Origami with strings: predicting how proteins fold

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:
  - Fold recognition
  - Local structure prediction
  - Ab initio methods
  - Comparative modeling
- Results

What is Biomolecular Engineering?

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

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

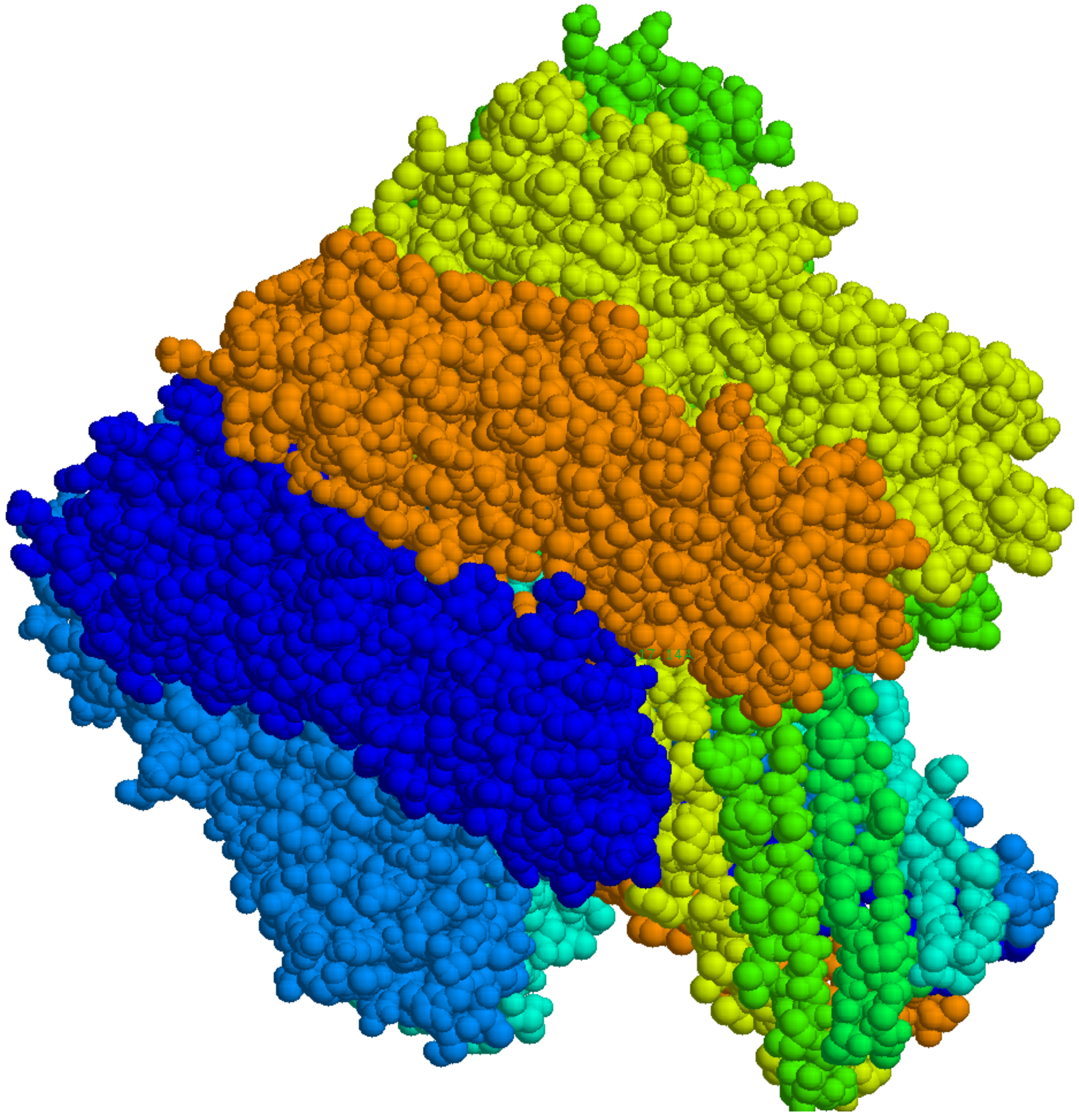
of: protein engineering—designing or artificially evolving proteins to have particular functions

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.
