

Lecture 16: Protein Structure Prediction

CS 273 - Algorithms for Structure and Motion in Biology
Stanford University

Instructors: Profs. Serafim Batzoglou, Jean-Claude Latombe
31 May 2006

Scribe: Christopher Egner

1. Performance of Structure Prediction Methods

There are four major classes of algorithms for the prediction of proteins structure. Homology based methods work from the assumption that proteins with shared ancestry will have mutually conserved sequence and structure. The objective is to identify homologous proteins with known structures and to use these similar structures to predict the structure for an unknown protein. This is done using template elements analogous to building blocks, such as Legos. Correspondence information is derived from primary structure (i.e. sequence) similarity. Accordingly, homology based algorithms are the most reliant of the four classes, generally requiring a database of known proteins with at least 30% sequence similarity and that provide at least 90% template coverage. Since homology algorithms work closely with similar, known natural structures, they also tend to have the best accuracy, producing predictions with an RMSD on the order of 1 - 3Å. They also tend to be the fastest and easiest to implement with running times on the order of seconds.

The second class of algorithms does not use sequence similarity to establish and instead attempts to establish correspondence between the unknown and the known structures by recognising commonalities in the folds. While this relaxes the algorithms' needs in terms of sequence identity (generally 20 - 30% is the minimum) and template coverage (reduced to 75%), this is paid for with losses in accuracy and computation time. Accuracy degrades in a general range of 2 - 5Å and computation time is on the order of minutes instead of seconds.

The third class of algorithms completely foregoes homology-based comparison and instead works in terms of fold recognition alone. Again relaxations in the sequence similarity and template coverage requirements, which fall to less than 20% and greater than 50% respectively, are offset by worse accuracy and increased computation time. Accuracy is generally between 3 - 10Å after computation requiring hours for these methods.

Ab initio methods comprise the last class of algorithms. While the previous three classes all use some manner Informatic about how the conformations of known proteins, pure *ab initio* techniques explore structure space with the guidance of an energy function that they seek to minimise without regard to known structures. These techniques are useful to predict novel structures for which comparisons against known structures do not yield useful information. In addition, as *ab initio* techniques improve, they may yield insight into the ways proteins fold in nature. Since these techniques start with less information, their performance is not as good. On the other hand, they can be applied in certain contexts where other techniques' information requirements are not satisfied. With sequence similarity below 10% (or zero for pure *ab initio*) and

no templates, *ab initio* algorithms can predict structures with an accuracy in the range of 5 - 20Å after computation lasting on the order of days.

Method	Sequence Similarity	Template Coverage	Accuracy	Difficulty	Comput. Expense
Homology	>30%	>90%	1-3 Å	Trivial	Seconds
FR/Homology	20-30%	>75%	2-5 Å	Easy	Minutes
Fold Recognition	<20%	>50%	3-10 Å	Moderate	Hours
Ab initio	<10%	0	5-20 Å	Hard	Days

Figure 1: Summary of Requirements and Performance

2. TRILOGY: An Algorithm for Motif Discovery

TRILOGY is an unsupervised motif discovery algorithm. A standard approach to motif finding in proteins is to manually identify important structures, e.g. an alpha helix, in proteins and then consult the amino acid sequence that produced the structure. Taking the sequences from many proteins that produce the structure in question, a probabilistic motif of sequence can be used to predict where the structure will occur in other proteins based on their sequence. TRILOGY improves on this method by not requiring manual identification of important structures. Instead, TRILOGY finds examines patterns of amino acids in known structures selects sequence to structure pairings that appear at a statistically significant rate according to a hypergeometric distribution. Patterns were required to span at least three SCOP superfamilies to ensure that they were truly general. The assumption is that frequent appearance of a structure implies that it will serve as a good template for later structure prediction. In its results, TRILOGY was able to find known structural motifs such as helix capping as well as useful novel motifs such as helix strands and interlinked disulfides.

