Origami with strings: protein folding by computer

Kevin Karplus

karplus@soe.ucsc.edu

Biomolecular Engineering Department

Undergraduate and Graduate Director, Bioinformatics

University of California, Santa Cruz

Outline of Talk

- What is Biomolecular Engineering? Bioinformatics?
- What is a protein?
- The folding problem and variants on it:
  - Local structure prediction
  - Fold recognition
  - Comparative modeling
  - “Ab initio” methods
  - Contact prediction

What is Biomolecular Engineering?

Engineering **with**, **of**, or **for** biomolecules. For example,

with: using proteins as sensors or for self-assembly.

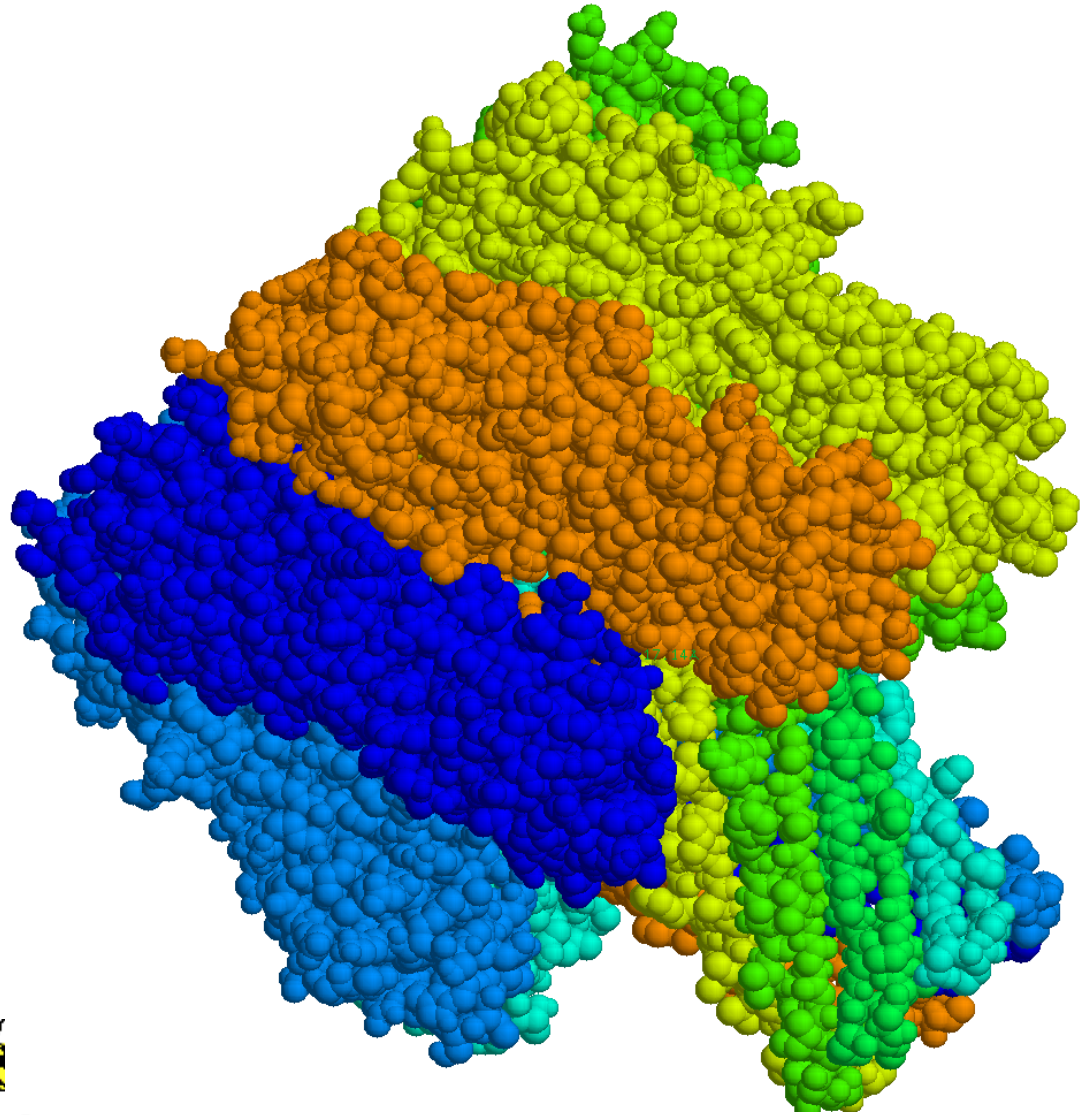
of: protein engineering—designing or artificially evolving proteins to have particular functions

for: designing high-throughput experimental methods to find out what molecules are present, how they are structured, and how they interact.

Nanopore: example of BME

- The “nanopore” experiments at UCSC use a protein (α-hemolysin) that self-assembles in a lipid bilayer membrane to punch a tiny (about 17 Angstrom diameter) hole.
- The nanopore is just large enough for single-stranded DNA to pass through, but not double-stranded DNA.
- Ion current through the hole is used to detect single molecules of DNA folding, unfolding, and passing through the pore.

Nanopore: PDB file 7ahl



What is Bioinformatics?

Bioinformatics: using computers and statistics to make sense out of the mountains of data produced by high-throughput experiments.

- Genomics: finding important sequences in the genome and annotating them.
- Phylogenetics: “tree of life”.
- Systems biology: piecing together various control networks.
- DNA microarrays: what genes are turned on under what conditions.
- Proteomics: what proteins are present in a mixture.
- Protein structure prediction.

What is a protein?

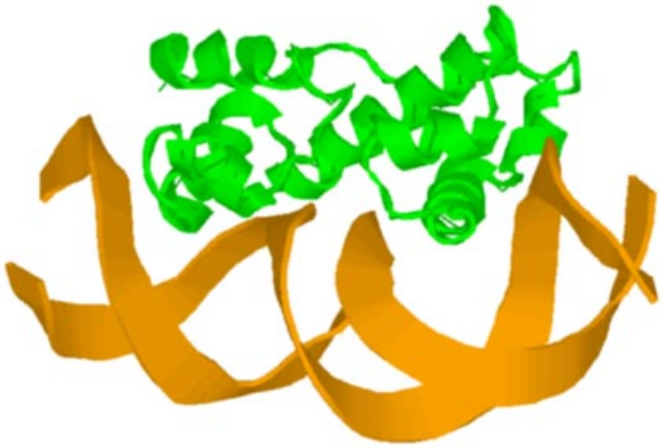
- There are many abstractions of a protein: a band on a gel, a string of letters, a mass spectrum, a set of 3D coordinates of atoms, a point in an interaction graph, ...
- For us, a protein is a long skinny molecule (like a string of letter beads) that folds up consistently into a particular intricate shape.
- The individual “beads” are amino acids, which have 6 atoms the same in each “bead” (the *backbone* atoms: N, H, CA, HA, C, O).
- The final shape is different for different proteins and is essential to the function.
- The protein shapes are important, but are expensive to determine experimentally.

