Combining local-structure, fold-recognition, and new-fold methods for protein structure prediction

Kevin Karplus*, Rachel Karchin, Jenny Draper, Jonathan Casper,
Yael Mandel-Gutfreund, Mark Diekhans, Richard Hughey

May 29, 2003

This is a preprint of an article accepted for publication in the CASP5 special issue of *Proteins: Structure, Function, and Genetics*.

Abstract

This paper presents an overview of the SAM-T02 method for protein fold recognition and the UNDERTAKER program for *ab initio* predictions.

The SAM-T02 server is an automatic method that uses two-track hidden Markov models (HMMs) to find and align template proteins from PDB to the target protein. The two-track HMMs use an amino-acid alphabet and one of several different local-structure alphabets.

The UNDERTAKER program is a new fragment-packing program that can use short or long fragments and alignments to create protein conformations. The HMMs and fold-recognition alignments from the SAM-T02 method were used to generate the fragment and alignment libraries used by UNDERTAKER.

We present results on a few selected targets for which this combined method worked particularly well: T0129, T0181, T0135, T0130, and T0139.

1 Introduction

In previous CASP experiments, our team has concentrated on fold-recognition using hidden Markov models (HMMs) with fairly good results [1, 2, 3]. We have also had some success using standard neural-net methods to predict secondary structure [4], as measured by the EVA project [5]. In 2000, we started incorporating secondary structure prediction in our fold-recognition method for CASP4 [3].

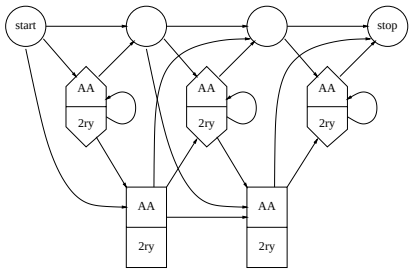

Figure 1: A multi-track HMM has multiple emission tables in each letter-generating (match or insert) state, but is otherwise similar to the standard profile HMMs used in the SAM package [7]. The multi-track HMMs model the amino acids and local structure as conditionally independent, conditioned on the state of the model.

We entered two automatic servers in CASP5 and CAFASP3: the old SAM-T99 server and a new server, SAM-T02. SAM-T02 incorporated the most important of the fold-recognition improvements we had made in CASP4: multi-track HMMs, in which each match node contains emission probabilities of predicted local structure information, in addition to amino-acid emission probabilities [3, 6]. The two-track HMM is illustrated in Figure 1.

Since both servers used the same protein sequence database and the same templates from PDB [8], any improvement in performance could be attributed to im-

*email:karplus@soe.ucsc.edu Mailing address: Computer Engineering Department, University of California, Santa Cruz, CA 95064 USA. Phone: 1-831-459-4250, Fax: 1-831-459-4829. Mail to other authors may be similarly addressed.

provements in the method, rather than in the underlying databases. The multi-track HMMs we used in CASP4 relied on a helix-strand-coil description of secondary structure, while those used in CASP5 use a variety of local structure descriptions.

For our human-assisted entry to CASP5, we added a new fragment-packing program, UNDERTAKER, to our tool set. The program was added in order to make predictions in the new-fold and difficult-fold-recognition areas, where we previously had had no success. The same method was used for all targets, independent of the degree of similarity to any targets that we found.

According to the CASP5 assessors, our group had good results in the new-fold category and the analogous fold-recognition category, so the new fragment-packing program, UNDERTAKER, will be the main focus of this paper.

