $1 \leq j \leq 3$. Let z be the jth literal of b_k and u_i the variable mentioned by z. By construction, $head\ (e) = v_{g+i}$. Four cases arise depending on whether z is negated and whether $j = l$. By exhaustive case analysis,

z negated?	$j = l$?	t.v. of u_i	t_{g+i}	$s(e)$	$\pi(v_{g+i}, \mathbf{t})$	$\pi(v_k, \mathbf{t}) = \alpha_j$?	σ_e
no	yes	TRUE	$3g + 2i - 1$	(j, P)	T	yes	0
no	no	FALSE	$3g + 2i$	(j, P)	F	no	0
yes	no	TRUE	$3g + 2i - 1$	(j, N)	T	no	0
yes	yes	FALSE	$3g + 2i$	(j, N)	F	yes	0

Under all cases $\sigma_e(e, \mathbf{t}) = 0$, as seen from Equation 5. Since e was arbitrary, all edges whose tail is v_k must score 0. Since v_k was an arbitrary vertex such that $1 \leq k \leq g$, and by construction only such vertices are the tails of edges, all edges must score 0. Since the vertex and loop scores were identically 0, we must have $f(\mathbf{t}) = 0$.

($\Leftarrow$) Conversely, if the exhibited threading problem has a threading $\mathbf{t}$ with $f(\mathbf{t}) = 0$ then a set of truth values for U solving the original problem can be read directly from $\pi(v_{g+i}, \mathbf{t})$, $1 \leq i \leq h$. This assignment sets:

$u_i =$

TRUE, if $\pi(v_{g+i}, \mathbf{t}) =$ T, (9)

FALSE, otherwise.

Let $\mathbf{t}$ be a threading such that $f(\mathbf{t}) = 0$. We will show that this assignment solves the original ONE-IN-THREE 3SAT problem.

Let b_k be an arbitrary clause from B. By construction, it corresponds to vertex v_k. Let $E_k = \{e | tail\ (e) = v_k\}$ and order E_k according to the order of the subscripts i for $e_i \in E_k$ (i.e. according to the order of the literals in b_k). By construction, $|E_k| = 3$. We must have $\pi(v_k, \mathbf{t}) = \alpha_l$ for some $1 \leq l \leq 3$, as otherwise $f_e(e, \mathbf{t}) = 1$ for every edge $e \in E_k$, contradicting the assumption that $f(\mathbf{t}) = 0$. Likewise, we must have $\pi(head\ (e), \mathbf{t}) \in \{T, F\}$ for every edge $e \in E_k$.

Consider the jth edge $e \in E_k$. By construction, $e = e_{3(k-1)+j}$. Let u_i be the variable mentioned by the jth literal z in b_k. Again, four cases arise depending on whether z is negated and whether $j = l$. By exhaustive case analysis, and noting that the values shown for $\pi(v_{g+i}, \mathbf{t})$ follow from Equation 5 and the assumption that $f(\mathbf{t}) = 0$,

z negated?	$j = l$?	$s(e)$	$\pi(v_k, \mathbf{t}) = \alpha_l$?	$\pi(v_{g+i}, \mathbf{t})$	t.v. of u_i	t.v. of z
no	yes	(j, P)	yes	T	TRUE	TRUE
no	no	(j, P)	no	F	FALSE	FALSE
yes	no	(j, N)	no	T	TRUE	FALSE
yes	yes	(j, N)	yes	F	FALSE	TRUE

It is easy to see that z is true exactly when $l = j$. By construction this is true for exactly one edge in E_k, and consequently b_k has exactly one true literal under this truth value assignment.

($\Leftrightarrow$) It is easy to verify that the transformation can be accomplished in polynomial time. Consequently, the original ONE-IN-THREE 3SAT problem is solvable in polynomial time if and only if the corresponding PRO-THREAD problem is. Because ONE-IN-THREE 3SAT is known to be NP-complete, PRO-THREAD is NP-complete. ■

Remark. In fact, the proof shows that PRO-THREAD is NP-complete in the strong sense, since the only numbers used are 0 and 1.

The optimal protein threading problem

The optimal protein threading problem is to find the minimum scoring threading of $\mathbf{a}$ into C, given the other parameters as described above.

Corollary. If the optimal protein threading problem has a polynomial time solution then so does PRO-THREAD.

Proof. If the optimal protein threading problem has a polynomial time solution, then we can find the minimum scoring threading in polynomial time. This immediately answers the decision problem of whether a threading exists with a score of K or less. ■

Remark. This corollary shows that the optimal protein threading problem is NP-hard.