Folding Problem

The *Folding Problem*:
If we are given a sequence of amino acids (the letters on a string of beads), can we predict how it folds up in 3-space?

```
MTMSRRNTDA ITIHSILDWI EDNLESPLSL EKVSERSGYS KWHLQRMFKK
ETGHSLGQYI RSRKMTEIAQ KLKESNEPIL YLAERYGFES QQTLTRTFKN
YFDVPPHKYR MTNMQGESRF LHPLNHYNS
```

↓



Too hard!

Fold-recognition problem

The *Fold-recognition Problem*:
Given a sequence of amino acids A (the *target* sequence) and a library of proteins with known 3-D structures (the *template* library),
figure out which templates A match best, and align the target to the templates.

- The backbone for the target sequence is predicted to be very similar to the backbone of the chosen template.
- Progress has been made on this problem, but we can usefully simplify further.

Remote-homology Problem

The *Homology Problem*:
Given a target sequence of amino acids
and a library of protein *sequences*,
figure out which sequences A is similar to and align them to A.

- No structure information is used, just sequence information. This makes the problem easier, but the results aren't as good.
- This problem is fairly easy for recently diverged, very similar sequences, but difficult for more remote relationships.

New-fold prediction

- What if there is *no* template we can use?
- We can try to generate many conformations of the protein backbone and try to recognize the most protein-like of them.
- Search space is huge, so we need a good conformation generator and a cheap cost function to evaluate conformations.

Secondary structure Prediction

- Instead of predicting the entire structure, we can predict local properties of the structure.
- What local properties do we choose?
- We want properties that are well-conserved through evolution, easily predicted, and useful for finding and aligning templates.
- One popular choice is a 3-valued helix/strand/other alphabet—we have investigated many others. Typically, predictors get about 80% accuracy on 3-state prediction.
- Many machine-learning methods have been applied to this problem, but the most successful is neural networks.

CASP ~~Competition~~ Experiment

- Everything published in literature “works”
- CASP set up as true blind test of prediction methods.
- Sequences of proteins about to be solved released to prediction community.
- Predictions registered with organizers.
- Experimental structures compared with solution by assessors.
- “Winners” get papers in *Proteins: Structure, Function, and Bioinformatics*.

Predicting Local Structure

- Want to predict some local property at each residue.
- Local property can be emergent property of chain (such as being buried or being in a beta sheet).
- Property should be conserved through evolution (at least as well as amino acid identity).
- Property should be somewhat predictable (we gain information by predicting it).
- Predicted property should aid in fold-recognition and alignment.
- For ease of prediction and comparison, we look only at discrete properties (alphabets of properties).

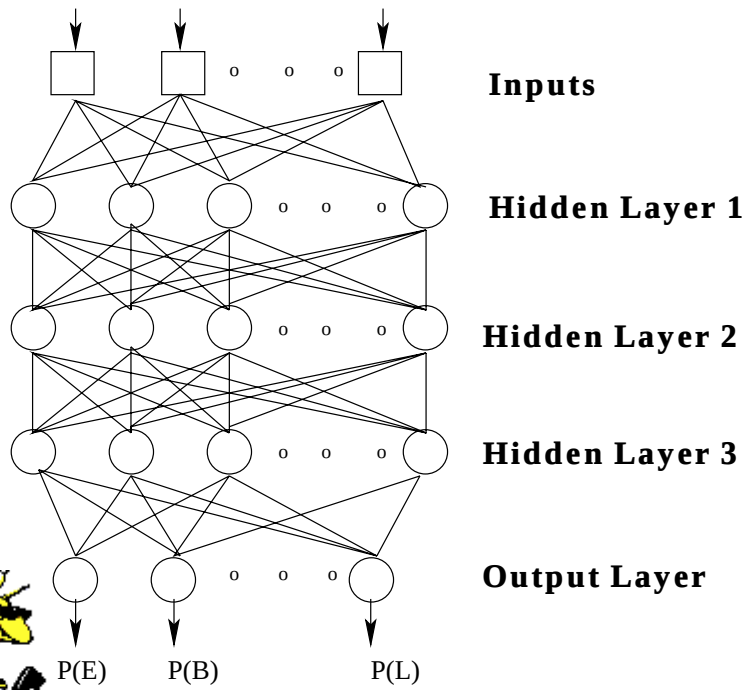
Using Neural Net

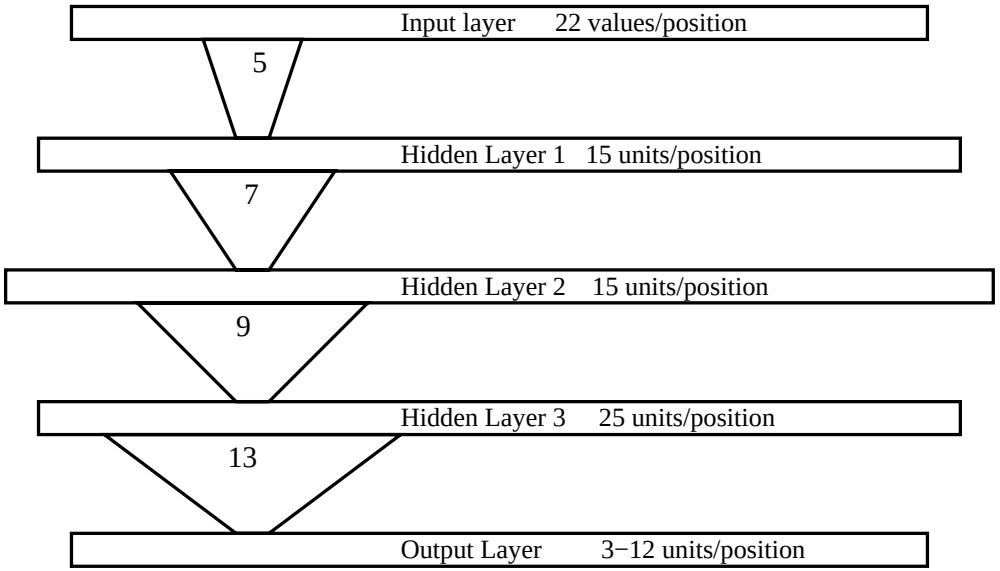
- We use neural nets to predict local properties.
- Input is profile with probabilities of amino acids at each position of target chain, plus insertion and deletion probabilities.
- Output is probability vector for local structure alphabet at each position.
- Each layer takes as input windows of the chain in the previous layer and provides a probability vector in each position for its output.
- We train neural net to maximize $\sum \log(P(\text{correct output}))$.

