The molecular structure of a protein can be broken down hierarchically. The primary structure of a protein is simply its sequence, the secondary structure is its localized folding, its tertiary structure is the long-range domain, its quaternary structure is its multimeric organization (if it is made up of multiple peptide chains), and its supramolecular structure is the global assembly. A significant and challenging problem in computational biology is coming up with algorithms which predict one or more of these types of folds.

1 Methods of Predicting Protein Structure

Experimental methods, such as X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy, are generally very effective. However, they have drawbacks that make it a relatively inefficient technique. For instance, such methods are very expensive, time consuming, and not robust (only some proteins can be crystallized). Also, NMR spectroscopy enables the observation of structural and dynamic properties of proteins, but is limited in the size of the protein.

Computational methods, at this point, are relatively unrefined. Ab initio predictions are structure predictions based only on the sequence of the protein in question, utilizing the fundamental principles of a protein fold, such as the geometric, kinematical and energy properties of a the molecule. These methods have not been very successful to date. The most successful ab initio method is currently Rosetta. This method functions by breaking down folds into subsequence folds, and then searching a library of structural motifs that may match. The entire fold is then evaluated by applying an energy function. The rationale behind this method is that local structures often fold independently of the full protein. Such a method is unsatisfying biologically, because it does not use our understanding of how proteins fold.

Homology prediction compares the protein sequence to existing sequences in the PDB with a known 3-Dimensional structure to determine the evolutionary, structural, and
functional changes. Once the structurally known homologous protein is determined, local refinements of the protein is made in order compute the full structure of the unknown protein.

2 Threading

2.1 Algorithm

Threading is another experimental method of protein structural prediction, which combines methods of homology comparison and molecular modeling. One of the differences between homology modeling and threading is that the latter computes an energy function during the sequence alignment. Threading first takes a protein sequence and a library of templates or known structures as shown in Figure 2. These are PDB structures and, ideally, the library contains all known protein folds. The collection of templates is a search space of all possible alignments of the protein. In order to prevent a statistical skew in favor of large protein families for finding an ideal match, databases such as SCOP, CATH, and FSSP are used to remove pairs of proteins with highly similar structures, thus normalizing the number of matches for each target.

![Figure 2](image)

We then take the sequence and "thread" it through each template in our library (Figure 3). Thus, we drag the sequence ACDEFG through each location on each template (we are really just searching for the best composition of the sequence as measured by the energy function). In the third alignment shown in Figure 3, the protein sequence is aligned so that it excludes part of the template. This problem of determining the best arrangement of residues counting such insertions and gaps is based on the scoring of structure to sequence alignment. Once all of the candidate structures together with their
scores are gathered, the lowest energy one is accepted as the structure prediction of the protein. The energy function is expressed as a weighted sum of how preferable it is to put to residues nearby, taking into effect steric clashes and hydrophobic groupings ($E_p$), the alignment gap penalty ($E_g$), the compatibility with local secondary structure prediction ($E_{ss}$), how well a residue fits its structural environment ($E_s$), and how often a residue mutates to the template residue ($E_m$).

![Figure 3](image)

The way in which we represent template structure is a contact graph. Contact graphs entail abstracting each residue to a point. Core elements are short subsequences, usually formed by alignments in helices or sheets, where we allow no alignment gaps. Cores are then connected by contacts represented in the graph. Consequently, contact graphs reduce the search space of possible threads.

Original Contact Graph

No gap allowed within cores

Simplified Contact Graph

Sequence
2.2 Difficulties

There are a significant number of computational problems associated with such an algorithm. For instance, the final prediction depends on the size and details of the initial library. It is convenient to have a small library, as calculations involved with threading are often slow, and a small library will reduce computational time. However, threading score functions are imperfect, so the closer a template is to the correct structure, the more likely a sequence is going to score well. Ideally, then, we should have large libraries to increase such a likelihood. Usually, certain groups will have template libraries from about 500 to 5000 members and there is no way to determine the optimal size of the library.

There are also difficulties with the representation of templates as contact graphs. Since alignment gaps are confined to the connecting non-core loop regions, the lengths of the connection loops are variable. As a result we have an exponentially large search space of possible threadings. When we search for such a sequence-structure alignment, and allow for variable gap lengths and interactions between neighboring amino acids, then finding the globally optimal threading is NP-hard. Thus, any algorithm takes an amount of time that is exponential in protein size. This can be proved when we consider that threading is at least as difficult as the MAX-CUT problem. This problem asks that, given a graph $G = (V, E)$, find a cut $(S, T)$ of $V$ with maximum number of edges between $S$ and $T$.

The mapping between MAX-CUT and threading is very similar. We start with a graph for which we want to apply MAX-CUT, and consider it a contact graph for a structure. Thus, we make a protein structure with $2n$ amino acids for a total of $n$ nodes, where each amino acid is represented by either a 0 or a 1. If we drop all of the energy terms except for $E_p$, and we say that each node labeled (0,1) or (1,0) gets a score of 1, the maximizing the threading score is maximizing number of 0’s next to 1’s. This is the same as finding the maximum cut of the graph.
3 Threading Programs

3.1 Branch and Bound

The branch and bound method is an exact method for searching for an optimal alignment, but it takes exponential time in the size of the protein. The algorithm functions by assuming that each solution can be partitioned into subsets, and that the upper limit on a subset’s solution can be computed quickly. In the diagram, each circle illustrates the space of possible threadings, the solid lines indicate partitions made in a previous step, and dashed lines indicate partitions made in the current step. Furthermore, numbers indicate lower bounds for newly created subsets, and arrows indicate the set that was partitioned. Branch and bound selects the subset with the best possible bounds, subdivides it, and computes a bound for each subsequent subset.