Examples:

- protein structure prediction.
- Genomics: finding genes in the genome and annotating them (function, cellular location, ...).
- DNA microarrays: finding out what genes are turned on under what conditions.
- Proteomics: finding out what proteins are present.
- Systems biology: piecing together the transcription, splicing, translation, post-translational modification, signaling, and degradation control networks.

What is a protein?

- 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).
- The final shape is different for different proteins and is essential to the function in our bodies.
- 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
```

$\downarrow$

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.
- Many machine-learning methods have been applied to this problem, but the most successful is neural networks.

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).

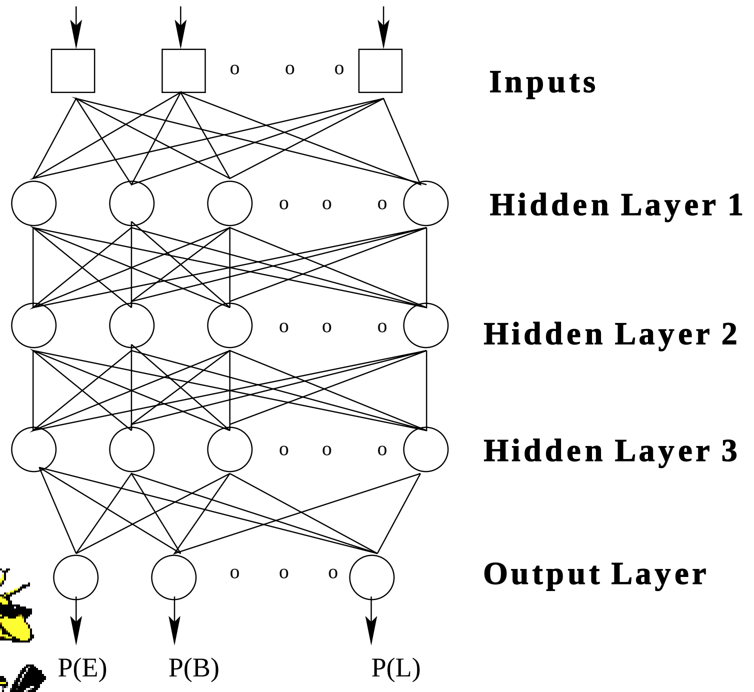
Using Neural Net

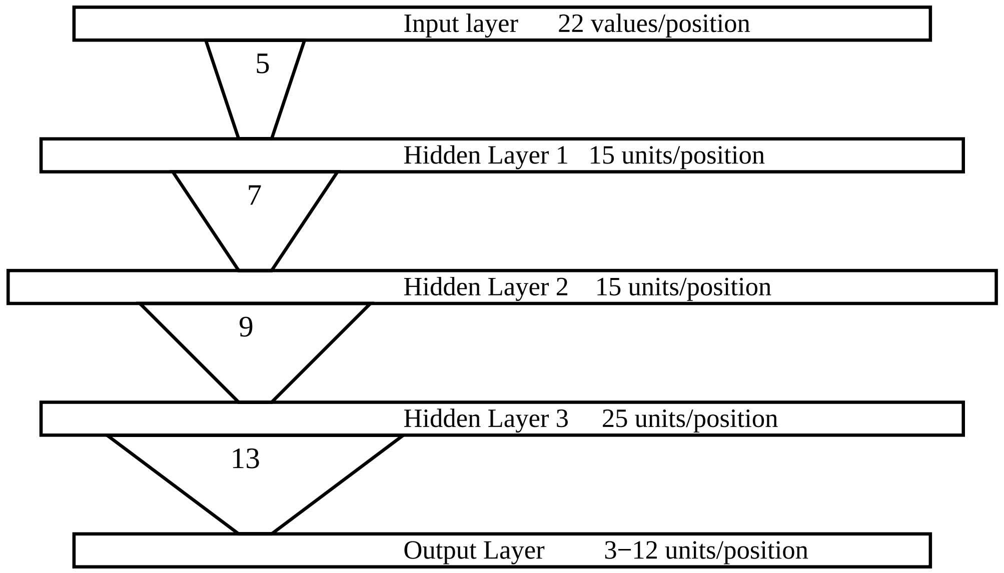
- We use neural nets to predict local properties.