Neural Net

Typical net has 4 layers and 6471 weight parameters:

input/pos	window	output/pos	weights
22	5	15	1665
15	7	15	1590
15	9	15	2040
15	13	6	1176



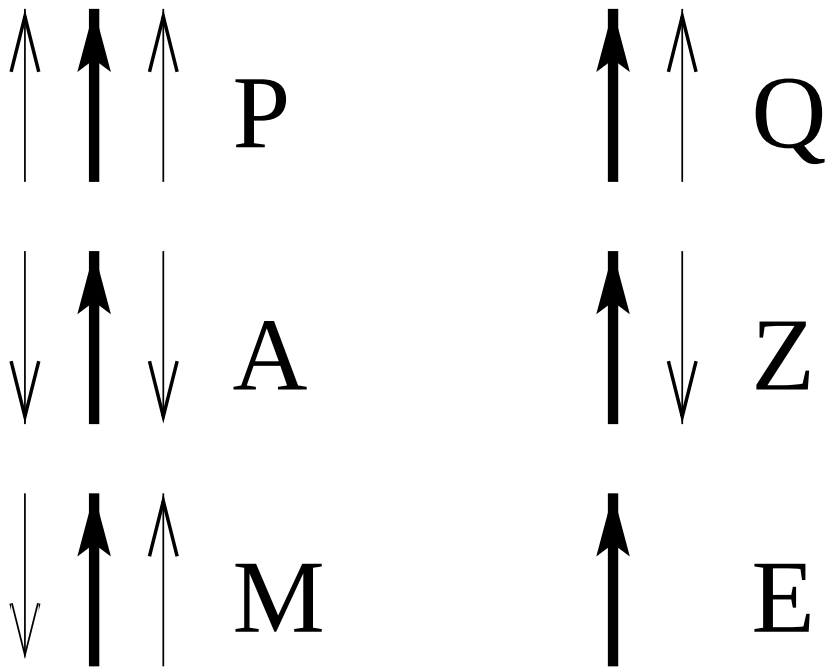


DSSP

- DSSP is a popular program to define secondary structure.
- 7-letter alphabet: EBGHSTL
  - E = β strand
  - B = β bridge
  - G = 3_{10} helix
  - H = α helix
  - I = π helix (very rare, so we lump in with H)
  - S = bend
  - T = turn
  - L = everything else (DSSP uses space for L)

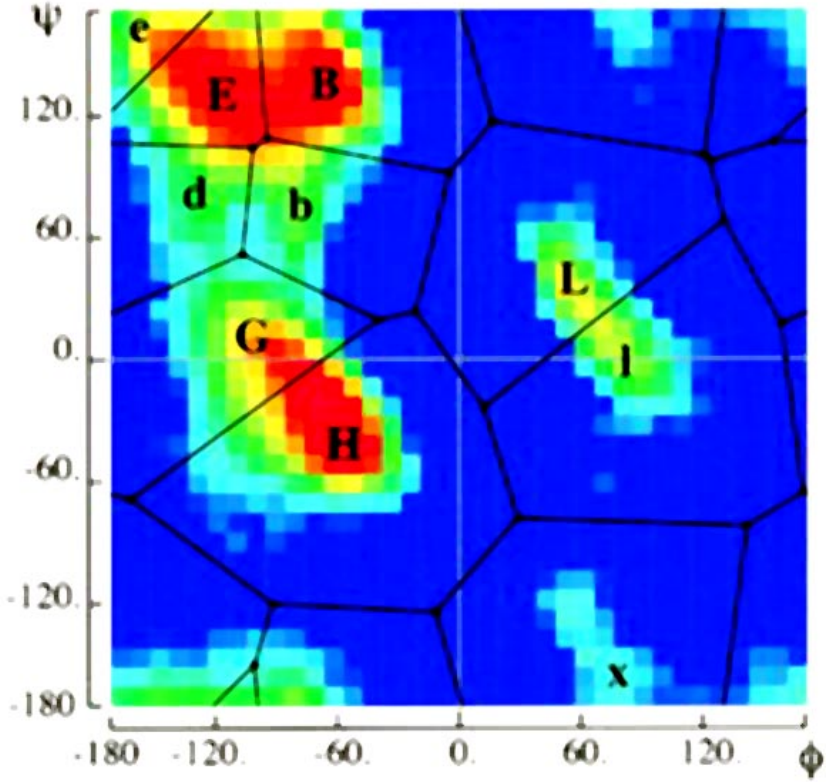
STR: Extension to DSSP

- Yael Mandel-Gutfreund noticed that parallel and anti-parallel strands had different hydrophobicity patterns, implying that parallel/antiparallel can be predicted from sequence.
- We created a new alphabet, splitting DSSP's E into 6 letters:



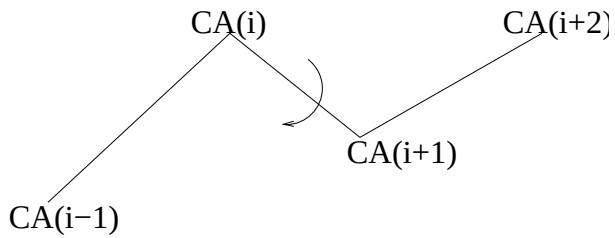
HMMSTR ϕ-ψ alphabet

- For HMMSTER, Bystroff did k-means classification of ϕ-ψ angle pairs into 10 classes (plus one class for cis peptides).
- We used just the 10 classes, ignoring the ω angle.

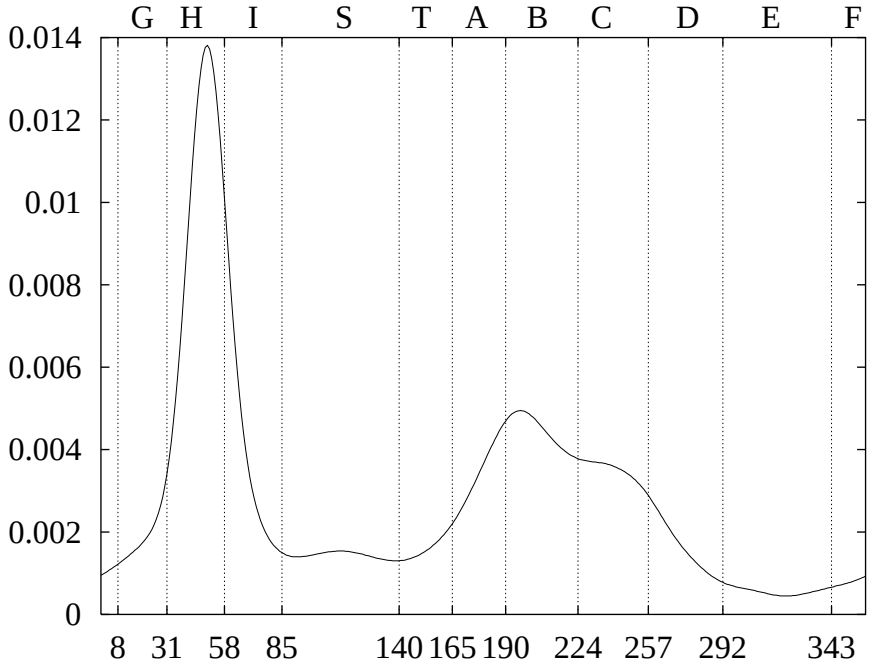


ALPHA11: α angle

- Backbone geometry can be mostly summarized with one angle per residue:

