

Topics in Computational Molecular Biology

March 2001

Lecturer: Mona Singh ¹

Protein Motif Recognition I

Introduction

One of the most important problems in molecular biology is the protein structure prediction problem: given the one-dimensional amino acid sequence that specifies a protein, what is the protein's fold in three dimensions? This problem is critical, since the structure or fold of a protein provides the key to understanding its biological function, and proteins play a variety of important roles in the body (*e.g.*, as enzymes, antibodies, etc.). Proteins may also be associated with particular human diseases, and thus, understanding protein structure may be used to better understand these diseases and to do rational drug design.

Unfortunately, determining the three-dimensional fold of a protein is very difficult. Experimental approaches such as nuclear magnetic resonance (NMR) and X-ray crystallography are expensive and can take a long time (sometimes longer than a year). As a result, there is a large gap between the number of known protein sequences and the number of known three-dimensional protein structures. This gap has grown over the past decade (and is expected to keep growing) as a result of the various worldwide genome projects. Thus, computational methods which may give some indication of structure and/or function are becoming increasingly important.

Levels of Protein Structure

The structure of a protein is characterized not only by its amino acid sequence and its full three-dimensional structure, but also by some intermediate levels. The following structures are in increasing order of complexity.

- *Primary structure*: the linear amino acid sequence of a protein.

¹This lecture is adapted from earlier lectures given by Bonnie Berger and myself. Scribe notes are adapted from notes taken by Casim Sarkar when I lectured at MIT.

- *Secondary secondary*: the local regular structures commonly found within proteins. These include α -helices and β -sheets. (See Figures 0.1 and 0.2.) Often amino acid residues in a particular protein structure that are not part of either an α -helix or a β -sheet are put into a catch-all “other” category.
- *Super-secondary structure (or motif)*: local folding patterns built up from particular secondary structures (*e.g.*, the EF-hand motif consists of an α -helix, followed by a turn, followed by another α -helix).
- *Tertiary structure*: the full three-dimensional structure of a protein.
- *Quaternary structure*: the arrangement of several protein subunits in space.

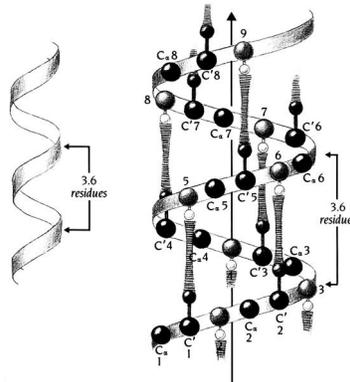


Figure 0.1: An α -Helix; taken from *Introduction to Protein Structure* by Branden and Tooze (1991)

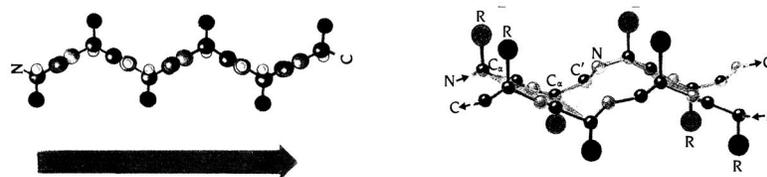


Figure 0.2: A β -Strand and antiparallel strands in a β -sheet; taken from *Introduction to Protein Structure* by Branden and Tooze (1991)

Types of Computational Problems

1. How can one try to predict the three-dimensional fold of a protein (either the exact or overall fold)? There are many approaches to this problem. Here we give just a few:
 - Model all the energetics involved in protein folding, and try to find the structure with lowest free energy. This is a very difficult problem, both in terms of the modeling as well with the searching of the vast conformational space.
 - Exploit high sequence similarity and use alignments. Two sequences that have just 25% sequence identity usually have the same the overall fold. Alignments are probably the most widely used tool for getting an idea of a protein's 3D structure; however, they are useful only when there are similar protein sequences for which structural information is known.
 - Use the threading approach. This approach starts with the observation that many protein structures have similar folds, and assumes that there are just a limited number of distinct folds. Then, for a protein sequence, the goal is to find the known protein structure which "best fits" it according to some statistics-based potential function.
2. Since predicting the three-dimensional structure of a protein is difficult, many researchers have focused on trying to predict the secondary structure of a protein. That is, for each amino acid in a protein, can you predict whether the amino acid is in an α -helix, β -sheet, or neither? Unfortunately, predicting the secondary structure of a protein is also a very difficult problem, perhaps because the secondary structure depends on the overall three-dimensional structure of the fold. For example, there are 5-long amino acid subsequences that are known to occur in helices in one protein, and in sheets in another. Most methods for secondary structure prediction are statistical in nature, and the overall three-state prediction accuracy is about 70%.
3. Can you recognize protein structural motifs? The *structural motif recognition* problem is: given a particular structural motif, determine if it occurs in a given amino acid sequence, and if so, in what positions. Protein structural motifs are made up of particular secondary structure units, often in particular conformations. Thus, they can often have a regular structure which makes them amenable to computer-based methods.