TRILOGY begins with triplets of amino acids, with representations for both sequence and structure. The sequence representation is similar to a regular expression. For example, $R_1x^{a-b}R_2x^{c-d}R_3$ represents the pattern of amino acid R_1 followed by between a and b amino acids of unspecified type, followed by R_2 , then between c and d amino acids of unspecified type, and finally R_3 . These sequence patterns may also contain options, such as that R_2 may actually be any one of several amino acids that tend to share properties and so substitute for one another more easily. The structure representation is analogous. Each amino acid is represented as a vector from its alpha carbon to its beta carbon, which encodes the orientation of the side chain, as well as a minimum and maximum distance from the other two amino acids in the pattern. As the data permit, triplets may be extended or combined to form larger patterns.

Citation: Bradley P, Kim PS, Berger B. TRILOGY: Discovery of sequence-structure patterns across diverse proteins. Proc Natl Acad Sci U S A. 2002 Jun 25;99(13):8500-5.

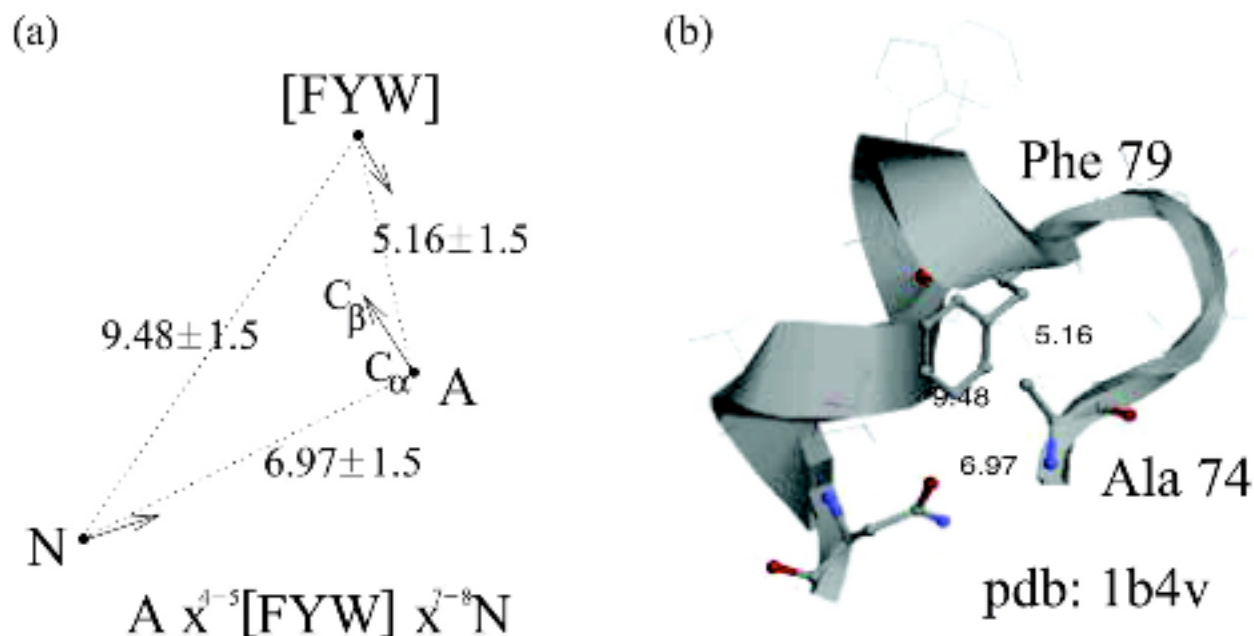


Figure 2: TRILOGY triplet representation and example

3. Small Representative Fragment Libraries

Small representative fragment libraries contain a general set of fragments, which are akin to TRILOGY patterns and motifs. A fragment is small, rigid portion of a three-dimensional protein structure. Given a set of protein fragments to use as building blocks, the fragments may be connected to form an approximation to the backbone of a known protein. Alternatively, the set of fragments may be viewed as an alphabet and proteins may be approximated to words or strings constructed from such an alphabet. The library consists of a set of fragments as well as a mapping from sequence patterns that commonly create them. Thus the sequence may be used to posit a string in the alphabet of fragments which is, in turn, a representation of a three-dimensional structure. This constitutes a framework for structure prediction from sequence.