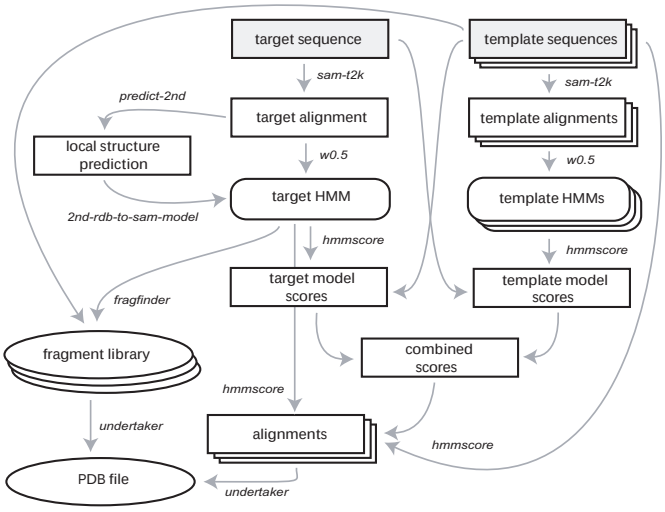

Figure 2: The SAM-T02 prediction process consists of several parts: building a multiple alignment for the target sequence using the SAM script `target2k`, predicting secondary structure with `predict-2nd`, building HMMs for the target sequence with the SAM script `w0.5`, scoring the template library against the target HMMs using the SAM program `hmmscore`, scoring the target sequence against the template HMMs with `hmmscore`, combining scores to select templates, choosing target-template alignments, generating fragments with the SAM program `fragfinder`, and doing fragment-packing with UNDERTAKER.

2 Methods

Our overall structure-prediction method can be conveniently divided into several parts:

- finding similar sequences with iterative search (using SAM-T2K);
- predicting local structure properties with neural nets;
- finding possible fold-recognition templates using 2-track HMMs (the SAM-T02 method);
- making alignments to the templates;
- building a specific fragment library for the target (with FRAGFINDER); and
- packing fragments and fold-recognition alignments to make a 3D structure (with UNDERTAKER).

The overall structure of the process is outlined in Figure 2.

The iterative search method was exactly the same as the SAM-T2K method used in CASP4—the only change was in the size of the NR database searched [9].

We predicted local structure with the same neural-net software as in CASP4, but with newly trained nets and different local structure alphabets. For CASP4, we used the standard EHL alphabet that is assessed in CASP. For CASP5 we used four local structure alphabets: EBGHSTL based on DSSP labeling [10], EBGHTL based on STRIDE labeling [11], an extension to DSSP (STR) that divided the beta strands into six classes (see Figure 3) [6], and an 11-letter alphabet (alpha11) based on the torsion angle formed by four successive C_α atoms [6]. We also provided a reduction to the 3-letter EHL alphabet for CASP assessment, but did not use this reduced alphabet for any other purpose.

As in CASP4 we used multi-track HMMs for target models, and amino-acid-only HMMs for template models. Since we now had five different target models (amino-acid-only and four different two-track HMMs corresponding to the four local structure alphabets), we did a weighted combination of scores for the template-based and target-based searches. Weights were chosen arbitrar-

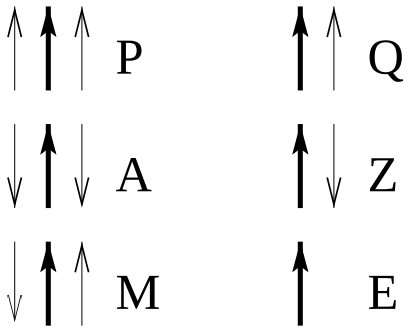

Figure 3: Six letters in the STR alphabet, which expand on the DSSP "E" or strand state. The strand of the residue being assigned is indicated with a bold arrow. In a beta sheet, this strand is either surrounded by two parallel partners "P", two anti-parallel partners "A", or one anti-parallel and one parallel partner "M". Edge strands (that have only one beta-strand partner) have either a parallel partner "Q" or an anti-parallel partner "Z". Finally, we retain the "E" label for strand residues to which DSSP assigns no partners (generally beta bulges).

ily, based on our assumptions about how well the methods would work. The combined weights were used to select templates.

The SAM-T02 automatic server did not include the alpha11 torsion-angle predictions and HMMs, but was otherwise the same as the template-selection step in our hand predictions. The automatic server made no attempt to produce 3D structures, but returned alignments to the templates based on a single two-track target HMM. The alignment HMM used our new STR alphabet as the local-structure track, since that method had performed best in our alignment tests [6]. Because our template library is highly redundant, the server attempted to remove duplication and report five distinct predictions. In some cases, we managed to get decent predictions for multiple domains as different models (for example, for T0184 with T0184_1 as models 1 and 2 and T0184_2 as models 3 and 4), even though the server did not explicitly consider domains.