Recognizing the Coiled Coil Structure

The rest of this lecture will be devoted to the third question and will specifically address coiled coils, although the techniques can be extended to recognizing other motifs. Our focus will be on the following question: given a subsequence of amino acid residues, does it fold into a coiled coil? Determining experimentally whether a subsequence folds into a coiled coil can be quite time-consuming. Thus, ideally, we would like a reliable computational method to predict whether a subsequence of amino acid residues folds into a coiled coil. These predictions can then be verified in the laboratory.

A coiled coil is a structural motif that is found in fibrous proteins such as those making up hair and skin, in several DNA-binding proteins, and in many viral membrane fusion proteins. It consists of two or more α -helices wrapped around each other with a slight left-handed superhelical twist. The amino acid sequences making up each helix can either be identical (homo-oligomers) or distinct (hetero-oligomers). Coiled coils have a characteristic heptad repeat unit (see Figure 0.3). Two turns of the helix correspond to the seven positions of the coiled coil: *a*, *b*, *c*, *d*, *e*, *f*, and *g*. Residues in the *a* and *d* positions are buried in the core of the coiled coil (between the helices making up the coiled coil); these are typically hydrophobic residues. Predominantly charged residues are found in the *e* and *g* positions.

Methods for recognizing coiled coils fall into the following framework:

1. Collect a database of known coiled coils and available amino acid subsequences.
2. Devise a method to determine whether the unknown sequence shares enough distinguishing sequence features with the known coiled coils to be considered a coiled coil.

Single frequency approach

The first methods for recognizing coiled coils looked at the single frequencies of each amino acid residue [5, 3]. This exploits the fact that in some of the positions in a coiled coil, certain residues are more likely to occur than others.

This approach examines the frequency of each residue in each position in a coiled coil. We can build a table from the protein database that represents the relative frequency of each amino acid in each position. That is, we have a table entry for each amino acid/coiled coil position pair. For example, for leucine and position *a*, the entry in

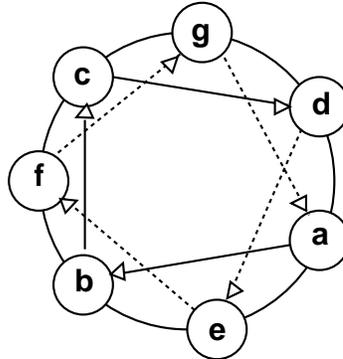


Figure 0.3: Top view of a single strand of a coiled coil. Each of the seven positions $\{a, b, c, d, e, f, g\}$ corresponds to the location of an amino acid residue which makes up the coiled coil. The arrows between the seven positions indicate the relative locations of adjacent residues in an amino acid subsequence. The solid arrows are between positions in the top turn of the helix, and the dashed arrows are between positions in the next turn of the helix.

the table is the percentage of position a 's in the coiled coil database which are leucine, divided by the percentage of residues in Genbank (a large protein sequence database) which are leucine. For example, if the percentage of position a 's in the coiled coil database which are leucine is 27%, and the percentage of residues in Genbank which are leucine is 9%, then the table entry value for the pair leucine and position a is 3. Intuitively, this table entry represents the "propensity" that a leucine residue is in position a in a coiled coil.

This approach actually looks at 28-long windows, since stable coiled coils are believed to be at least 28 residues long. Thus for each residue, it looks at each possible position (a through g), and at all 28-long windows that contain it. It then calculates the relative frequencies for each residue in the window. If the product of the relative frequencies for each residue in the window is greater than some threshold, we conclude that the residue is part of a coiled coil. Overall, the single-frequency method does rather well. It has been implemented as the COILS program [3] and is widely used. One weakness of the method is that it tends to over-predict the number of coiled coils; that is, it has a significant false positive rate.

Probabilistic Framework

It is possible to state the coiled coil recognition problem within a probabilistic framework, and to use this in methods exploiting higher order dependencies within coiled coil structures; this has led to coiled coil recognition methods with lower false positive rates [1, 2]. Before we give the algorithm, we give the probabilistic framework.

Given a subsequence $z = r_1, r_2, \dots, r_{28}$, is z a coiled coil?

We will estimate $\Pr[z \in C]$, where C is the class of coiled coils. Note that this is not a probability *per se*, since either z is a coiled coil or not. However, for convenience, we will look at this as a probability. If the estimate is large, then we will conclude that z is a coiled coil; otherwise, we will conclude that z is not a coiled coil.