The aspects of this search which determine its efficiency are how the lower bound for the set of possible threadings is computed, and how the threading set is partitioned into subsets. Ideally, the lower bound should take into consideration the interaction of the set with the preceding set, and the best interaction with other sets. A threading set is partitioned by selecting a core segment, and choosing a split point in the set.

3.2 RAPTOR

Currently, the best protein threading program is probably the RAPTOR. This a linear function of a given number of variables subject to linear constraints, which are typically inequalities. This is a problem that is evidently solvable in polynomial time. We then solve the problem with the addition of integer constraints on the variables.
Raptor uses integer linear programming to minimize the protein energy function based on the constraints. In the constraints, the variable \( x(i,k) \) denotes that core \( i \) is aligned to sequence position \( k \), \( y(i,k,j,l) \) denotes that core \( i \) is aligned to position \( k \) and core \( j \) is aligned to position \( l \), \( D(i) \) gives all of the positions to which core \( i \) can be aligned, and \( R(i, j, k) \) is the set of possible alignments of core \( j \) given that core \( i \) aligns to position \( k \).

In RAPTOR’s energy function, only interactions between residues in the cores are considered, as it is generally accepted that interactions involving loop residues can be ignored since they contribute very little to fold recognition.

\[
\begin{align*}
\text{Minimize} \\
E &= W_w E_w + W_e E_e + W_p E_p + W_g E_g + W_s E_s
\end{align*}
\]

\[
\begin{align*}
\text{s.t.} \\
x_{(i,j)} + x_{(i+1,k)} &\leq 1, k \not\in R[i,i+1,l] \\
y_{(i,j,k,l)} &\leq x_{i,k}, k \in R[i,i,l] \\
y_{(i,j,k,l)} &\leq x_{j,k}, l \in R[j,i,k] \\
y_{(i,j,k,l)} &\geq x_{i,k} + x_{j,k} - 1 \\
\sum_{i \in D(l)} x_{i,l} &= 1 \\
x_{i,j}, y_{(i,j,k,l)} &\in \{0,1\}
\end{align*}
\]

Raptor then relaxes integer constraints to linear constraints, so that \( x \) and \( y \) between 0 and 1. Subsequently, a standard linear method, such as IBM’s Optimization and Solution Library is utilized to solve the linear problem. If the resulting solution is integral, then the program is complete. The problem is that if the solution is not integral, then we get fractional placements of each core. As a solution, we heuristically select a non-integral variable and generate two sub-problems by setting the variable to 0 and 1. We then solve each sub-problem using Branch and Bound. In practice, though, we do not usually this last step, because almost all (99%) of solutions are integral.
3.3 Results

According to benchmark testing, such as CAFASP3, RAPTOR does relatively well in structural prediction. Overall, though, threading is losing ground in competitions because ab initio methods are becoming better for unknown proteins and homology predictions are getting increasingly better as protein databases are expanding.

4 Specialized Prediction Methods

4.1 Predicting Motifs

Some prediction algorithms seek to identify structural motifs within proteins. For instance, a coiled coil, which is doubly twisted helix, contains 3.5 amino acids per turn instead of 3.6, causes helix to bend around itself. This structure is very strong, and is typically used by viruses to penetrate the cell wall. Other unique motifs are zinc fingers, helix-loop-helix structures, beta helices, beta trefoils, and beta barrels.

Because the coiled coil sequence is identical on left and right side, it is computationally easy to identify. The algorithm NewCoils computes the propensity of each amino acid to occupy each position and applies weight matrix. Similarly, PairCoils computes the pairwise probabilities of the neighboring positions in the structure. Each residue is given a score which is based on the likelihood that this residue is in a coiled coil. In order to compute this score, the maximum window score over the 28 residue windows containing the amino acid is determined.

A similar template method, BetaWrap, can predict beta helices. This program scores sequences based on the compatibility with the right-handed beta-helix fold. It incorporates residue pair preferences taken from beta sheets in non-beta helices to create and score potential wraps of a given sequence into a beta-helical structure using the hydrophobicity property. The program then returns the best scoring parses of the sequence into multiple rungs.

Motif recognition is useful as an intermediate step toward full structure prediction, because it is more compliant with sequence analysis, and because it offers useful information about the function of the protein.
4.2 Predicting Secondary Structure
Secondary structure is generally harder to predict than finding motifs because of the relative lack of constraints. The best algorithms usually use Neural Networks. For instance, PSIPRED generates a profile based on the given sequence, and then passes it to a pre-trained Neural Network, which determines whether the sequence folds into an alpha helix, beta sheet, or loop. The algorithm starts with a database of predetermined folds, and removes structural redundancies within proteins, leaving 187 proteins. This program performs with an accuracy of 76%, and is recognized as the best such algorithm.

References

Diagrams and some descriptions were obtained from:
CS273 Lecture Slides 15.