- Input is profile with probabilities of amino acids at each position of target chain, plus insertion and deletion probabilities.
- Output is probability vector for local structure alphabet at each position.
- Each layer takes as input windows of the chain in the previous layer and provides a probability vector in each position for its output.

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

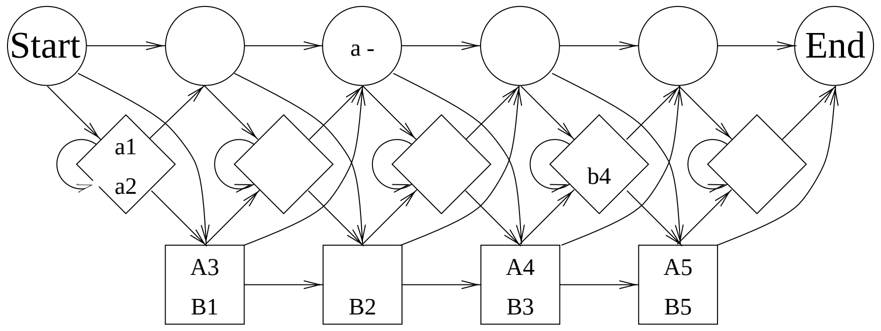



Input layer 22 values/position
5
Hidden Layer 1 15 units/position
7
Hidden Layer 2 15 units/position
9
Hidden Layer 3 25 units/position
13
Output Layer 3−12 units/position

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.

How is HMM built?

Overview of method for building a target HMM, given a single sequence (or a seed alignment):

loop: Construct a profile HMM with one fat state for each letter of sequence (or column of multiple alignment).

find: Find sequences in a large database of protein sequences that cost little with M. This is the *training set*.

Retrain M (using forward-backward algorithm) to re-estimate all probabilites, based on the training set.

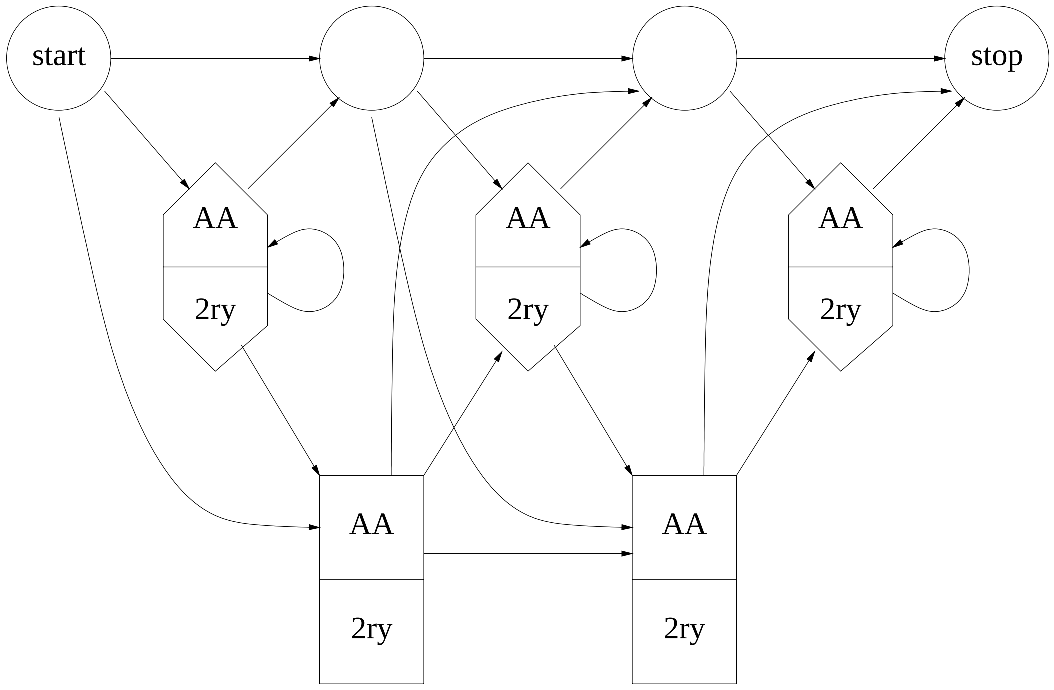
Make a multiple alignment (using Viterbi algorithm) of all sequences in the training set. The multiple alignment has one alignment column for each fat state of the HMM.