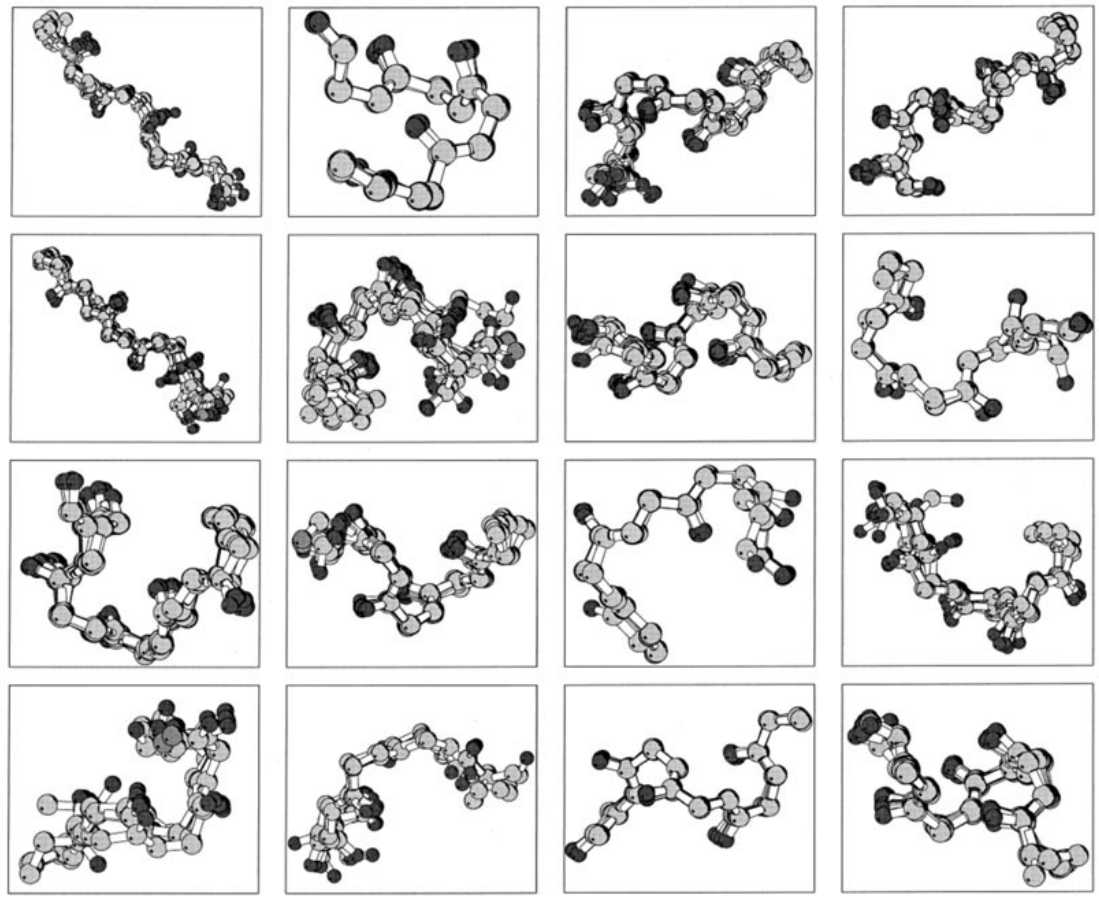


- We discretize into 11 classes:



de Brevern's Protein Blocks

Clustered on 5-residue window of ϕ-ψ angles:



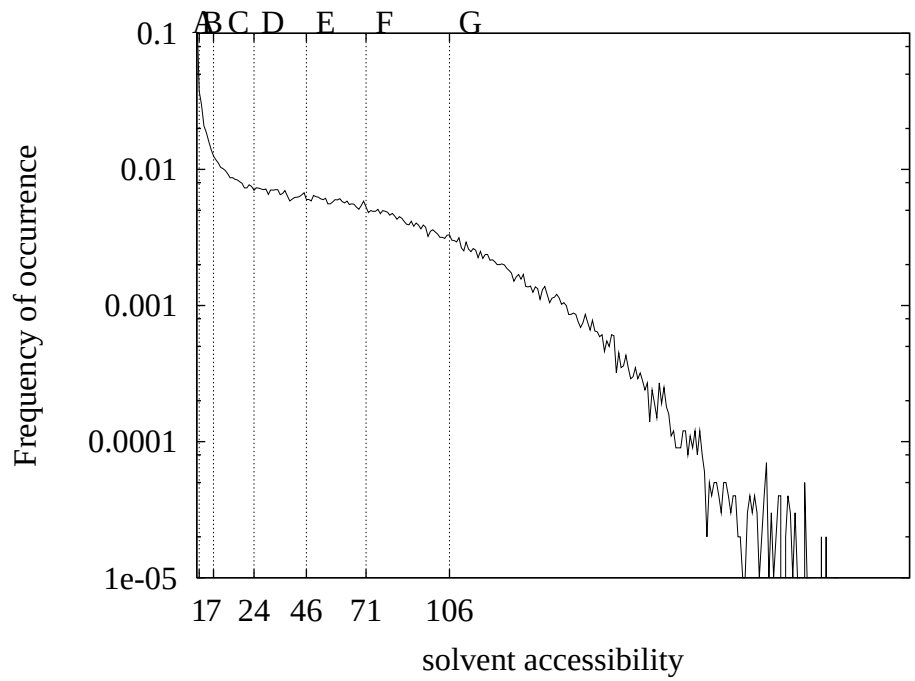
Burial alphabets

Our second set of investigations was for a sampling of the many burial alphabets, which are discretizations of various accessibility or burial measures:

- solvent accessible surface area
- relative solvent accessible surface area
- neighborhood-count burial measures