Let $X = R_1, R_2, \dots, R_{28}$ be a random subsequence selected from Genbank. Then,

$$\begin{aligned} \Pr[z \in C] &= \Pr[X \in C | X = z] \\ &= \Pr[X \in C | (R_1 = r_1) \wedge (R_2 = r_2) \wedge \dots \wedge (R_{28} = r_{28})] \\ &= \frac{\Pr[(X \in C) \wedge (R_1 = r_1) \wedge \dots \wedge (R_{28} = r_{28})]}{\Pr[(R_1 = r_1) \wedge \dots \wedge (R_{28} = r_{28})]} \end{aligned}$$

Using repeated applications of the definition of conditional probability, we can expand the numerator of this expression (*i.e.*, $\Pr[(R_1 = r_1) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)]$):

$$\begin{aligned} &= \Pr[R_1 = r_1 | (R_2 = r_2) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)] \\ &\quad \cdot \Pr[(R_2 = r_2) | (R_3 = r_3) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)] \\ &\quad \vdots \\ &\quad \cdot \Pr[R_{28} = r_{28} | X \in C] \cdot \Pr[X \in C] \\ &= \prod_{i=1}^{27} \Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)] \\ &\quad \cdot \Pr[R_{28} = r_{28} | X \in C] \cdot \Pr[X \in C] \end{aligned}$$

The denominator can be expanded in a similar manner.

To estimate these probabilities, we start making assumptions. We can naively assume that the residues are independent of each other:

$$\Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)] = \Pr[R_i = r_i | X \in C]$$

and

$$\Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge \dots \wedge (R_{28} = r_{28})] = \Pr[R_i = r_i].$$

If we simplify the previous equations with this assumption, then the resulting formula gives the product of the single frequencies. That is, we get the single frequency-approach.

Of course, in reality, we do not expect the residue probabilities to be completely independent. For example, we would expect that a residue is influenced by its neighboring residues. So a better assumption is that only adjacent positions in the sequence are dependent:

$$\begin{aligned} \Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)] \\ = \Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge (X \in C)] \end{aligned}$$

Here we are assuming that dependencies are captured by adjacency in the sequence. There can be other dependencies further on in the sequence, but we are assuming that whatever dependencies exist can be captured by neighboring residues. Using this assumption (and the analogous assumption for the probabilities in the denominator), and some simplification, we get:

$$\begin{aligned} \Pr[z \in C] &= \Pr[X \in C] \\ &\cdot \frac{\prod_{i=1}^{27} \Pr[(R_i = r_i) \wedge (R_{i+1} = r_{i+1}) | (X \in C)]}{\prod_{i=2}^{27} \Pr[R_i = r_i | X \in C]} \\ &\cdot \frac{\prod_{i=2}^{27} \Pr[R_i = r_i]}{\prod_{i=1}^{27} \Pr[(R_i = r_i) \wedge (R_{i+1} = r_{i+1})]} \end{aligned}$$

This is better than the naive assumption of complete independence between the residues. However, we can make even better assumptions. Namely, we assume that the probability of a residue depends upon *several* neighboring residues, and the nature of this dependence is determined by the motif structure. For example, for the coiled coil motif, a residue in position i is near the next residue in the sequence but is also near residues in positions $i + 3$ and $i + 4$, because these positions come back in the three-dimensional structure and are near position i . For example, position a is near positions b , d , and e (see Figure 0.3). Thus the following is a better assumption for coiled coils:

$$\begin{aligned} \Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (X \in C)] \\ = \Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge (R_{i+3} = r_{i+3}) \wedge (R_{i+4} = r_{i+4}) \wedge (X \in C)] \end{aligned}$$

We can again plug this in, but now the equation is much more complicated. Although this is a better assumption than the others, we still have a problem: there are terms with 4-tuples, and the table gets big ($(7 \cdot 20)^4$ entries). Moreover, we do not have enough data for this assumption, so many of the entries are 0. As a result, we try to

capture this by assuming functions over pairwise dependencies. Namely, we assume that:

$$\begin{aligned} \Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge \dots \wedge (R_{28} = r_{28}) \wedge (x \in C)] \\ = f(\Pr[R_i = r_i | (R_{i+1} = r_{i+1}) \wedge (X \in C)], \\ \Pr[R_i = r_i | (R_{i+3} = r_{i+3}) \wedge (X \in C)], \\ \Pr[R_i = r_i | (R_{i+4} = r_{i+4}) \wedge (X \in C)]) \end{aligned}$$

where the function f can be a weighted average, a minimum, a maximum, or a product.

These assumptions can now be used to estimate $\Pr[z \in C]$. We can estimate these probabilities because there is enough data to make a pairwise table. Experimentally, it was determined that the taking the average over pairwise probabilities worked the best.