Repeat from *loop*, with thresholds in step *find* loosened.

Multi-track HMMS

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

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



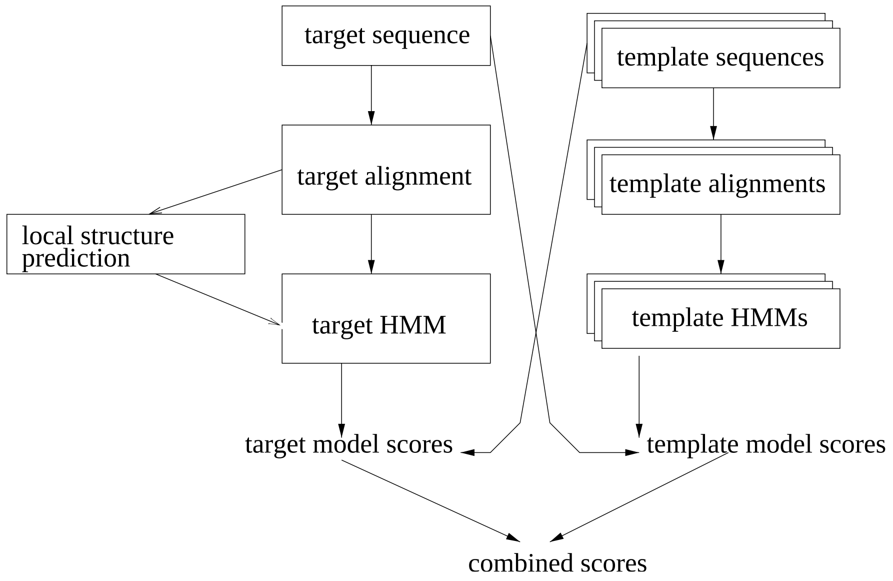
Target-model Fold Recognition

- 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 all proteins that have known structure.
- Select the best-scoring sequence(s) to use as templates.