Colony et al. examine the accuracy of this technique. Using 200 unique proteins with distinct and reliable structures, they produce libraries with fragments of constant length four, five, six, or seven amino acids. The library is further compressed by using a clustering algorithm to determine representative fragments with only the representative fragments being stored in the library. Thus the richness of the library depends on the number of instances of each residue sequence pattern seen and the number of clusters used. As the length of fragments in the library increases and the number of example proteins is held constant, the number of instances of each residue sequence pattern falls. They examine the accuracy, in terms of cRMSD for varying fragment lengths, library sizes, and complexities (number of permitted states of a fragment, which are rigid).

Table 2. Average accuracy of global and local cRMS deviations

Fragment length	Library size	Complexity (states/residue) ^c	Average ^{a,b}	
			Local cRMS (Å)	Global cRMS (Å)
4	4	4.00	0.39	2.23
4	6	6.00	0.35	1.64
4	7	7.00	0.33	1.48
4	8	8.00	0.32	1.39
4	10	10.00	0.30	1.12
4	12	12.00	0.28	1.01
4	14	14.00	0.26	0.92
5	10	3.16	0.57	2.57
5	20	4.47	0.47	1.85
5	30	5.48	0.43	1.59
5	40	6.32	0.40	1.41
5	50	7.07	0.39	1.28
5	60	7.75	0.37	1.20
5	80	8.94	0.35	1.06
5	100	10.00	0.34	0.99
5	150	12.25	0.31	0.86
5	225	15.00	0.29	0.76
6	40	3.42	0.65	2.30
6	60	3.91	0.59	2.02
6	70	4.12	0.58	1.92
6	80	4.31	0.56	1.87
6	100	4.64	0.54	1.72
6	200	5.85	0.48	1.41
6	300	6.69	0.45	1.26
7	50	2.66	0.85	2.89
7	100	3.16	0.76	2.41
7	150	3.50	0.72	2.16
7	200	3.76	0.68	2.04
7	250	3.98	0.66	1.91
Micheletti <i>et al.</i> fragment libraries ^d				
5	40	6.32	0.48	1.64
6	100	4.64	0.57	1.88

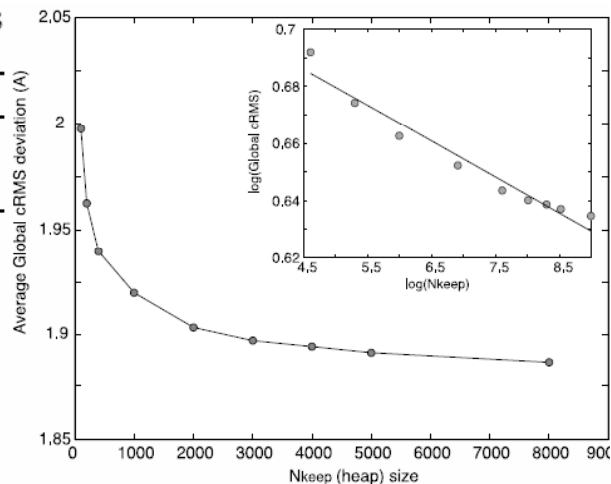


Figure 3: Accuracy Results for Variations in Fragment Length, Library Size, and Complexity

Citation: Colony R, Koehl P, Guibas L, Levitt M. Small Libraries of Protein Fragments Model Native Protein Structures Accurately. *J Mol Biol.* 2002;323:297–307.

4. Side-Chain Packing

The side-chain packing problem is defined as, given the coordinates of the protein backbone, find the orientation of the side chains of the amino acids. Any algorithm attempting to solve this problem must contend with steric clashes, which are where two or more side-chains are arranged such that they occupy the same space, and with some objective function for assessing the quality of different possible arrangements of the side-chains.

One solution is to discretise the problem in two ways. First, the three-dimension space containing protein is discretised according to a mesh. Atoms, and by extension, side-chains, are said to fill a volume, frequently a cube, in the mesh. A steric clash can be defined as the two side-chains occupying the same volume. Computation expense and accuracy generally increase with the resolution of the mesh.