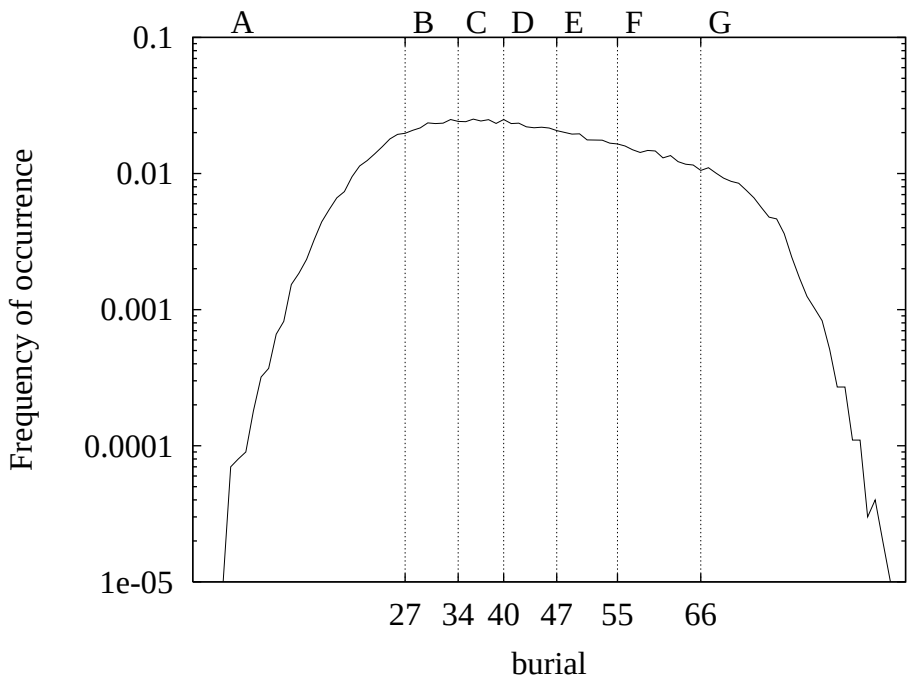
Solvent Accessibility

- Absolute SA: area in square Ångstroms accessible to a water molecule, computed by DSSP.
- Relative SA: Absolute SA/ max SA for residue type (using Rost's table for max SA).



Burial

- Define a sphere for each residue.
- Count the number of atoms or of residues within that sphere.
- Example: center= C_β, radius=14Å, count= C_β, quantize in 7 equi-probable bins.



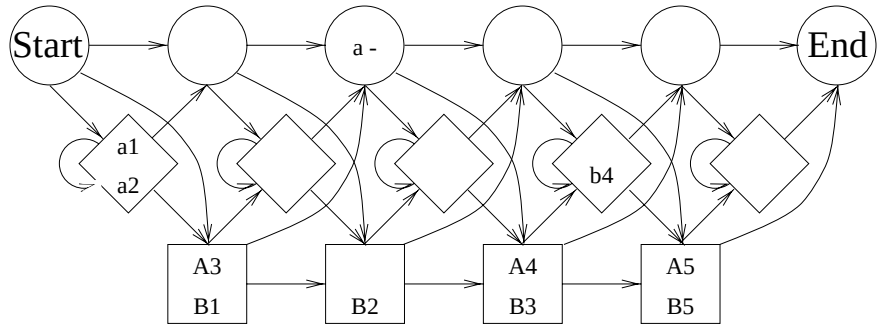
Conservation and Predictability

Name	alphabet size	entropy	MI with AA	conservation mutual info	predictability info gain per residue	$Q_{\|A\|}$
str	13	2.842	0.103	**1.107**	1.009	0.561
protein blocks	16	3.233	0.162	0.980	**1.259**	0.579
stride	6	2.182	0.088	0.904	0.863	0.663
DSSP	7	2.397	0.092	0.893	0.913	0.633
stride-EHL	3	1.546	0.075	0.861	0.736	0.769
DSSP-EHL	3	1.545	0.079	0.831	0.717	0.763
CB-16	7	2.783	0.089	**0.682**	0.502	
CB-14	7	2.786	0.106	0.667	**0.525**	
CB-12	7	2.769	0.124	0.640	0.519	
rel SA	7	2.806	0.183	0.402	0.461	
abs SA	7	2.804	0.250	0.382	0.447	

Hidden Markov Models

- *Hidden Markov Models* (HMMs) are a very successful way to capture the variability possible in a family of proteins.
- An HMM is a stochastic model—that is, it assigns a probability to every possible sequence.
- An HMM is a finite-state machine with a probability for emitting each letter in each state, and with probabilities for making each transition between states.
- Probabilities of letters sum to one for each state.
- Probabilities of transitions out of each state sum to one for that state.
- We also include *null states* that emit no letters, but have transition probabilities on their out-edges.

Profile Hidden Markov Model



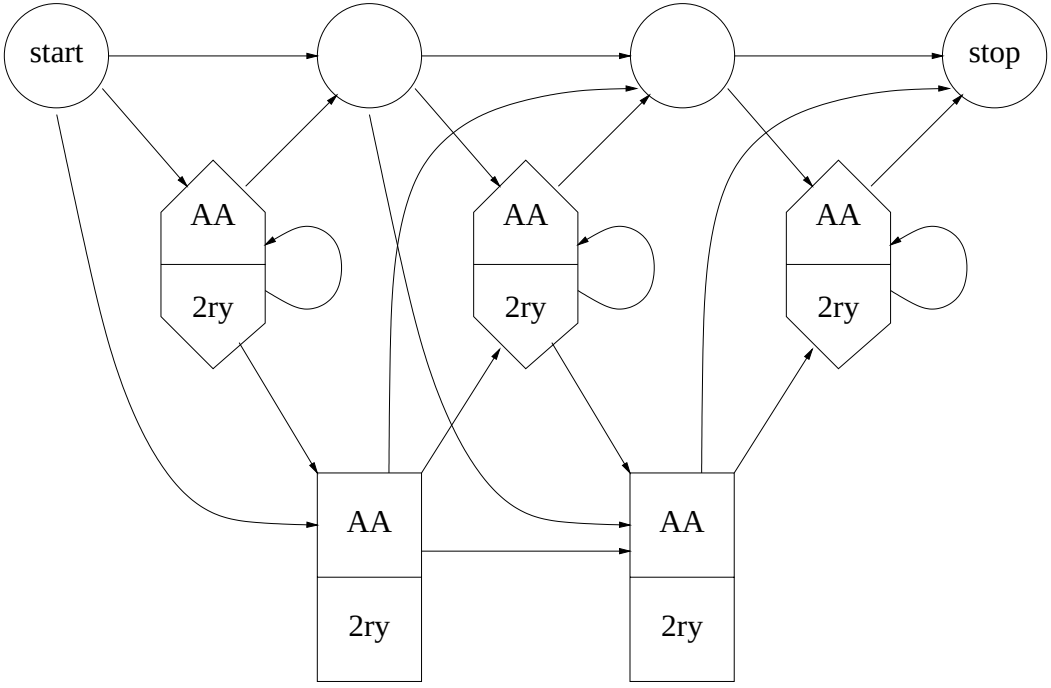

```
a1  a2  A3     -     A4   .   A5
 .   .  B1    B2     B3  b4  B5
```

- Circles are null states.
- Squares are *match states*, each of which is paired with a null *delete state*. We call the match-delete pair a *fat state*.
- Each fat state is visited exactly once on every path from Start to End.
- Diamonds are *insert states*, and are used to represent possible extra amino acids that are not found in most of the sequences in the family being modeled.

Multi-track HMMs

We can also use alignments to build a two- or three-track target HMM:

- Amino-acid track (created from the multiple alignment).
- Local-structure track(s) with probabilities from neural net.
- Can align template (AA+local) to target model.