Template-library Fold Recognition

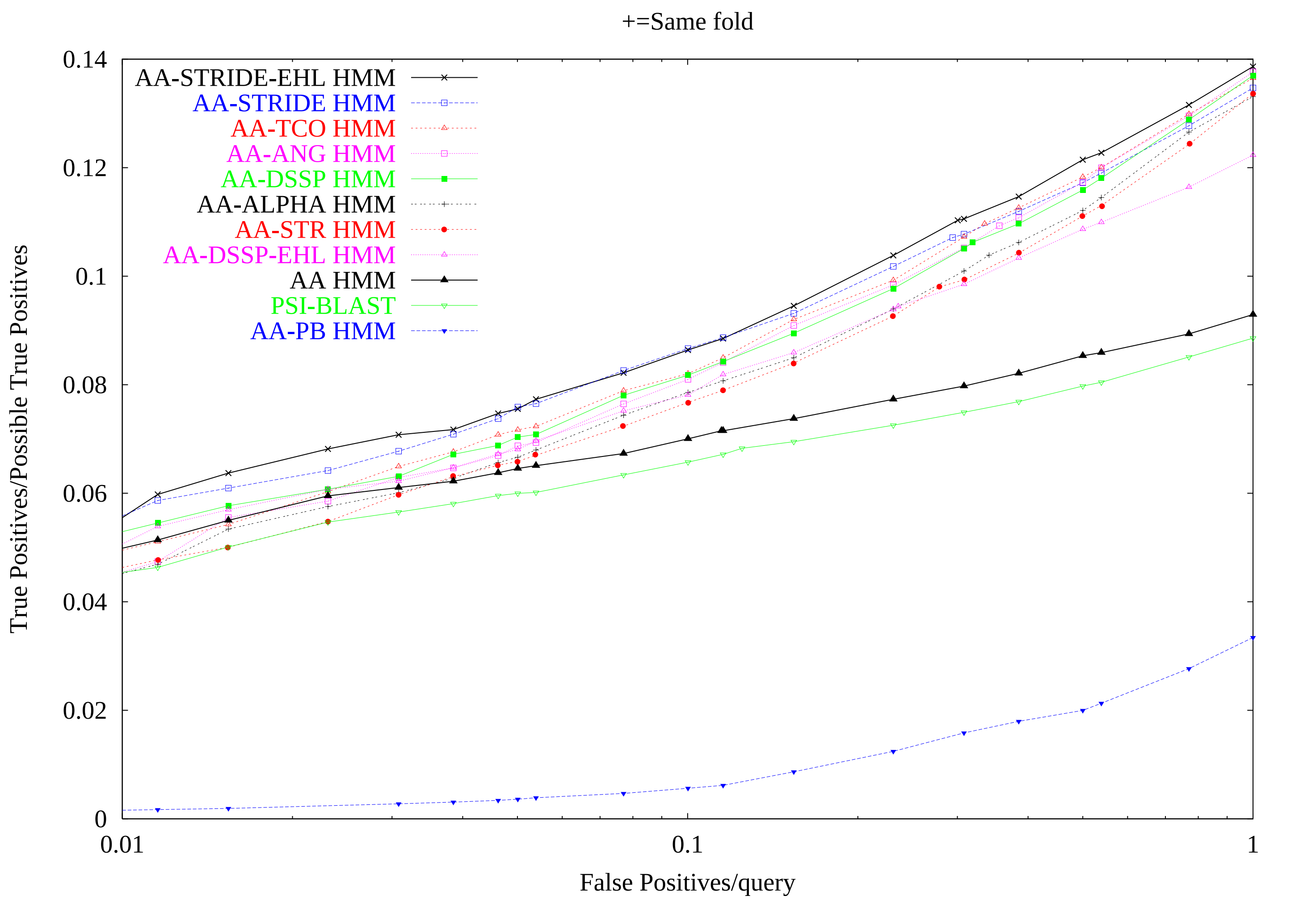
- Build an HMM for each protein in the template library, based on the template sequence (and any homologs you can find).
- The library currently has over 7000 templates from PDB.
- For the fold-recognition problem, structure information can be used in building these models (though we currently don't).
- 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



Fragment Packing

- Fragment packing was introduced by Simon and Baker's Rosetta program.
- It provides intelligent conformation generation for new folds.
- Rosetta conformation is contiguous chain.
- New conformations are created by randomly replacing fragment of backbone with different fragment (from library), keeping chain contiguous.
- Stochastic search by simulated annealing.