The second discretisation is of the orientations for the side-chains. In principle, a side-chain could be at any one of an infinite number of positions rotating around the alpha carbon. The

orientations are generally discretised into a handful of representative orientations, called rotamers. It is possible to perform preprocessing to eliminate combinations of rotamers that produce steric clashes or to allow the objective function to include a penalty for clashes. Rotamers in the core of the protein are generally significantly more restricted than those on the surface.

The objective function frequently draws its inspiration from a potential energy function and includes a penalty term for steric clashes. The clash penalty need not be constant but may rather vary with the distance that separates two atoms. This yields the general objective function

$$f(p, A) = \sum_{i \in \text{residues}(p)} \left[S(i, A(i)) + \sum_{j \in \text{residues}(p), i \neq j} P(i, j, A(i), A(j)) \right]$$

where p is a protein, A is the set of selected rotamers, $A(i)$ is the rotamer selected for residue i , S is the value of assigning $A(i)$ to i , and P is the penalty function for clashes between two rotamer assignments for given residues.

Generally, structure prediction algorithms using side-chain packing proceed in four stages. The first is to predict the path of the backbone to which the side chains will be attached. This may be done by several techniques, including protein threading, homology modeling, or *ab initio* folding. Once the backbone is in place, the placement of the loops between secondary structure elements is refined. Then the algorithm attempts to pack the side-chains given the structure of the backbone. Finally, once the side chains are packed, the model is run through a structural refinement process which may involve, for example, a simulation to find a local minimum in a potential energy function. This produces a molecule that is generally more stable than the more approximate model as it exists at the end of the side-chain packing stage.

One insight into side-chain packing is that, if two side chains cannot directly clash with one another, then the extent of their interaction is limited in some sense. More specifically, let G , the residue interaction graph, be a graph containing one vertex for every residue of the protein. Let uv be an edge in G if and only if there is a rotamer assignment for the residue represented by u that clashes with any rotamer assignment for the residue represented by v . Now assume there is one and only one uv path in G and that this path contains vertex w . If we assign a rotamer to the residue of w , then assignments of rotamers to the residues of u and v cannot influence one another. In a sense, u and v are independent given w . Taking advantage of these conditional independencies can greatly reduce the computation required to find non-clashing rotamer assignments.

The ordering of assignments and the computations necessary depend on the structure of a tree decomposition of the interaction graph (cf. clique trees). A tree decomposition, (T, X) , of $G = (V, E)$ has the following properties.

- T is a tree with node set I and edge set F .
- X is a set of subsets of V , called components, such that the union of all the sets in X is V .
- There is a 1-1 mapping between I and X . In other words, the nodes of T represent subsets of vertices in G .
- For every uv edge in E , there is at least one set in X that contains both u and v . By

implication, if u and v are directly connected in G , then there is at least node of T that contains both u and v .

- In T , if a node j is on a path from i to k , then $X(i) \cap X(k) \subseteq X(j)$. This is known as the Running Intersection Property.

The tree width is defined as the size of the largest set in X , minus one. The tree width of a graph is defined as the minimum tree width of all tree decompositions of the graph. Note that there is not one unique tree decomposition of a graph and that finding the tree with the minimum is NP hard. For this reason, approximations using heuristics such as greediness frequently used.