Our SAM-T02-human method did not use a single alignment to the template. Instead we generated about 25 alignments for each template, using the different target- and template-based HMMs and different alignment options (global vs. local, Viterbi vs. posterior decoding). In addition to fold-recognition alignments, we also used FRAGFINDER, a new tool in our SAM tool suite, to find six fragments of length 9 for each position in the sequence. The fragments were found with a two-track HMM that uses the STR local structure alphabet, taking the best six gapless matches in the template library for each position.

Although we have not yet optimized the FRAGFINDER method nor done extensive testing, we expect that the use of two-track HMMs for finding fragments will be a big help when the local structure prediction is accurate and will be comparable to other fragment-finding techniques when the local structure prediction is weak. The HMMs will cause serious problems when the local structure prediction is confident, but wrong.

The alignments and specific fragment library were given to UNDERTAKER, along with a generic fragment library containing all 1-, 2-, 3-, and 4-residue fragments indexed by their amino acid strings from a training set of 448 monomeric protein chains. UNDERTAKER used the alignments to get an initial conformation, then applied many rounds of a genetic algorithm with randomly applied conformation-change operations to minimize a cost function.

The following subsections will describe some of the internals of UNDERTAKER.

2.1 Conformation representation

Selection of a conformation representation and data structure is critical to effective fragment packing, as it affects the computation time, the possible conformation-change operators, and the possible cost functions. UNDERTAKER represents protein conformations as the 3D coordinates for all heavy atoms (not hydrogens). Using a full 3D representation for all heavy atoms, rather than a more compact one such as ϕ-ψ angles or side-chain centroids, slows down conformation generation slightly, but allows much more flexibility in defining cost functions. One decision we plan to revisit is whether to include explicit hydrogens—having explicit hydrogens would make hydrogen-bond scoring simpler, but increases the size of the conformational space, since torsion angles for the NH_3 and OH groups would then need to be set. We could optimize the torsion angles after determining that an H-bond

was desired, but this does not seem to offer much advantage over the current implicit hydrogens.

UNDERTAKER does not require the backbone to be contiguous, but allows breaks between residues. This allows us to represent directly the multiple-segment information we get from fold-recognition alignments, bringing fold-recognition and new-fold techniques into a unified framework. For homology modeling with UNDERTAKER, unlike Rosetta [12], we do not pick a single alignment and freeze the backbone for the core residues, but allow many alignments to be sampled and parts of different ones to be combined. For distant target-template relationships, mixing several alignments can help find the correct parts, but for closer relationships, choosing a single alignment that is most likely to be good avoids adding noise to the search.

Our poor performance on comparative modeling targets (relative to our automatic server) is probably due in large part to not freezing the core.

Allowing broken backbones introduces a problem that is not present in programs (like Rosetta) that use a frozen core or contiguous backbone: what is the relationship between unconnected parts of the backbone? what moves when a backbone fragment is replaced?

To solve this problem in UNDERTAKER, we represent the protein as a tree with *segments* as leaves, where each segment is a contiguous piece of the protein with properly formed peptide bonds. When we do fragment replacement within a segment, the transformation does not propagate across the gap between segments. To preserve 3D relationships between segments, we add edges between segments, called *tertiary edges*, that indicate which pairs of atoms are thought of as holding the segments together. These are usually chosen to be the closest pair of atoms in the two segments, such as a disulfide bond or Van der Waals contact.

Any two residues in the protein chain are connected by a unique path through the tree. Removing a peptide bond or a tertiary edge breaks the tree into two trees each of which can be rigidly transformed, maintaining the structure within that subpart of the protein, without requiring a contiguous backbone or a frozen core. The rigid transformations may result from any of several conformation changing operations, described in Section 2.2.

Note that a subtree may consist of segments that are widely separated along the protein chain, as would be necessary for holding together a domain while another inserted domain changes shape.

2.2 Conformation-changing operators

We have implemented several conformation-changing operations in UNDERTAKER, beginning with the fragment replacement operation introduced by Simons and Baker [13]. Fragments to use for replacement came from three sources: very short ones (1–4 residues) from a generic fragment library, which must match exactly on all residues, medium-length ones (9–12 residues) found by FRAGFINDER, and variable-length ones that come from fold-recognition alignments. We also used an operator for replacing two fragments simultaneously, to allow for hinge-like motions of part of the conformation, though there is currently no constraint that the conformation change actually be hinge-like.