This method works very well. In fact, if it is run on the PDB, the protein database of solved structures, it removes all false positive results. PairCoil was later extended to the domain of three-stranded coiled coils using multidimensional clustering in the MultiCoil program [8]. This program is also able to distinguish 2- and 3-stranded coiled coils. The LearnCoil-Histidine Kinase program [7] was written to predict coiled-coils in histidine kinase linker domains. LearnCoil-VMF [6] was developed to predict coiled coils in viral membrane-fusion proteins. This program was used to identify coiled coils in many diverse viral membrane fusion proteins, including those of many retroviruses (e.g., human T-cell leukemia virus, HIV, Visna), paramyxoviruses (e.g., parainfluenza viruses, mumps) and filoviruses (e.g., Ebola). Several of these predictions have since verified. For example, the predicted regions of Visna virus were synthesized and used to set crystal trays, and X-ray crystallography revealed the predicted structure [4].

Window based algorithm

We have talked about the probabilistic approach for deciding whether a 28-residue subsequence is a coiled coil. Now we will consider a longer sequence of some arbitrary length n , and determine where, if at all, coiled coils occur in the sequence. One way of doing this is to look at every contiguous subsequence of length $w = 28$ and to run the test we just described to determine which subsequences are likely to be coiled coils. The running time of such an algorithm would naively be $O(pwn)$ where p is the period of the motif ($p = 7$ for the coiled coil motif). That is, the most naive way is to look at all continuous subsequences of length w and run the test we just

described to see which subsequences are likely to be coiled coils. The running time of this method is $O(pwn)$, because it needs $O(w)$ steps per period per window, and there are p periods per window and $O(n)$ windows total.

Actually, we want to compute a score for each position in the sequence. We can define the score of each position j as the maximum, over all windows \mathcal{W} of size w which contain residue j and over all possible periods p , of the probability that window \mathcal{W} with period p is a coiled coil.

Now, we show how to compute the score of each position in $O(pn)$ time.

We define $score(j)$ as the score of the w -long window ending at position j . After we have computed the score at position $j - 1$ (*i.e.*, $score(j - 1)$), the score of position j can be computed from $score(j - 1)$ by adding the contribution of position j and subtracting the contribution of position $j - w$. For each position, the score is computed for all periods p , so it takes $O(pn)$ steps total to compute all the window scores.

To compute the *maximum* window score $M(j)$ for each residue j (*i.e.*, the maximum score of any window that contains residue j), we can divide the sequence into $\lceil n/w \rceil$ contiguous blocks of size w (except the last block, which will be of size $\leq w$). Then we define two scores: $N_L(j)$ is the maximum $score(k)$, where k ranges from the beginning of j 's partition block to position j and $N_R(j)$ is the maximum $score(k)$ where k ranges from the end of j 's partition block to position j . $N_L(j)$ (and $N_R(j)$) can be computed by a single left-to-right (right-to-left) segmented prefix over the sequence, starting a new maximum each time a barrier of a partition block is crossed. It takes $O(n)$ time to compute these values for each residue. It can be shown that

$$M(j) = \max\{N_R(j), N_L(j + w - 1)\},$$

and these values can also be computed in $O(n)$ time. Thus, the total time to compute scores for each residue is $O(pn)$.

References

- [1] Bonnie Berger. "Algorithms for protein structural motif recognition." *Journal of Computational Biology*, volume 2, pages 125–138, 1995
- [2] Bonnie Berger, David B. Wilson, Theodore Tonchev, Mari Milla, and Peter S. Kim. "Predicting coiled coils using pairwise residue correlations." *Proceedings of the National Academy of Sciences*, volume 92, pages 8259–8263, 1995.

- [3] A. Lupas, M. van Dyke, and J. Stock. “Predicting coiled coils from protein sequences.” *Science*, 252:1162–1164, 1991.
- [4] V. Malashkevich, M. Singh and P. S. Kim. “The trimer-of-hairpins motif in viral membrane-fusion proteins: Visna virus.” *Proceedings of the National Academy of Sciences*, 98: 8502–8506, 2001.
- [5] D. A. D. Parry. “Coiled coils in alpha-helix-containing proteins: analysis of residue types within the heptad repeat and the use of these data in the prediction of coiled-coils in other proteins.” *Bioscience Reports*, 2:1017–1024, 1982.
- [6] M. Singh, B. Berger and P. S. Kim. “LearnCoil-VMF: Computational evidence for coiled coil-like motifs in many viral membrane-fusion proteins.” *Journal of Molecular Biology* 290: 1031–1041, 1999.
- [7] M. Singh, B. Berger, P. S. Kim, J. Berger and A. Cochran. “Computational learning reveals coiled coil-like motifs in histidine kinase linker domains.” *Proceedings of the National Academy of Sciences* 95: 2738–2743, 1998.
- [8] E. Wolf, P. S. Kim and B. Berger. “MultiCoil: A Program for Predicting Two- and Three-Stranded Coiled Coils.” *Protein Science* 6: 1179–1189, 1997.