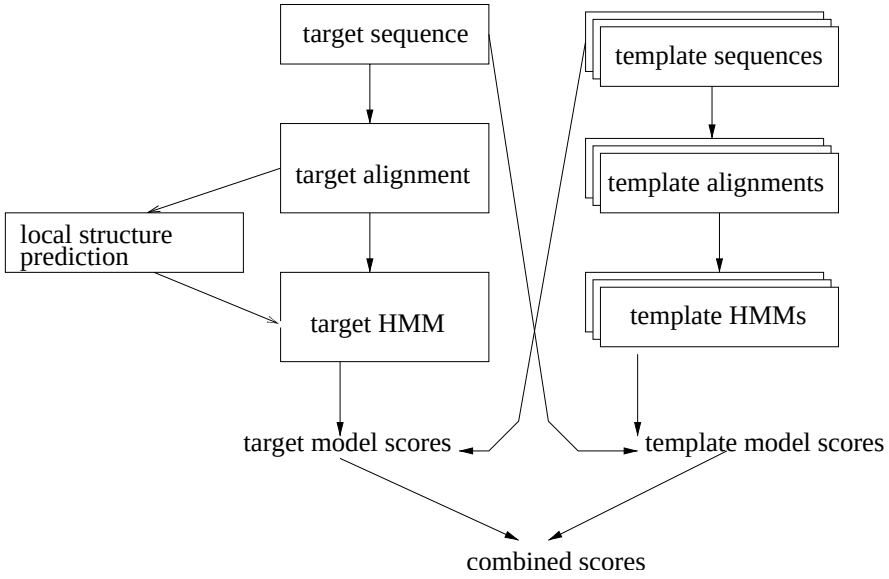
Target-model Fold Recognition

- Find probable homologs of target sequence and make multiple alignment.
- Make secondary structure probability predictions based on multiple alignment.
- Build an HMM based on the multiple alignment and predicted 2ry structure (or just on multiple alignment).
- Score sequences and secondary structure sequences for proteins that have known structure (all sequences for AA-only, 8,000-11,000 representatives for multi-track).
- Select the best-scoring sequence(s) to use as templates.

Template-library Fold Recognition

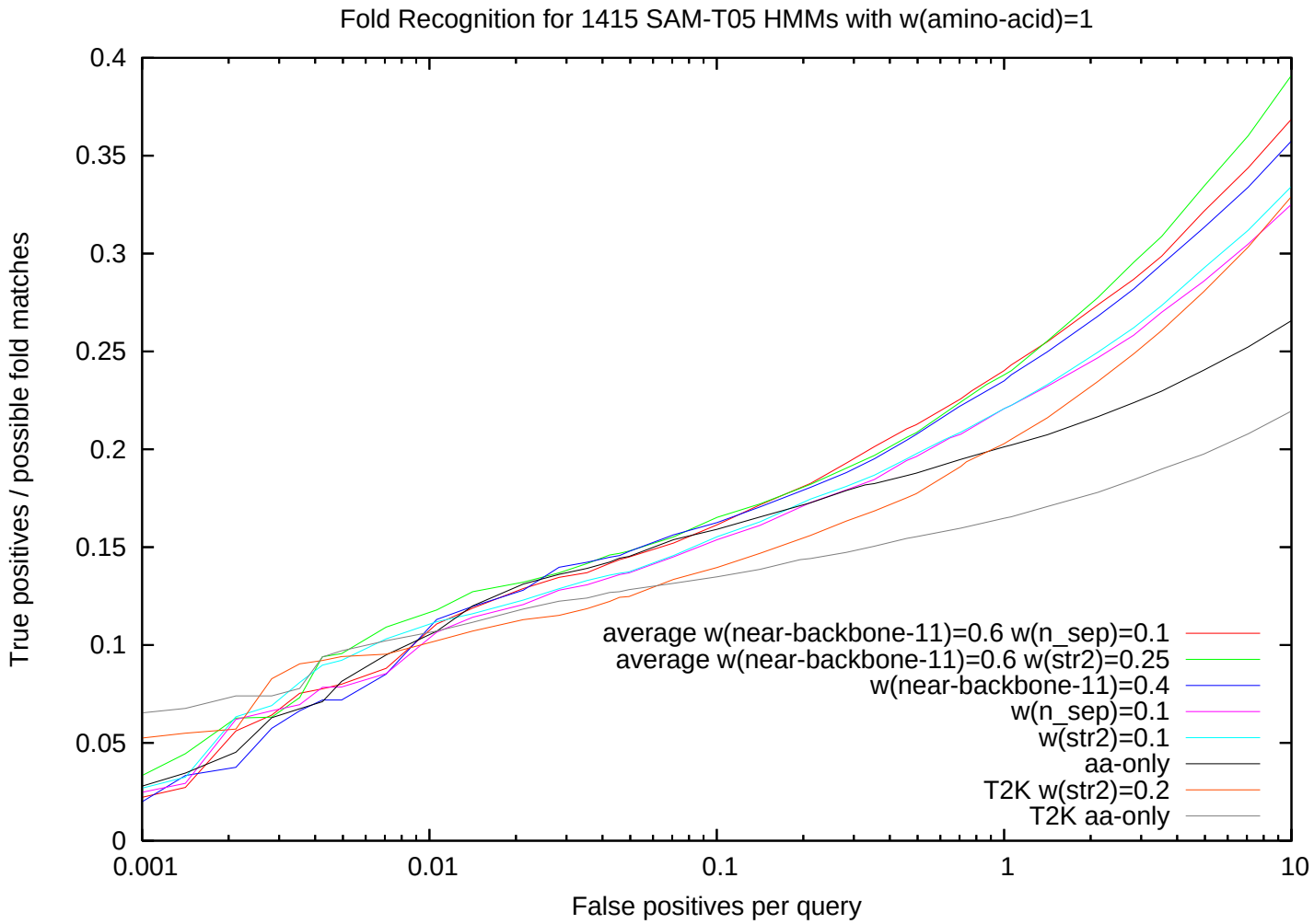
- Build an HMM for each protein in the template library, based on the template sequence (and any homologs you can find).
- The T2K library has over 11,000 templates from PDB.
- For the fold-recognition problem, structure information can be used in building these models (though we currently don't).
- Score target sequence with all models in the library.
- Select the best-scoring model(s) to use as templates.