In addition to fragment replacement, UNDERTAKER has alignment replacement, which replaces several segments at once, keeping them in the same physical relationships as they have in the template they are copied from. This operator allows us to import complete fold-recognition results into our fragment-packing optimization.

UNDERTAKER includes a number of operators that attempt to improve some part of our cost function—reducing breaks, forming or improving disulfide bonds, reducing clashes, reducing the cost of user-specified constraints, and so forth. Many of these operators work by trying a small number of potential fragment replacements and computing for each the effect that it would have on only part of the cost function, selecting the fragment replacement that appears to make the most improvement.

UNDERTAKER also has operators for repositioning subtrees. It can either jiggle them a small amount or try to find the optimal placement for them, given the constraints and peptide bonds on the segments in the tree. There are various ways of splitting the tree into subtrees, which move larger or smaller sections of the protein. On a smaller scale, the method also has operators for changing the rotamers of the residues without changing the backbone, to improve packing or reduce clashes.

Since UNDERTAKER uses a genetic algorithm for the stochastic search, the method also includes crossover operators that combine parts of two conformations to get a new one.

2.3 Stochastic search

As mentioned above, UNDERTAKER uses a genetic algorithm to search conformational space. We start from a set of conformations (random conformations, based on fold-recognition alignments, or from previous runs of UNDERTAKER), and randomly apply operators to generate new conformations. New conformations that score well are added to the pool for the next generation, and poorly scoring older conformations are eliminated. To make sure that the pool mixes rapidly, we keep no more than 40% of the conformations from the previous generation.

We keep track of the success rate for each operator (how often it results in a conformation being kept in the pool) and adjust the probability of applying the operators based on their success. The adaptation scheme we are currently using is rather crude and sometimes gets stuck applying only one or two of the operators, if it has initial success with them.

We use the results of several runs of the genetic algorithm to seed the pool for another run, often getting noticeable reduction in cost from applying crossover operators to conformations from different runs.

2.4 Cost function

Substantial effort was put into making the cost function in UNDERTAKER easy to modify and extend, as it was quite clear that much future work would be put into different scoring functions. Much less effort has been put into making a good first version of the cost function. For example, our cost function does not yet include a hydrogen-bonding term, but such a term is essential for forming beta sheets. For close fold recognition targets and for alpha-helical proteins, the compactness of the right structure usually held it together in the subsequent optimization of the cost, but for more distant folds and new folds, beta sheets often came apart during optimization, even if they were present in the initial conformation. We often had to add desired hydrogen bonds as manual constraints in the cost function.

The cost function can be defined at run time as a linear combination of any subset of a large number of different basic cost functions, and the basic cost functions themselves can be parameterized at run time. New basic cost functions are very easily added to the code, and they add no computational cost unless they are specifically requested in the linear combination specified at runtime. Currently, we have over two dozen basic cost functions, and there are several more that we believe we should implement and test.

One of UNDERTAKER's most important cost functions, indeed the one that gives the method its name, is the *burial* function. This is a parameterized function that counts the number of atoms within a given radius for each residue and scores the sphere based on the probability of seeing that number of atoms. The sphere is referred to as a *spot*, and the number of atoms whose centers are within the sphere as the *burial* of the spot. The parameter files for a burial function include a specification of where the center of the sphere is relative to the residue, the size of the sphere, and the smoothed probability distribution of burial for each residue type.

The UNDERTAKER program includes functionality for optimizing the spot locations. We define *dry spots* as those for which burial has been maximized, *wet spots* as those for which burial has been minimized, and *generic spots* whose location does not depend on the type of residue. For generic spots, we maximize the mutual information between the burial and the residue identity.

UNDERTAKER also has basic cost functions that can accept the predicted probabilities over a local structure alphabet for a target and score the conformation using them (currently working only for the ALPHA11 torsion-angle alphabet).

One important basic score function accepts user-specified distance constraints on pairs of atoms and tries to satisfy these constraints while generating conformations. These constraints can come either from educated guesses by the user of the program or from experimental data (such as NMR experiments or cross-linking experiments). The use of constraints turned out to be essential for our CASP5 predictions.