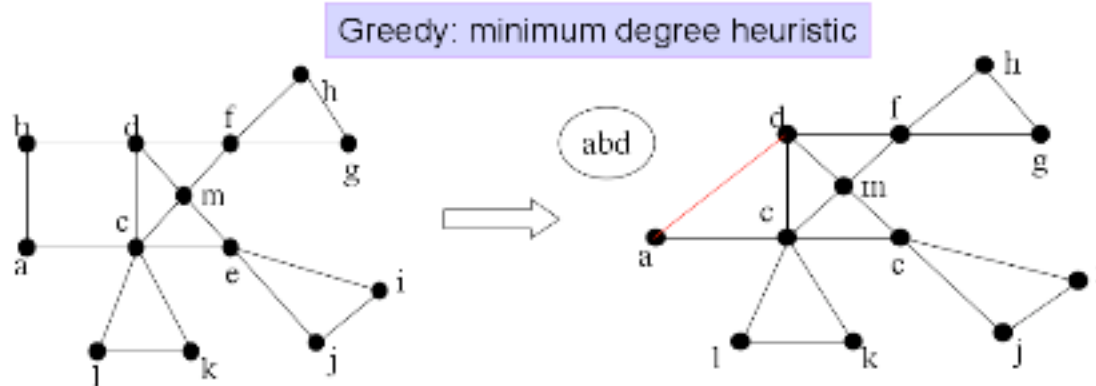


Figure 4: One Step of Tree Decomposition as Guided by the Minimum Degree Heuristic

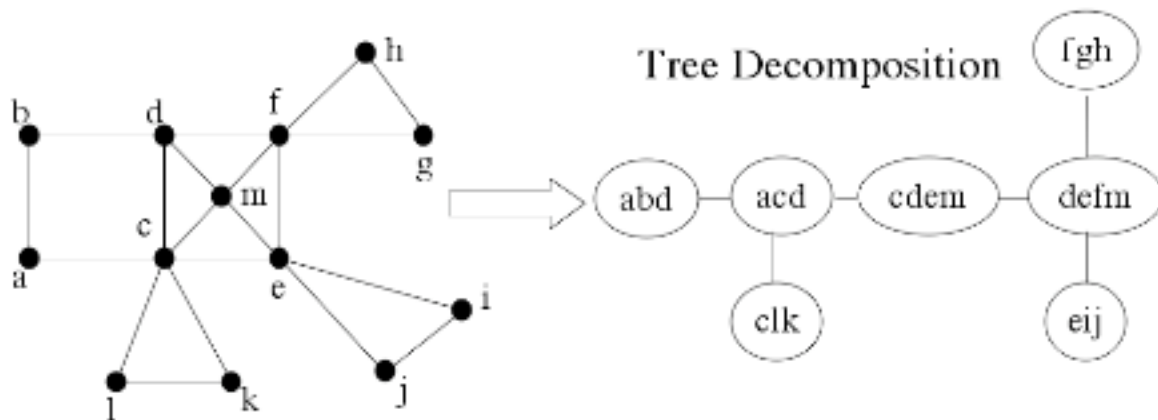


Figure 5: Completed Tree Decomposition Using Minimum Degree Heuristic

In Figure 5, the tree width of the shown tree decomposition is 4 - 1, or 3.

The brute force, most simplistic approach of assigning rotamers is to examine each possible assignment. Assuming a bounded number of rotamers per residue, this approach is exponential in the number of residues, or equivalently, in the number of vertices in the interaction graph. Using the tree decomposition, this can be reduced to exponential in the tree width of the best decomposition available. Since the tree width is never greater than the number of vertices in the graph minus one and, for this domain, is generally much, much smaller (due to the large number of independencies), significant reductions in computation time are possible.

The algorithm for computing side-chain packing scores as guided by the tree decomposition of the interaction graph proceeds along the following lines. First, a leaf node in the tree is chosen. For this node, a table is constructed containing the score for all possible assignments of rotamers to the residues represented by the node. Let the intersection set be the subset of residues in the current node that are also in the node's parent. For the node $\{abd\}$ in Figure 5, its parent is $\{acd\}$ and hence its intersection set is $\{ad\}$. The table for $\{abd\}$ contains one row for every assignment of a rotamer to a crossed with every rotamer for b crossed with every rotamer for d . The table is then collapsed to having rows with the Cartesian cross product of just the intersection set. This means that, many times, several rows must be collapsed into one, by taking the minimum across all values of b for every possible assignment to a and d , in this case. The node $\{abd\}$ is then eliminated from the tree. When the parent, $\{acd\}$ is computed, it uses the collapsed table just produced in order to compute the scoring function. Since the size of any table produced by this method is bounded by the largest number of residues any node represents, it follows that this algorithm is exponential in the tree width.