Combined SAM-T02 method



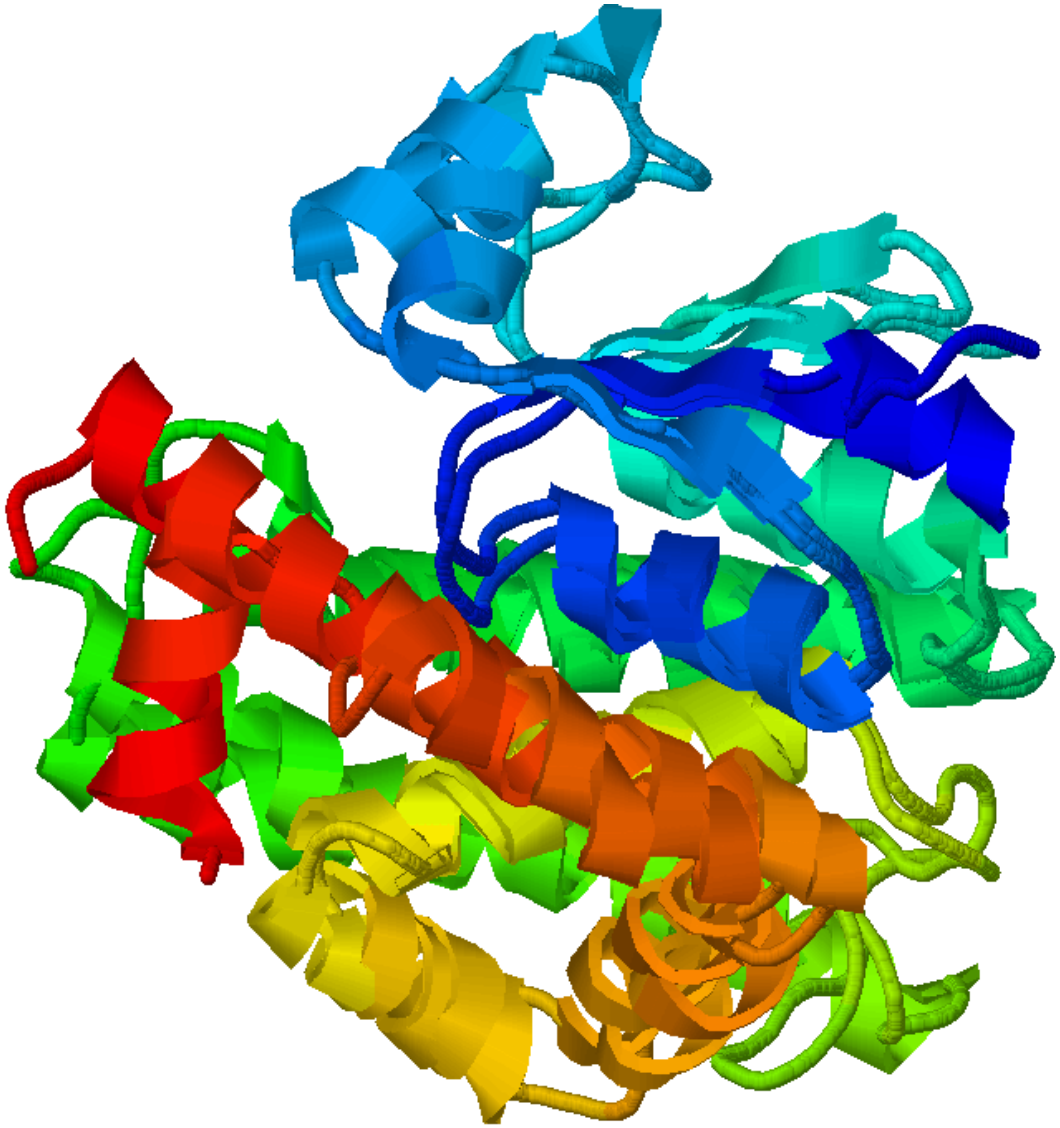
- Combine the costs from the template library search and the target library searches using different local structure alphabets.
- Choose one of the many alignments of the target and template (whatever method gets best results in testing).
- `http://www.soe.ucsc.edu/research/compbio/HMM-apps/T02-query.html`

Fold recognition results



Comparative modeling: T0232

RMSD= 5.158Å all-atom, 4.463Å C_α



Undertaker

- Undertaker is UCSC's attempt at a fragment-packing program.
- Named because it optimizes burial.
- Representation is 3D coordinates of all heavy atoms (not hydrogens).
- Can replace backbone fragments (a la Rosetta) or full alignments—chain need not remain contiguous.
- Conformations can borrow heavily from fold-recognition alignments, without having to lock in a particular alignment.
- Use genetic algorithm with many conformation-change operators to do stochastic search.

Fragfinder

Fragments are provided to undertaker from 3 sources:

- Generic fragments (2-4 residues, exact sequence match) are obtained by reading in 500–1000 PDB files, and indexing all fragments.
- Long specific fragments (and full alignments) are obtained from the various target and template alignments generated during fold recognition.
- Medium-length fragments (9–12 residues long) for every position are generated from the HMMs with `fragfinder`, a new tool in the SAM suite.

Cost function

- Cost function is modularly designed—easy to add or remove terms.
- Cost function can include predictions of local properties by neural nets.
- Clashes and hydrogen bonds are important components.
- There are over 40 cost function components available: burial functions, disulfides, contact order, rotamer preference, radius of gyration, constraints, ...

Target T0201 (NF)

- We tried forcing various sheet topologies and selected 4 by hand.
- Model 1 has right topology (5.912Å all-atom, 5.219Å C_α).
- Unconstrained cost function not good at choosing topology (two strands curled into helices).
- Helices were too short.

Target T0201 (NF)

Contact prediction

- Use mutual information between columns.
- Thin alignments aggressively (30%, 35%, 40%, 50%, 62%).
- Compute e-value for mutual info (correcting for small-sample effects).
- Compute z-score of log(e-value) within protein.
- Feed e-values, z-scores, conservation, amino-acid profile, separation along chain into neural net.

Evaluating contact prediction

Two measures of contact prediction:

- Accuracy:

$$\frac{\sum \chi(i,j)}{\sum 1}$$

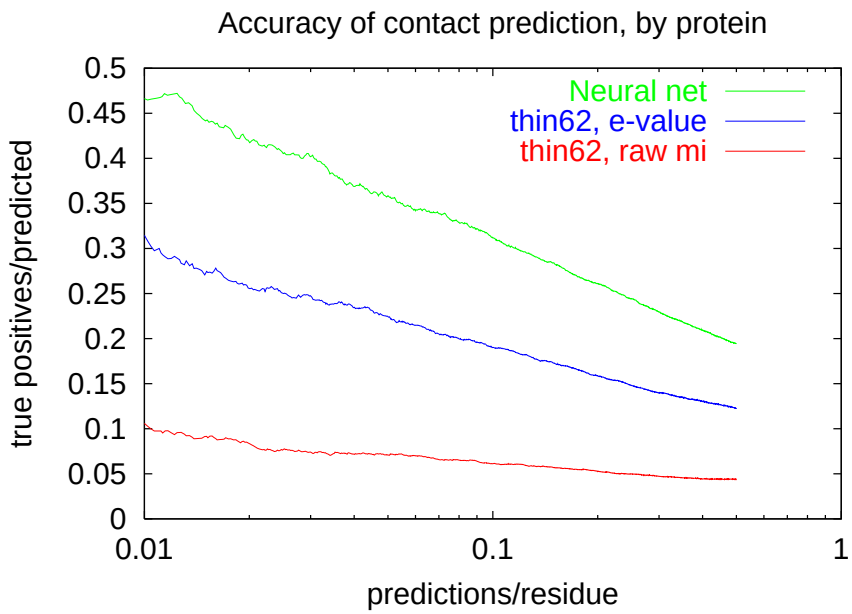
(favors short-range predictions, where contact probability is higher)