3 Results and Discussion

Since our human-assisted prediction method began with essentially the same fold-recognition process as was used by our SAM-T02 automatic server, it is instructive to look at the differences in performance between the two. For the comparative modeling targets, the server did better

(according to the GDT score) on 76% of the targets—our use of UNDERTAKER without freezing the core resulted in an overall loss of model quality for the closer homologs. For the easier fold-recognition targets (classes CM/FR and FR(H)), the server did better on about a third of the targets, and the additional input, either by the UNDERTAKER program or by hand, made improvements on the remaining two-thirds. For the difficult targets (FR(A), NF/FR, and NF), the hand-assisted UNDERTAKER program did better than the automatic server on about 84% of the targets.

We looked at the results of the automatic servers registered with CAFASP, and often included the models generated by Robetta (the automatic server produced by the Baker group using Rosetta [14]) as possible conformations in the initial pool for our genetic algorithm.

On most of the new-fold targets, we did not come up with anything resembling a correct structure. This is not surprising, given the crude nature of our cost function and the amount of handwork necessary to get vaguely protein-like conformations. We will discuss only the rarer successes in this paper.

We did reasonably well for the new-fold targets T0129 and T0181. There was also an FR(A) target that we did well on (T0135), and a CM/FR target that was popular with most of the speakers at CASP5 (T0130). One target that was withdrawn from CASP5, T0139, deserves some comment. Each of these targets will be discussed below.

3.1 New fold: T0129

Target T0129 was the first target to be released, and so we had plenty of time to look at it and to adjust the scoring function of UNDERTAKER to produce more protein-like conformations. Our secondary structure predictor gave strong predictions for seven helices and for an extended piece. It turned out that our secondary-structure prediction was reasonably accurate (Q3=82%), which helped in assembling the protein.

The first three helices of the N-terminal domain were usually packed by the fragment-assembly quite consistently, but we had difficulty with the C-terminal domain. We mentally partitioned the target into two domains, but mistakenly grouped helix 4 in the second domain instead of the first, which resulted in mispacking both domains. The program was not informed of our domain division, but we selected ten hand-created constraints: four to try to keep helices 4–7 straight and six to try to form an up-down bundle of the four helices plus the extended piece from F80 to G85. We would have done better not to constrain the N-terminal end of helix 4.

Our best model was model 3, whether the domains were considered separately or together. This model was one where we liked the (incorrect) packing of helix 4 with helices 6 and 7, but did not like the way that helix 5 was messed up. Although we had included conformations provided by the Robetta server in some of our optimizations, they had not been included in the optimizations leading to models 2 and 3.

3.2 New fold: T0181

Our model 2 for target T0181 had the N-terminus basically right, but we had trouble getting the third strand of the sheet (which we had correctly predicted as a strand) to join the sheet, probably because of the large number of residues between the second and third strands. We tried adding constraints by hand to position the third strand, but we could not simultaneously form the sheet and keep the backbone contiguous. Because of the bad break in the backbone, we never submitted any of our models that had the complete sheet—these might very well have been better than what we did submit.

We had some weak fold-recognition results for T0181, but since we still have not seen the correct structure, it is difficult to decide what went right and wrong.

3.3 Fold-recognition (analogous): T0135

We submitted only one model for T0135, which we obtained through a combination of fold-recognition and new-fold techniques. Our fold-recognition method by itself had the correct fold in third place in its list of hits, but the E-value of 9.6 gave us no confidence in the result, and there were several other folds that scored essentially as well. We had no way of choosing the correct fold using just our fold-recognition methods.

When UNDERTAKER was run with no hand-added constraints, the sheet was not assembled. To assemble it, we tried to find topologies that were consistent with our predictions that strand 1 would be an anti-parallel or mixed middle strand, strand 2 would be an anti-parallel edge

strand, strand 3 would be a parallel strand, and strand 4 would be a mixed middle strand (using a neural net with our extended STR alphabet). We also wanted strands 1 and 2 to be oriented the same way, since we had predicted a single helix between them. We did not find any topologies that met all our predictions, so we experimented with adding constraints for various topologies. The most promising one was a 4132 anti-parallel sheet. We obtained a model that looked roughly like a protein to us, so we submitted it to the VAST web server [15] to see if any existing proteins had a similar structure. We got excellent alignments to proteins with a ferredoxin-like fold, probably because our library of fragments and fold-recognition alignments contained templates with this fold.