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 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:

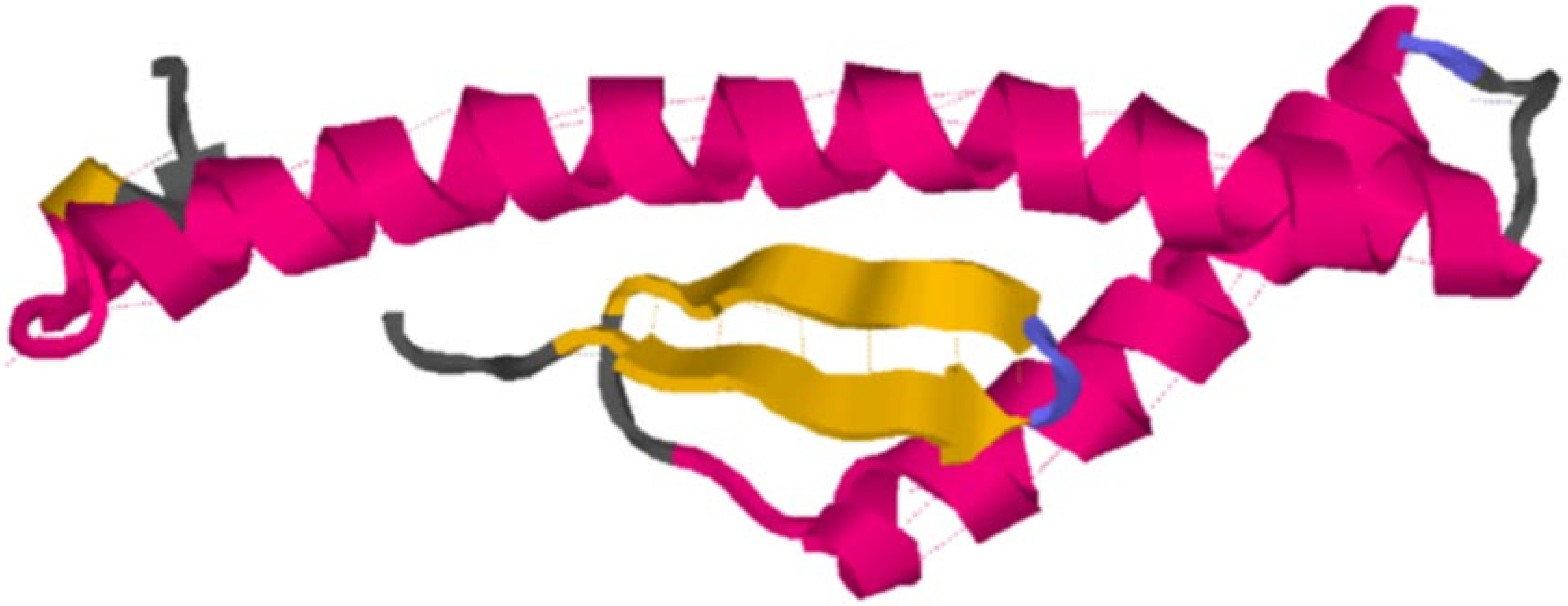
- Generic fragments (2-4 residues, exact sequence match) are obtained by reading in 500–1000 PDB files, and indexing all fragments.
- Long specific fragments (and full alignments) are obtained from the various target and template alignments generated during fold recognition.
- Medium-length fragments (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.
- Main components are variants on burial cost:
  - *Burial* is the number of atoms whose centers are in a particular sphere.
  - We define points for each residue where burial is checked.
  - We use histograms of burial conditioned on residue type to convert burial to cost ($-\log$ Prob).
- Cost function can include predictions of local properties by neural nets.
- There are currently about 20 other cost function components (clashes, disulfides, contact order, radius of gyration, constraints, ...) that can be used.

Undertaker example: T0131

Ab-initio prediction:



Undertaker example: T0147

Fold-recognition plus ab-initio prediction:

Web sites

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`

HMM **servers:** `http://www.soe.ucsc.edu/research/compbio/HMM-apps/`

SAM-T02 prediction server:

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

These slides:

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