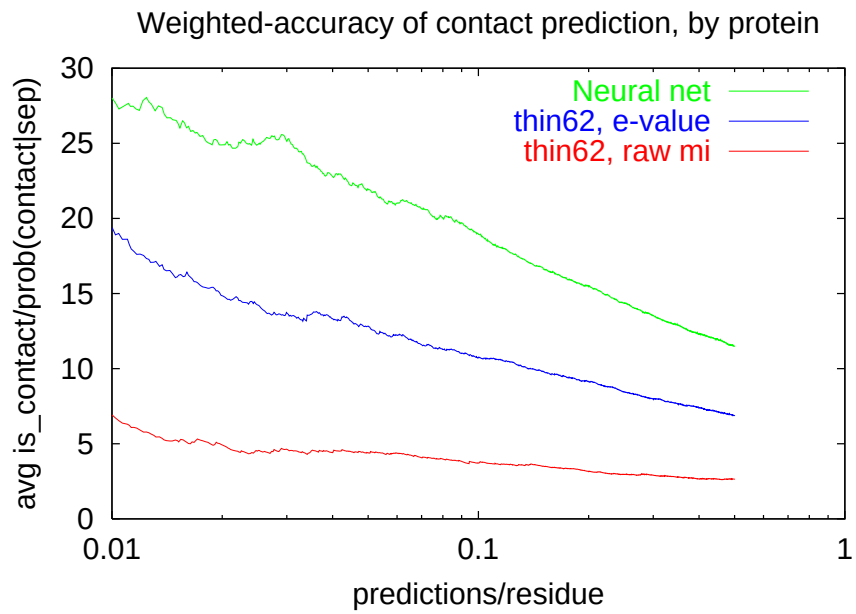
- Weighted accuracy:

$$\frac{\sum \frac{\chi(i,j)}{\text{Prob}\left(\text{contact}|\text{separation}_{=|i-j|}\right)}}{\sum 1}$$

(1 if predictions no better than chance based on separation).

Contact prediction results

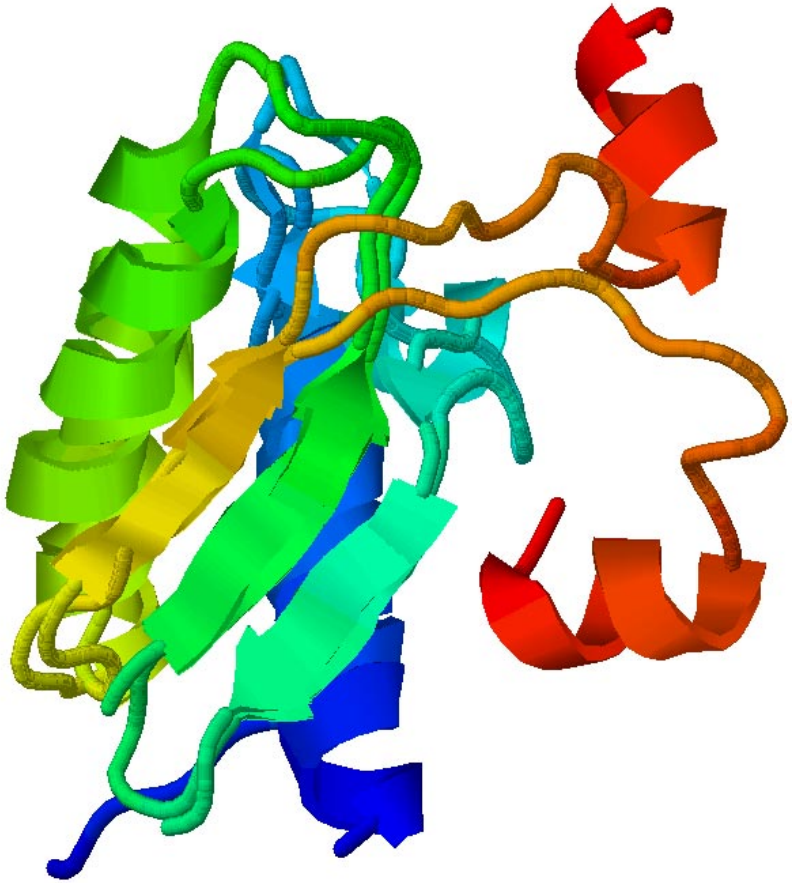




Target T0230 (FR/A)

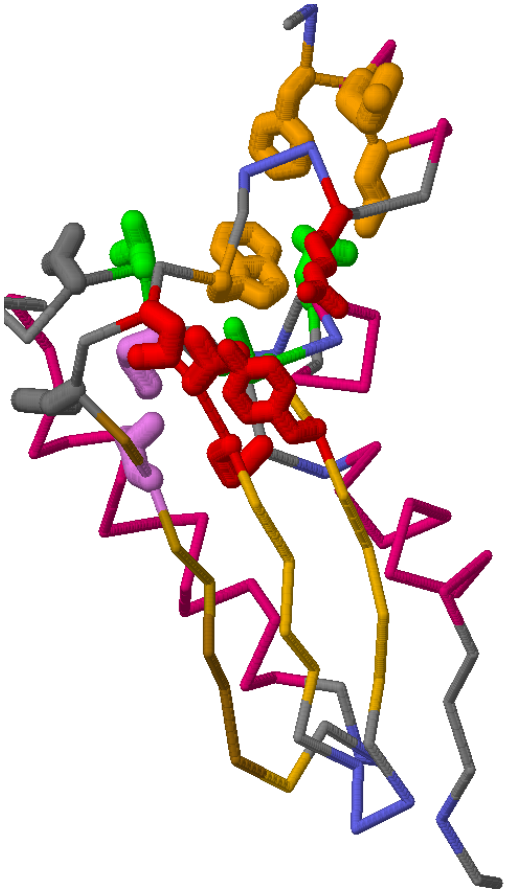
- Good except for C-terminal loop and helix flopped wrong way.
- We have secondary structure right, including phase of beta strands.
- Contact prediction helped, but we put too much weight on it—decoys fit predictions better than real structure does.

Target T0230 (FR/A)



Target T0230 (FR/A)

Real structure with contact predictions:



Web sites

These slides:

`http://www.soe.ucsc.edu/~karplus/papers/origami-with-strings-mar-2006`

SAM-T02 prediction server:

`http://www.soe.ucsc.edu/research/compbio/HMM-apps/T02-query.html`

CASP6 all our results and working notes:

`http://www.soe.ucsc.edu/~karplus/casp6/`

Predictions for all yeast proteins:

`http://www.soe.ucsc.edu/~karplus/yeast/`

UCSC bioinformatics (research and degree programs) info:

`http://www.soe.ucsc.edu/research/compbio/`

SAM tool suite info:

`http://www.soe.ucsc.edu/research/compbio/sam.html`