We edited VAST's structural alignments to add more fold-recognition alignments for this fold to UNDERTAKER's collection. Several runs of UNDERTAKER, both with and without constraints on the sheets, resulted in models with different flaws. We superimposed the models and did cut-and-paste editing to put together a model with better features, which we then reoptimized. We fiddled with hand-added constraints and cut-and-paste editing, to try to close gaps and pack the helices against the sheet. The final run did not use the packing constraints, but did include constraints corresponding to the hydrogen bonds of the predicted sheet, since our score function still does not include a hydrogen-bonding term.

For target T0135, the new-fold methods allowed us to recognize and align a fold that was just a little too remote for our fold-recognition methods alone to manage. Our success on this target is exactly what we were hoping for by combining methods, but several other targets in the FR(A) category were not nearly as successful.

3.4 Comparative modeling/fold recognition: T0130

Target T0130 is one that almost all the presenters at CASP5 felt obligated to present—indeed, one could almost have selected the speakers for CASP5 just based on their performance on this target!

Recognizing the nucleotidyltransferase fold was easy (almost all the fold-recognition servers got it), but getting a good alignment was harder. Most of the servers (including both of ours) did not have the third aspartic acid of the catalytic triad (D46, D48, D79)—there was excellent sequence conservation up to residue 1, but a hairpin had to be deleted from the templates to get the third strand reasonably aligned.

We added constraints by hand to keep this triad properly spaced (based on the triads in 1bpyA, 1fa0A, and 1fa0B). These constraints managed to get most of the fold for us, but we incorrectly predicted a helix for the final strand. UNDERTAKER consistently unwound the helix, but we did not think to question the rather weak predictions of the neural net on this segment and try to attach it as a strand. Instead we kept adding constraints to try to force the incorrectly-predicted helix to form and to pack against the sheet.

3.5 Withdrawn: T0139

Target T0139 had a picture of its structure published just a week before the CASP5 deadline [16]. We found the picture about 24 hours before the CASP5 deadline. We tried estimating constraints from the picture and adding these constraints to the UNDERTAKER score function. There were a number of problems creating these constraints (unlabeled atoms, mislabeled residues, and distances that were difficult to guess). We ended up adding about 40 rather loose distance constraints. Just adding these noisy constraints was not enough to get a good solution—helix 4 ended up on the wrong side of the cluster of helices 1, 2, and 3. We ended up moving the helix by hand to the other side and re-optimizing, as our move set seemed unwilling to unfold the conformation enough to change which side the helix was on, and we did not have enough time to start over from a random configuration. This re-optimization resulted in a roughly correct structure, so we did some further optimization without the constraints from the article. This re-optimization did not make many changes (our model 1 submission included the distance constraints and our model 2 submission did not).

In short, we got a good model for target T0139 (4.86 øAngstrom for all CA atoms) by adding about 40 correct but noisy distance constraints and knowledge of the chirality of the helix bundle. This was, of course, cheating, and so we informed the organizers that target T0139 should be removed from the CASP5 evaluation. Much more information could have been extracted from the article—Alexey Murzin managed to get a 3.84 Angstrom

CA-RMSD model using the same article. We were encouraged to see how little extra information was needed to go from a rather bad model to quite a good one, as one of our hopes is that the UNDERTAKER program will be useful for aiding structure determination from data sets that would normally be insufficient or of too low quality for the purpose.

4 Conclusion

The CASP5 experiment this year let us test both our new use of local-structure alphabets in fold recognition (comparing the SAM-T02 server to the older SAM-T99 server) and our new fragment-packing method.

Almost universally, the SAM-T02 server made better predictions than the older SAM-T99 server, showing that the use of predicted local structure is valuable in fold recognition.

Hand-assisted fragment packing did substantially better than the fold-recognition server on the more difficult targets, but worse on the easiest (comparative modeling) targets. This loss of performance is almost certainly due to having a large number of alignments to various templates, with no information given to UNDERTAKER about the scores of the alignments. UNDERTAKER's crude cost function was not able to pick out the best template and alignment reliably from the set it was presented with, and the fragment packing often resulted in some movement of the core residues.

Our future work will concentrate on improving the cost function in UNDERTAKER, adding new conformation-change operators, and providing a way to preserve good conformations from fold recognition without having to freeze the core.

Acknowledgments

This work was supported in part by NSF grants DBI-9808007 and EIA-9905322, DOE grant DE-FG0395-99ER62849, and a National Physical Sciences Consortium graduate fellowship. We are grateful to David Haussler and Anders Krogh for starting the hidden Markov model and Dirichlet mixture work at UCSC, as these approaches were instrumental to our success. We are also grateful to Christian Barrett and Spencer Tu Basu, who implemented earlier versions of our prediction server, and who made other contributions to the techniques.

We began work on T0129 and T0130 while Kevin Karplus was on sabbatical in David Baker's lab and conversations with members of that lab were fruitful in guiding our initial work on these targets.

References

[1] Kevin Karplus, Kimmen Sjölander, Christian Barrett, Melissa Cline, David Haussler, Richard Hughey, Liisa Holm, and Chris Sander. Predicting protein structure using hidden Markov models. *Proteins: Structure, Function, and Genetics*, Suppl. 1:134–139, 1997.

[2] Kevin Karplus, Christian Barrett, Melissa Cline, Mark Diekhans, Leslie Grate, and Richard Hughey. Predicting protein structure using only sequence information. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):121–125, 1999.

[3] Kevin Karplus, Rachel Karchin, Christian Barrett, Spencer Tu, Melissa Cline, Mark Diekhans, Leslie Grate, Jonathan Casper, and Richard Hughey. What is the value added by human intervention in protein structure prediction? *Proteins: Structure, Function, and Genetics*, 45(S5):86–91, 2001.

[4] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[5] V.A. Eyrich, M.A. Marti-Renom, D. Przybylski, M.S. Madhusudhan, A. Fiser, F. Pazos, A. Valencia, A. Sali, and B. Rost. EVA: continuous automatic evaluation of protein structure prediction servers. *Bioinformatics*, 17(12):1242–1243, December 2001.

[6] Rachel Karchin, Melissa Cline, Yael Mandel-Gutfreund, and Kevin Karplus. Hidden Markov models that use predicted local structure for fold recognition: alphabets of backbone geometry. *Proteins: Structure, Function, and Genetics*, 51(4):504–514, June 2003.

[7] Richard Hughey, Kevin Karplus, and Anders Krogh. SAM: Sequence alignment and modeling software system, version 3. Technical Report UCSC-CRL-99-11, University of California, Santa Cruz, Computer Engineering, UC Santa Cruz, CA 95064, October 1999. Available from `http://www.soe.ucsc.edu/research/compbio/sam.html`.

[8] F.C. Bernstein, T. F. Koetzle, G. J. Williams, E. E. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112:535–542, 1977.

[9] NR (All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF Database) Distributed on the Internet via anonymous FTP from `ftp://ftp.ncbi.nlm.nih.gov/blast/db`. Information on NR is available at `http://www.ncbi.nlm.nih.gov/BLAST/blast_databases.html`.

[10] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, Dec 1983.

[11] Dimitrij Frishman and Patrick Argos. Knowledge-based protein secondary structure assignment. *Proteins: Structure, Function, and Genetics*, 23:566–579, 1995.

[12] Richard Bonneau, Jerry Tsai, Ingo Ruczinski, Dylan Chivian, Carol Rohl, Charlie E. M. Strauss, and David Baker. Rosetta in CASP4: progress in ab initio protein structure prediction. *Proteins: Structure, Function, and Genetics*, 45(S5):119–126, 2001.

[13] Kim T. Simons, Rich Bonneau, Ingo Ruczinski, and David Baker. Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):171–176, 1999.

[14] Kim T. Simons, Charles Kooperberg, Enoch Huang, and David Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *Journal of Molecular Biology*, 268:209–225, 1997.

[15] J. Gilbrat, T. Madej, and S. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6:377–85, 1996.

[16] K. Fukushima, J. Kikuchi, S. Koshiba, T. Kigawa, Y. Kuroda, and S. Yokoyama. Solution structure of the DFF-C domain of DFF45/ICAD. a structural basis for the regulation of apoptotic DNA fragmentation. *Journal of Molecular Biology*, 321(2):317–327, August 9 2002.